Outlier detection based on neighborhood proximity

Nguyen, Hoang Vu

2010

Nguyen, H. V. (2010). Outlier detection based on neighborhood proximity.. Master’ s thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/42448

https://doi.org/10.32657/10356/42448

Outlier Detection Based On Neighborhood Proximity

a Dissertation
submitted to the School of Computer Engineering
of Nanyang Technological University

by

Nguyen Hoang Vu

(Matric No.: G0800360H)

in partial fulfillment of the requirements
for the degree of
Master of Engineering

June 2010

Curriculum Vitae

Hoang Vu Nguyen was born on May 24th, 1985, in Ho Chi Minh city, Vietnam. He attended Nanyang Technological University (NTU), from July 2004 to June 2008, and graduated with a Bachelor of Engineering degree (First Class Honor). He has been working for Credit Suisse AG since July 2008. He started a Master of Engineering (research-based) degree since January 2009 under Dr. Vivekanand Gopalkrishnan's supervision, his long time teacher and master. His research was on designing scalable and efficient algorithms for outlier detection. His career objective is to discover things that no one has found out before. Because of that, research seems to be the only job that may fulfill his desire.

Acknowledgments

At first, I thought I had to make everything in this thesis be formal since it will be graded. Thus, I wrote only a few lines in this section with very polished words. That is obviously not my style. After reading a few theses, I learnt that "I do not need to be always formal when going to office throughout five working days of week". At least, my company has a "jeans day" policy.

To me, Dr. Vivekanand Gopalkrishnan is very special. I did learn about research when I was in secondary school as well as high school. I read so many books on Mathematics describing the glorious scientific lives of great mathematicians. Believe me, those books are misleading! They fail to describe how difficult it is to get even one normal paper accepted (I am not talking about a shocking scientific discovery). I failed as they did, too. My papers got rejected so many times that I had ever decided I was not born for research. Understanding all of these feelings, Dr. Vivekanand had helped me to learn how to accept failure, stand up and continue fighting till succeeding. Those are not all of the things he has given to me as to enumerate them exhaustively, I need more space for this section. To sum up, I would like to thank him so much for his guidance on how to become a good person and how to pursue my research dream.

Next, I am grateful to my girlfriend for her encouragement and deep understanding. Her constant support gives me much strength to proceed my study all the way to this phase.

This study would not have been possible without the help and support of various people. Here, I would like to express my great appreciation to other following persons who have graciously, in different ways, helped me during the course of my study:

- Dr. Chang Kuiyu for being my official supervisor.
- Ang Hock Hee for his fruitful advices, suggestions, and collaboration.
- Lab technicians of CAIS, for all the technical help they have provided me.
- Staffs of SCE graduate office for all the administrative support they have given me.

Abstract

Outliers, also called anomalies are data patterns that do not conform to the behavior that is expected or differ too much from the rest. In some cases, outliers could be caused by errors in data generating/collecting methods or by inherent data variability. However, in many situations, outliers are indications of interesting events that have never been known before and hence, an adaptation of the theory to capture the new events is required to explore the underlying mechanisms. The two-side effect of outliers necessitates the development of efficient methods to detect them for either (a) eliminating/minimizing their impacts on general performance of information systems or (b) capturing the underlying interesting knowledge (e.g. intrusive connections in a network). In general, outlier detection has many practical applications, especially in domains that have scope for abnormal behavior, such as fraud detection, network intrusion detection, medical diagnosis, marketing, customer segmentation, etc.

There are many ways in practice to solve our problem of interest. This thesis deals specifically with outlier notions based on measures of neighborhood dissimilarity. Related works can be divided into two main categories: distance-based and density-based. In our study, we place our focus more on distance-based approaches. With considerations to the limitations of existing works, we propose two techniques, tackling separate aspects of outlier detection.

Different neighborhood-proximity-based techniques introduce/use different notions of outliers. That makes their performance vary greatly through disparate datasets. Therefore, using only one technique may not yield desired outcome. Motivated by the issue, we present a novel scheme for classifying and combining various outlier detectors in order to exploit their own advantages. The ensemble framework applies each detector on a randomly chosen subspace. Hence, error suffered by one detector is unlikely repeated by others. In high-dimensional spaces, subspace irrelevance is a local rather than a global property. Choosing different subspaces for different detectors helps our technique to overcome the curse of dimensionality. Extensive experiments point out that our method yields better detection accuracy than existing ones on high-dimensional datasets.

The second technique utilizes an existing well-known definition of distance-based outliers and aims to improve the temporal cost of the detection process. In particular, it is built based on the traditional nested-loop algorithm but different from existing techniques, it is coupled with multiple pruning rules. Hence, we are able to reduce the inherent quadratic cost to a cost linear w.r.t. the dataset's size. Empirical study carried out on real datasets shows that our proposed approach consistently outperforms state-of-the-art techniques on distance-based anomaly detection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Outliers, also called anomalies are patterns that do not conform to the behavior that is expected or differ too much from the rest. While there is no universally accepted definition of what outliers are, the notion of outliers introduced by Hawkins et al. [40] captures the spirit: "An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism". Examples of outliers abound in social as well as scientific contexts. We usually hear about UFOs (Unidentified Flying Objects). Under Hawkins' definition, UFOs can be considered as outliers since they are very different from any flying entities that human has ever invented. There are ten students in a research lab; nine of which study data clustering while only one carries out research on outlier detection. That only student is in fact an outlier in his research lab.

The task of detecting outliers includes identifying those abnormal patterns and if possible, extracting some interesting knowledge contained in them to support the learning process. Outliers arise because of various reasons such as human error, instrumental error, natural deviations in populations, fraudulent behavior, changes in behavior or faults of systems. In many cases, outliers are simply detected and discarded to ensure the cleanliness of data (i.e. no further investigation is required). However, outliers could also be indication of interesting events that have never been known before and hence, detecting outliers may lead to the discovery of critical information contained in data. In such cases, uncovering underlying cause(s) is necessary. For example, if one happens to find an UFO, throwing it away is obviously not a good idea. Studying its structure to understand its flying mechanism is certainly much more interesting and beneficial.

The problem of detecting abnormal events, also called outliers has been widely studied in different research communities as rare classes mining [48], exception mining [76], novelty detection [29, 22, 34, 69], outlier detection [25, 40, 49, 67, 77, 10, 79, 47, 73], etc. It also has many applications in various areas such as network intrusion detection, medical diagnosis, image processing, etc. This highlights the importance of studying outlier detection.

1.2 Challenges

From the time when very first publications on abnormality detection (e.g. [40]) appear in the literature, research on the problem has evolved tremendously in various domains. The field of data mining itself has also observed significant development for solutions on outlier detection. They range from exploring new ways to define abnormality [49, 68, 15, 25, 67], minimizing the computational costs (CPU and I/O) for detecting outliers [68, 20, 15, 13, 78, 46], mining anomalies in data streams [65, 12], ensemble outlier detection [56], subspace outlier mining [61], to designing parameter-free detection algorithm [23].

Despite the advances seen so far, many issues of outlier detection are left open or not yet completely resolved. In the following, we present some of the key challenges that we have to consider whenever developing a detection solution. They pertain to almost all existing methods in the field and are hard to be resolved completely.

1.2.1 Distance Function

Distance functions are used to measure the dissimilarity between any two data records. They thus play a very important role for techniques working based on the measurement of similarity among objects. In general, a distance function is required to satisfy the triangular inequality to facilitate the searching as well as pruning processes. That makes the underlying set of data objects along with the distance function become a *metric space*. Most techniques, not only in outlier detection, specialize the metric space to *vector space* since a vector space besides preserving the triangular inequality property also contains some geometric properties which are not available in a general metric space [27]. In a vector space, data records are represented in the form of vectors so the cost of distance computation between any two vectors are linear to the dimension of the space. Hence, distance computation in vector spaces becomes simpler than in general metric spaces which normally require some indexing techniques to reduce the heavy workload of distance measurement. In the field of outlier detection, many approaches claiming to work in any metric space such as those in [53, 78] are however implemented using Euclidean distance, a typical distance function for vector space. There are a variety of distance functions for vector space utilized besides Euclidean, such as Minkowski or Mahalanobis distance functions. Given two data points $p_1 = (a_1, a_2, \ldots, a_{dim})$ and $p_2 = (b_1, b_2, \ldots, b_{dim})$ in a *dim*-dimensional dataset, the Euclidean distance between p_1 and p_2 in the simplest format has the following form:

$$D(p_1, p_2) = (\sum_{i=1}^{dim} (a_i - b_i)^2)^{1/2}$$

The Euclidean distance is in fact a special form of Minkowski distance. The Minkowski distance of *order* m (m-*norm* distance) is defined as follow:

$$D_m(p_1, p_2) = (\sum_{i=1}^{dim} |a_i - b_i|^m)^{1/m}$$

When $m \to \infty$, we have the infinity-norm distance:

$$D_\infty(p_1, p_2) = \max_{1 \le i \le dim} (|a_i - b_i|)$$

On the other hand, the Mahalanobis distance between p_1 and p_2 is defined as:

$$D(p_1, p_2) = \sqrt{(p_1 - p_2)^T \Sigma^{-1} (p_1 - p_2)}$$

where $p_1 - p_2 = (a_1 - b_1, a_2 - b_2, \ldots, a_{dim} - b_{dim})$ and $\sum$ is the sample covariance matrix of the normal data.

While Euclidean as well as Minkowski distance is used widely in outlier detection, their performance degrades if normalization is not carried out for dataset with attributes of various scales or containing linear correlation. Mahalanobis distance on the other hand is effective when there exists a linear correlation among features [45]. However, as pointed out by Lazarevic et al. [55], Mahalanobis distance performs poorly in applications where the underlying dataset contains more than one distribution. Such kind of datasets however is very common in practical applications.

Despite of the simplicity and advantages of using distance functions that can make up vector spaces, not in all datasets can we construct such a space, especially for multimedia data [27]. In such cases, distance computation is far from inexpensive. Indexing techniques are then developed to mitigate the problem. The fact that they add up additional overheads (both in time and space) pose some impacts on the design of an efficient detection technique. This has been addressed recently in [13]. However, more work is needed to handle the issue associated with general metric spaces.

1.2.2 Pattern Representation

Selecting relevant features for the detection process, transforming features into other features, etc. are examples of the tasks that need to be accomplished in order to find a relevant representation for data records.

Most of detection techniques present data in the forms of feature vectors where each record is represented as a multi-dimensional tuple. That stems from the fact this representation facilitates the distance computation. The features of each vector can be *quantitative* or *qualitative*. Quantitative features can be further divided into *continuous* and *discrete* values, while qualitative features can be divided into *nominal* and *ordinal* ones.

Most real-life datasets are multi-dimensional and similar to other data mining applications, choosing a suitable subset of features to use in the detection process is desirable. However, the scope of outlier detection is to uncover *abnormal* patterns, pruning out some specific attributes prior to the detection process might not be a good idea since by discarding some certain features, we may lose some interesting knowledge (outliers may only be discovered when we look at some certain projections of data points on some dimensions [7]). While feature selection focuses on selecting a relevant set of features for the mining task, feature extraction calculates new features based on the original data attributes. Once again, careful attention is needed to ensure the validity of the detection outcome yielded by the new set of features [64].

Other methods of data transformation also play important roles in the detection process. Here, we discuss two issues of data transformation which are data normalization and data type transformation. Data normalization is very crucial, especially when Euclidean distance is used. In a general dataset, features have different scales. Normalizing features to the same scale helps to avoid the issue of feature bias. As with data type conversion, since distance functions normally manipulate on numerical values [66], additional work is required to convert categorical features into numerical ones. The most common method utilized is *inverse document frequency* which has previously used in outlier detection problem [56, 20]. However, as shown in [66], this process is shown to be ineffective since it is possible to create two different distance functions for categorical and numerical attributes respectively and combine them in a single application. The problem of detecting outliers in categorical datasets has also been addressed in [28].

1.2.3 The Need of Labeled Data for Training and Testing

Detection techniques once built need to be trained and validated on datasets. Training is required for various purposes such as identifying good thresholds for some input parameters to the respective detection algorithms. On the other hand, the need of data for testing is also an issue. Detection techniques may be created to serve optimization requirements (e.g. optimizing time and space). Nevertheless, the core service that a detection technique provides is uncovering outliers. After building a method, the most important thing we are concerned is how good it is in identifying anomalies. Consequently, to evaluate the quality of outliers, we require labeled data. If labeled data is not available, human interpretation can be employed. The subjectivity of this evaluation method nevertheless prevents it from becoming a common choice. Thus, labeled data are still required. A well-labeled set of data allows us to compare our detected result against the actual labels of data points. However, not all labeled datasets are directly suitable for outlier detection problem. Although conversion methods can be used (Section 3.3), we really need datasets that are specifically designed for outlier detection. Stemming from this need, synthetic datasets are generated. Whereas some can be standardized (methods for generating are available) such as Mixed Gauss dataset [13], their use raises the validity issue since such datasets rarely appear in practical applications. We can also generate outliers based on the normal behavior observed from real datasets. This method however also has its own problem since in general outliers do not have common behavior, i.e., it is impractical to devise mechanisms to generate outliers. The problem of having well-labeled datasets for testing is still an open issue.

1.3 Motivation

Outliers could be caused by errors in data generating/collecting methods or by inherent data variability. However, in many situations, outliers are indications of interesting events that have never been known before and hence, an adaptation of the theory to capture the new events is required to explore the underlying mechanisms. The two-side effect of outliers necessitates the development of efficient methods to detect them for either (a) eliminating/minimizing their impacts on general performance of information systems or (b) capturing the underlying interesting knowledge (e.g. intrusive connections in a network).

Eliminating noise in data benefits many practical applications, one of which is data clustering. Many clustering techniques, like K-means, have their performance significantly impacted due to noise present in the data [45]. Thus, purging the noise before performing the clustering task is very crucial for the final outcome, which in turn helps us to gain some insights about the data structure. In other words, outlier detection as a noise elimination step indirectly nurtures the mining process by providing clean information where mining techniques can be applied while not being impacted by data irregularities.

Nevertheless, nowadays, outlier detection is primarily studied as an independent knowledge discovery process merely because outliers might be indications of interesting events that have never been known before. For example, in network intrusion detection, attacks are always present and evolve over time. Detecting attacks by means of outlier detection methods and using them to derive categories of intrusions help us to better enhance our security mechanisms. In fraud detection, detection methods are utilized to capture suspicious activities of credit card usage and hence, prevent successive damage by giving recommendation to block such accounts. There are many more such fields of applications where outlier detectors are employed as tools for learning new knowledge underlying the vast information collected, e.g., customer segmentation, medical diagnosis, to name a few.

Despite the great demand for outlier miners, it is in fact very difficult to derive a notion of outliers which fits well with a specific application domain. Further, with the emergence of new types of data (e.g., streams), outlier detection researchers are faced with new challenges requiring them to keep devising new notions to effectively mine the nuggets. Besides, like many other data mining tasks, ensuring accuracy only is far from enough in many situations - efficiency is also an important criterion, especially when dealing with huge datasets. As for high-dimensional data, existing techniques suffer *the curse of dimensionality* which limit their applicability to the corresponding domains. This necessitates the development of relevant approaches to handle the issue. These are to point out that outlier detection is a very active field of data mining research and an extensive study will bring many benefits to various practical applications as mentioned above.

In our work, we primarily focus on subspace outlier detection and improving the detection process' efficiency. While the latter aspect is very important for almost all data mining tasks, the former stems from the fact that outliers are located in locally varying subspaces. Thus, mining outliers in subspaces will help us to better solve the curse of dimensionality and uncover interesting knowledge.

1.4 Our Problems of Interest and Solutions

We place our research on outlier detection based on neighborhood proximity measurements. In other words, we focus on outliers defined by their relative distances to nearest neighbors. Defining outliers based on nearest neighbors is a popular topic in the field and has been studied for a long time. Many techniques have been proposed so far, e.g. [49, 68, 15, 25]. They nonetheless suffer some cons limiting their applicability to practical applications: low accuracy [7], and high execution time [14] in high-dimensional data.

1.4.1 Subspace Outlier Mining

Existing techniques usually compute distances (*in full feature space*) of every data sample to its neighborhood to determine whether it is an outlier or not [7, 25, 49, 67]. This causes two side-effects. First, for high-dimensional datasets the concept of locality as well as neighbors becomes less meaningful [21]. Second, not all features are relevant for outlier mining. More specifically, popular distance functions like Euclidean and Mahalanobis are extremely sensitive to noisy features [55]. Despite the presence of the curse of dimensionality, it is difficult in practice to choose a relevant subset of features for the learning purpose [7, 43, 56].

While the nature of data is unpredictable, there is a need for an efficient technique to combine different outlier detection techniques to overcome the drawback of each single method and yield higher detection accuracy. The motivation here is similar to the advent of ensemble classifiers in the machine learning area [43, 52]. With the feasibility of ensemble learning and subspace mining demonstrated, the natural progression would be to combine them both. Lazarevic et al. [56] propose the first solution for *semi-supervised* ensemble outlier detection in feature subspace. That work assumes the existence of outlier scores where a *combine* function can be applied directly. However, this is not practically true since different detection methods can produce outlier scores of different *scales*. For example, it can be recognized that the scores produced using k^{th} Nearest Neighbor Distance-based Outlier [68] are smaller in scale than those using Cumulative Neighborhood [11]. Furthermore, as pointed out in Section 3.2.4.1, different detection techniques also produce different *types* of score vectors. In particular, some vectors are real-valued while others are binary-valued. This leads to the need of a unified notion of outlier score and an efficient technique to specifically deal with scores' heterogeneity. The availability of such notion would facilitate the task of combination.

In order to address this problem, we present the *Heterogeneous Detector Ensemble on Random Subspaces* (HeDES) framework. In HeDES, each member detector operates on a randomly sampled subspace. Therefore, error suffered by one detector in some subspace is unlikely repeated by others on other subspaces. Furthermore, the fact that different detector works on disparate subspaces instead of the full-dimensional space helps HeDES to overcome the curse of dimensionality. Additional advantage of using HeDES lies in its ability to incorporate various heuristics for combining different types of score vectors. Extensive empirical studies show that the HeDES framework can outperform state-of-the-art detection techniques and is therefore suitable for outlier detection in real-world applications.

1.4.2 Reducing Temporal Cost of Distance-based Outlier Detection

Distance-based detection is an important branch of proximity-based outlier detection. Related techniques usually involve in computing data points' nearest neighbors, which is very time-consuming (time complexity is $O(N^2)$ with N being the dataset's size). Therefore, majority of related works proposed aim to introduce algorithms with very low time complexity. Among them, pruning outlier searching space and computation reduction are dominant. Computation reduction techniques [68, 37, 20, 15, 11] usually try to limit the number of detected outliers (e.g. top n outliers), and employ similar data structures used in Ramaswamy's index-based algorithm [68]. In particular, a list of top n outliers and the minimum outlier score found so far are employed to help reduce the computational cost. Bay et al. [20] provide detailed analysis for this type of algorithm (nested-loop) and find out that in average case, the time complexity becomes linear with the dataset's size. However, such linear computational cost can only be obtained when the dataset contains many outliers, which is impractical [37]. Otherwise, its computational cost becomes to $O(N^2)$.

Motivated by the need of an efficient method for mining distance-based outliers, we improve the nested-loop algorithm and propose a method that is able to detect distance-based outliers in nearly $O(N)$ time, regardless of how many outliers there are in the considered dataset. Our approach is a two-phased *MultI-Rule Outlier* (MIRO) detection approach using the outlier scoring criterion proposed in [15]. In the first phase, we partition the data into clusters, and make an early estimate on the lower bound of outlier scores. This phase prunes clusters that cannot have outliers, and the second phase then processes the remaining clusters using the traditional nested-loop algorithm. Here two pruning rules are utilized: (a) first triangular inequality on the data point's outlier score is used, and then (b) the outlier score is compared with the minimum score required to be an outlier. The second check is similar to that of ORCA [20]. However, while ORCA starts with a cutoff of 0, in MIRO, the initial cutoff is obtained from the first phase, and hence converges faster. Though the pruning rules seem simple, their combined effect is strong and efficiently reduces the search space.

1.5 Organization

The rest of this report is organized as follows:

- Chapter 2 provides a background study on the solutions for outlier detection in the literature. The study encompasses many aspects of the problem ranging from notions of outliers to theoretical analysis of existing techniques in the field.

- Chapter 3 presents our solution, HeDES, towards ensemble outlier detection on subspaces for improving accuracy. Taking into account the limitations of existing approaches, in HeDES, we propose a comprehensive framework for combining detection results of heterogeneous detectors on random subspaces. Empirical studies on real datasets show that HeDES is comparable with other prominent notions of outliers in terms of detection quality.

- Chapter 4 introduces another technique developed by us for outlier detection. While HeDES targets at improving the detection accuracy, this method, called MIRO, tackles another aspect of the problem: reducing the computational complexity of detecting distance-based anomalies using the outlier notion proposed in [15]. Along side with introducing our proposed approach, the presentation is coupled with solid theoretical as well as empirical studies aiming at demonstrating the efficiency of MIRO compared to related outstanding techniques in accomplishing the same task.

- Chapter 5 presents the future work that will be addressed in the prospective study of the student.

- Chapter 6 concludes this report.

Chapter 2

Background and Related Work

In this chapter, we review the background of the problem of outlier detection. The structure of our discussion is as follows. We first discuss some definitions of outlier. Following are outlier detection's applications. Next, available metrics for evaluating existing techniques are covered. We then provide a way to classify them. Finally, we summarize the characteristics/limitations of existing methods. The materials in this chapter serve as solid background to understand the subsequent chapters in this report.

2.1 What Are Outliers?

Apart from Hawkins' definition about outliers mentioned in Chapter 1, there are also two other similar definitions which are introduced in [38, 19]. They are shown in Definitions 2.1 and 2.2, respectively.

Definition 2.1 [Grubbs' outlier definition] *An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.*

Definition 2.2 [Barnett and Lewis' outlier definition] *An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.*

These definitions capture the meaning of outliers from a general point of view. To design solutions for mining outliers, we need operational notions; those that allow us to design detection algorithms conveniently. However, since the nature of the detection problem is dependent on the application domain [44], across domains and even within each domain, the existing notions of outliers vary greatly. Definitions 2.3 (by Knorr et al. [49]) , 2.4 (by Ramaswamy et al. [68]), 2.5 (by Angiulli et al. [15, 11]) defining outliers from the *distance-based* domain (which is discussed elsewhere later) point of view illustrate such variance.

Definition 2.3 [r-neighborhood Distance-based Outlier] *An sample p in a dataset DS is a outlier if it has less then P objects lying within distance r from p.*

Definition 2.4 [k^{th} Nearest Neighbor Distance-based Outlier] *For a positive integer k and a data point $p \in DS$, let $D^k(p)$ denote the distance between the k^{th} nearest neighbor of p and p. Then given two positive integers k and n, a point p is in the top n outliers of DS if no more than $n-1$ other points in DS having a higher value of D^k than p.*

Definition 2.5 [Cumulative Neighborhood] *Let $w_k(p)$ be the sum of the distances from a point $p \in DS$ to its k nearest neighbors. p is the n^{th} outlier with respect to k in DS if there are exactly $n-1$ points $q \in DS$ such that $w_k(q) \leq w_k(p)$.*

From these definitions, it can be observed that even within a domain there are so many ways in practice to define what outliers exactly are. In fact, the problem of defining a unified notion of outliers is nontrivial.

2.2 Applications of Outlier Detection

Though it is difficult to unify the existing definitions about outliers, outlier detection plays a very important role in various emerging applications. Consequently, it can be observed that solutions for detecting anomalies have been shifted significantly from simple statistical methods [38, 40, 19] which are only applicable to data with low number of dimensions to techniques that are able to deal with large and high-dimensional datasets [20, 37, 46, 78], and even data streams [6]. Some examples of outlier detection applications are:

- In data clustering, outliers are detected to ensure a robust outcome of the clustering process [30, 62, 81, 9, 42, 72, 80].
- In network intrusion detection, anomaly connections are identified and learnt to ensure security of the whole system.
- In fraud detection, fraudulent transactions are captured to prevent abuse of stolen credit cards or handphones.
- In medical diagnosis, monitoring sudden changes in heart-rate of patients is an application of outlier detection.
- In stock markets, detecting changes in stock prices may signal special events of the economy, such as the current housing slump in US has caused significant reduction in market values of financial companies which signals a possible economic recession [1].

- Especially, in databases, the task of identifying outliers helps learn about new knowledge contained inside the novel records or eliminate noise caused by errors in collecting methods.

For a better view of outlier detection's applications, we recommend readers to explore the survey in [44] or the newer one in [26] for more information on contemporary approaches.

2.3 Available Metrics for Evaluating Detection Techniques

Similar to other data mining research, an evaluation framework is important to assess the effectiveness and accuracy of a specific outlier detection method. According to the particular issue(s) that a technique focus on, e.g. execution time, one or more relevant metrics are required to assess its efficiency. Popular metrics that are currently employed to evaluate outlier detection techniques are discussed below.

2.3.1 ROC Curve

The most widely used tool to assess detection techniques' accuracy is ROC (Receiver Operating Characteristic) curve [55]. This curve shows how detection rate changes as false alarm rate varies from 0% to 100%. The definitions of detection rate and false alarm rate are shown in Table 2.1. This table actually represents a typical *confusion matrix*. From Table 2.1, we have:

$$Detection\ rate = \frac{TP}{TP + FN}$$

$$False\ alarm\ rate = \frac{FP}{FP + TN}$$

Intuitively, detection rate gives information about the number of correctly identified outliers, while the false alarm rate represents the number of outliers misclassified as normal data records. The ROC curve illustrates the tradeoff between the detection rate and the false alarm rate and is typically displayed on a 2-D graph, where false alarm rate and detection rate are plotted on the x-axis, and y-axis, respectively. An example of ROC curve is presented in Figure 2.1. Ideally, the ROC curve has 0% false alarm rate while having 100% detection rate. However, such kind of curve is hardly achieved in practice. Hence, different pairs of (false alarm rate, detection rate) are computed to construct the curve. For a good detection technique, as the false alarm rate increases, the detection rate should increase. In other words, the closer the curve follows the left and the top border of the unit square, the more accurate the method is. In addition, the area under the curve (AUC) can also be used to measure the the considered method's ability

	Predicted as Outliers	**Predicted as Normal**
Actual Outliers	True Positives (TP)	False Negatives (FN)
Actual Normal	False Positives (FP)	True Negatives (TN)

Table 2.1: Classification of the Query Objects.

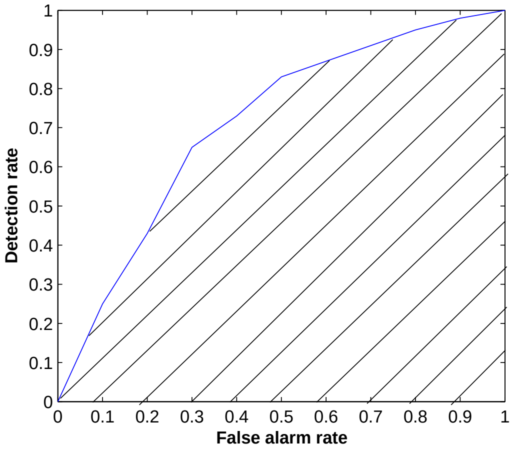


Figure 2.1: An example of ROC curve. The shaded region represents for the area under the curve (AUC).

in separating normal data records from outliers. Based on the discussed characteristics of ROC curve, the larger the value of AUC is, the better the detection technique is. The AUC of the ideal ROC curve is 1, whereas in general AUCs of detection algorithms are less than 1. As shown in Figure 2.1, the shaded area corresponds to the AUC of the displayed ROC curve.

However, this metric has an inherent drawback: it is only applicable when the knowledge about datasets (e.g. which records are outliers or normal points) is already known [56, 4, 11, 55]. In other words, the tested datasets must be well *labeled*. For labeled datasets that contains more than two classes of data or do not directly correspond to anomaly detection problem, a preprocessing step is required to convert it into *binary-class* data. Normally, one or more classes whose total cardinality is less than a specified threshold (e.g. 10% of the original dataset size) are chosen as outliers while the remaining classes are merged together to form the group of normal data points. This procedure is first applied in outlier detection problem by Lazarevic et al. [56] and reused in [4, 18].

As mentioned above, to construct the ROC curve for a detection technique, it is compulsory to have different tuples of (false alarm rate, detection rate). This can be done by varying the value of one parameter that is used as user input in the proposed algorithm, e.g. the number of nearest neighbors k, while fixing the values of the remaining ones. Finally, interpolating may be employed to produce enough tuples for the curve. This method is employed in [11].

2.3.2 Execution Time

Similar to ROC curve, the plot for execution time is a 2-D graph where the x-axis usually represents the value of a parameter used in the proposed algorithm while the y-axis stands for the execution time. To construct such a plot, we also need to vary the value of one factor and keep the remainings fixed to obtain different values of the technique's execution time. The plot of execution time expresses the sensitivity of the proposed technique on parameters that affect its performance, e.g. the dataset's size. It is therefore used to assess methods' scalability. Among the existing work, the execution time metric is widely used, e.g. in [68, 46, 20, 37], since designing methods with inexpensive computational cost is always an issue in data mining applications.

2.3.3 Analyzing The Meaning of Outliers Detected

While ROC curve can be employed whenever the tested dataset is labeled, this metric is often used for datasets whose outliers are not known before or cannot be converted to binary-class problems. More specifically, when outliers are unknown, the only way to assess the quality of a detection technique is to analyze how meaningful and intuitive the detected outliers are. Examples of techniques employing this metric can be found in [7, 16, 67]. Recalling the experimental result performed in [7] on Arrythmia dataset in the UCI machine learning repository [2], when analyzing the detection result, it is found that one of the outliers has the following attribute values: *height=780 cm* and *weight=6 kg*. The Arrythmia dataset consists of 279 attributes corresponding to different measurements of physical and heart-beat characteristics that are utilized to diagnose arrythmia. Therefore, the anomalous point detected contains attribute values that are totally not compatible to standard human measurements. In other words, that point is clearly an outlier.

The advantage of this metric is that it is usually based on human sense and expert knowledge on the domain where the technique is applied. Therefore, to some certain extent, it is more intuitive than ROC curve. However, the major problem of this evaluation metric is its difficulty for comparing different detection approaches because of its human-subjectivity characteristic. Furthermore, it does not scale well for applications with large number of data points where the number of outliers detected though about 1% of the dataset size, is still high. In such cases, it is very time-consuming and nearly infeasible to assess the detected outliers one by one.

2.4 Classification of Existing Techniques

In general, existing detection methods can be classified to: *supervised, semi-supervised, and unsupervised*. Supervised techniques [60, 31, 57, 58, 59] make an assumption that the domain knowledge on both normal and abnormal data exists and can be used to build a data model. That model is then used to classify data points as normal or outliers depending on how well they fit into it. This is analogous to *supervised classification*.

Semi-supervised techniques [32, 41] require that labels on normal data do exist. Outliers are then those deviating from the identified normal behavior. This approach however tends to classify previously unseen yet normal records as anomalies, causing unnecessarily high false alarm rate. Therefore, similar to supervised approaches, semi-supervised ones also suffer the curse of *concept-drift*. A typical example for illustrating this drawback of semi-supervised approaches is taken from the current credit-crisis which creates unprecedented turmoil in the history of global financial market. Before this disaster, the term *credit-crunch* is rarely mentioned or even does not appear in our daily life. With the collapse of the banking industry as well as the lack of confidence among banks, this term is now widely used because of its unexpected popularity to describe a situation where credit is nearly *"dry"* in the market. If using semi-supervised approach to detect anomalous events in the financial market, the credit-crunch event will always be classified as outlier despite of its so frequent appearance nowadays. That obviously causes loss of critical knowledge.

Unsupervised techniques [25, 49, 68] do not make any explicitly assumption about available knowledge. Each of them introduces or uses a specified notion of outliers, and then exploits it as a key criterion to mine outliers. Some of the problems associated in unsupervised methods include notion quality (how good the notion is), time/space complexity of the proposed techniques, and the method's accuracy. The metrics for assessing how good an unsupervised approach is in handling those problems are analyzed in Section 2.3. In the context of this report and our research, unsupervised techniques are explored in more details than the others because of its popularity. The remainder of this section is also devoted for the classification of unsupervised approaches.

We observe that unsupervised detection techniques are usually classified into four groups: statistics-based approach, clustering-based approach, distance-based approach, density-based approach. However, the method by Aggarwal et al. [7] proposes a very different approach compared to the rest. It defines outliers as those data points which are present in some abnormally low density regions that are formed by taking the combination of feature ranges. We name this method as evolutionary-based approach to express its distinction compared to the remaining existing ones. The details of outstanding approaches are given below.

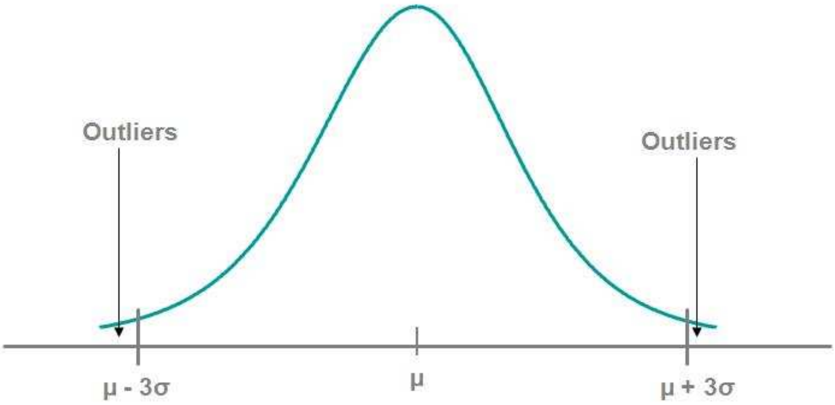


Figure 2.2: Gaussian distribution.

2.4.1 Statistics-based Techniques

Statistics-based techniques are usually discussed in books about statistics such as the one in [40]. In a typical statistics-based approach, the considered data are assumed to follow a standard distribution (Gaussian, Poisson etc.) and outliers are points that deviate from the distribution model (this is done by a statistical *discordancy test* with two hypotheses: a working hypothesis and an alternative hypothesis). A simple example of a statistics-based approach is shown in Figure 2.2. The data here is assumed to follow Gaussian distribution whose mean $= \mu$ and standard deviation $= \sigma$. Outliers are then those that lie more than 3-standard deviation from the mean. In other words, outliers are those points that lie outside the range $[\mu - 3 \cdot \sigma, \mu + 3 \cdot \sigma]$. If the underlying distribution is not known before, a searching process is required to find out the best model to fit with the data. But this process is very time consuming and does not always work, especially for data that come from different sources with different distributions. Furthermore, most of the distribution models typically are univariate. Therefore this approach is not suitable for high-dimensional datasets. Also, for many KDD applications, the underlying distribution is unknown [25]. Related work about statistics-based can be found in [19, 70, 38, 54]. This research direction has now become inactive.

2.4.2 Clustering-based Techniques

Clustering-based techniques are based on the assumption that normal data points belong to large and dense clusters while outliers do not. The common framework for such methods are as follows:

- Perform a clustering process on the data

- Analyze the clusters obtained to assess their significance
- Outliers are objects that do not fit into any clusters or belong to clusters with low support

From the above framework, it can be observed that usually outliers are by-products of these techniques since the main purpose of clustering is to figure out data clusters [45, 17, 30, 81]. Hence in nature, clustering methods aim to optimize the clustering process and outliers are simply assumed to be background noise. An example of this category that can be found in [17] typifies the common characteristics of those techniques:

- An explicit definition of outliers is not presented
- Since an exact definition of outliers is not defined, outliers are simply background noise
- The outliers detected will simply be discarded without any further investigation

Since these methods are not mainly used for outlier detection, further discussion will not be provided.

2.4.3 Distance-based Techniques

The best way to describe distance-based methods is to use the related outlier definitions. There are currently three outstanding definitions associated with distance-based techniques which can be found in [68, 49, 15]. The details are given in definitions 2.3, 2.4 and 2.5. Definition 2.3 is proposed by Knorr et al. [49] and further explored in [50, 51]. The two remaining definitions are introduced by Ramaswamy et al. [68] and Angiulli et al. [15, 11], respectively. Distance-based outlier detection techniques in general exploit distances of data points to their corresponding neighborhood to flag outliers. The distance, also called outlier score, can be computed using only one neighbor [68] or k nearest neighbors [15, 11]. It can simply be used to count the total number r-neighbors, i.e. the number of data points within distance r, of each data point [49]. Normally, distance-based techniques do not assume any distribution of the data. However, they suffer expensive computational cost of searching nearest neighborhood. This limitation has recently motivated researchers to develop more efficient techniques with lower time complexity [20, 37, 15, 13, 78]. These have excellent applicability for large and multi-dimensional datasets.

The first distance-based detection technique is introduced by Knorr et al. [49]. According to their proposal, outliers are points from which there are fewer than P other points within distance r (Definition 2.3). In order to detect such outliers, they introduced a nested-loop and a cell-based algorithm. The nested-loop algorithm has time complexity $O(N^2)$ and hence is usually not suitable for applications on large datasets. On the other hand, the cell-based algorithm has time complexity linear with N, but exponential with the number of dimensions dim. In practice, this can only work efficiently when $dim \leq 4$, so it is inapplicable for dealing with high-dimensional datasets.

The second distance-based detection technique is introduced by Ramaswamy et al. [68]. Instead of counting the r-neighborhood of a data point, this technique only takes the data point's distance to its k^{th} nearest neighbor into account (Definition 2.4). As pointed out in [15], this definition of outlier is not intuitive enough since information of other neighbors is simply ignored when computing the outlier score. In [68], three algorithms are proposed: nested-loop with $O(N^2)$ time complexity, index-based and partition-based algorithms. The general idea of index-based algorithm is that: by maintaining a list of top n outliers, we then can prune out data points whose outlier score computed so far is less than the minimum score in the list. Usually this idea can be used in techniques where outlier score computed so far is always upper bound the true score. Some of other techniques that exploit the same idea can be found in [20, 15]. The descriptive algorithm of index-based is illustrated in Algorithm 1. It is noted that in this algorithm, $OutHeap$ is the top n outliers based on the defined outlier score while $Min(OutHeap)$ returns the minimum outlier score of the heap. $NN(p, k)$ contains the set of k nearest neighbors of a data point p. $PointHeap$ is a data structure for maintaining the set of data points utilized in iterations of k nearest neighbors computation. In [68], a spatial index structure like R^*-tree is employed to facilitate such computation. For each data point, $OutScore$ is its outlier score computed so far. The computation process of a data point terminates whenever its $OutScore$ falls below the $Min(OutHeap)$, and hence the time complexity is reduced. Partition-based algorithm proceeds even further in pruning the searching space. The underlying dataset is first grouped into clusters. Each cluster is then assessed whether it contains some candidate outliers, else it will be eliminated. With the remaining clusters, index-based or nest-loop algorithm can be used to detect outliers. Ramaswamy's technique shows better performance in terms of execution time than the technique in [49].

While the outlier definition introduced in [68] only considers the distance from a data point to its k^{th} nearest neighbor as the outlier score, techniques proposed by Angiulli et al. [15, 11] use a much more meaningful metric by taking the total distances from a point to its k nearest neighbors as the outlier score (Definition 2.5). The increase in the number of distances used for computing outlier score does not lead to any increase in time complexity (compared to Ramaswamy's technique) since the number of nearest neighbors that must be found for each data point in each definition is still the same, which is k. Therefore, the notion of outliers used in [15, 11] is better and more intuitive.

Algorithm 1: Final_Processing

```
Set $OutHeap = \varnothing$
Set $Min(OutHeap) = 0$
foreach each data point $p$ in the dataset do
    foreach each data point $q$ in the dataset do
        if $q \neq p$ then
            Update $NN(p, k)$ using $q$
        if $|NN(p, k)| = k$ and $p.OutScore < Min(OutHeap)$ then
            continue outer loop with the next data point
    Update OutHeap using $p$
```

As mentioned before, distance-based techniques usually involve in computing points' nearest neighbors, which is very time-consuming. Therefore, later techniques in distance-based outlier aim to introduce algorithms with less time complexity than the previous ones. Among the methods for reducing the computational cost, pruning outlier searching space and computation reduction are dominant. Computation reduction techniques [68, 37, 20, 15, 11] usually try to limit the number of detected outliers (e.g. top n outliers), and employ similar data structures used in Ramaswamy's index-based algorithm. More specifically, a list of top n outliers found and the minimum outlier score found so far are employed to help reduce the computational cost. Bay et al. [20] provide detailed analysis for this type of algorithm and find out that in average case, the time complexity becomes linear with the dataset's size. In their analysis, any distance-based outlier definition can be used. However, their proposed technique, called ORCA depends on some assumptions such as: (a) the data are in random order and (b) the data points' values are independent. The analysis provided also depends on the cutoff threshold c, which is identical to $Min(OutHeap)$. As can be observed from the Algorithm 1, $Min(OutHeap)$ is usually starts at 0. However, domain knowledge or a training phase can help to achieve a better pruning value. In particular, it is suggested that by training a subset of the original dataset, an initial cutoff threshold can be obtained. The training phase continues if the obtained threshold at the first attempt is not as expected. During the testing phase, the final training set is placed at the top of the dataset so that the cutoff threshold calculated during training phase can be retrieved very soon, and hence the pruning occurs at the very first stage of the detecting process. Domain knowledge can also help in choosing a suitable value for $Min(OutHeap)$.

The linear time complexity presented in [20] can only be obtained if the cut-off threshold c converges to $O(\sqrt{N})$ quickly [37]. However that only happens when the dataset contains many outliers. Motivated by this issue, Ghoting et al. [37] propose an algorithm, called RBRP for detecting outliers that is able to overcome the weakness of ORCA. Its asymptotic time complexity is $O(N \cdot lgN)$. The key idea behind this algorithm is instead of finding the exact nearest neighbors for each data point, the approximate ones are searched for. The approximate nearest neighbors of a data point p is k points within distance c from p. A clustering algorithm is employed (e.g. K-Means clustering) to partition points into bins such that points that are close to each other in space are likely to be assigned to the same bin. Data point p's approximate nearest neighbors are searched in p's bin and consecutive bins. For all normal points, the searching time is linear w.r.t. the dataset's size, i.e. $O(N)$. On the other hand, we need to perform a full scan on the entire dataset for each outlier. That searching strategy leads to reduction in execution time.

Angiulli et al. [15] propose a detection technique using Hilbert space filling curve [71] to map a multi-dimensional space to the interval $I = [0, 1]$ to reduce the computational cost for finding k nearest neighbors. This is done through two steps: map the dataset DS to $D = [0, 1]^{dim}$ where dim is the number of dimensions of DS. Hilbert space filling curve is then used to map D to I. Two data points that are close in I will be close in D but the reverse is not always true. Searching for a data point p's nearest neighbors becomes searching p's approximate nearest neighbors in I by assessing p's predecessors and successors in I. The proposed technique consists of two phases. During the first phase, the approximate outliers (based on approximate outlier score) are extracted from the dataset using the mentioned mapping. The approximate score is always upper bound the true score. In the second phase, true outliers will be extracted from the set of approximate ones. The time complexity of the first phase is reported to be $O(dim^2 \cdot N \cdot k)$ where k is the number of nearest neighbors taken into account. The second phase has time complexity to be $O(N' \cdot N \cdot dim)$ where N' is the number of candidate outliers left after the end of the first phase.

While the aforementioned techniques attempt to reduce execution time of the detection process, Tao et al. [78] aim at reducing I/O cost. The proposed technique, SNIF, scans the dataset two to three times, and reduces I/O cost by keeping a sample set of small size in memory to build a summary of the original dataset. The sample's size is proven to occupy less than 10% of the total dataset's size. This summarization is then used to early prune normal data points. The I/O overhead of SNIF is proven to be $O(N)$. However, as pointed out in [13] the time complexity aspect of SNIF is not optimized. Motivated by that, Angiulli et al. [13] introduce a new detection technique with linear CPU and I/O cost called DOLPHIN. DOLPHIN builds a data structure called DBO-index in memory for scanning the considered dataset. DBO-index exploits *pivot-based index* for executing range query search. Its size is empirically verified to be less than 2% of the dataset's size. Through extensive experimental results, DOLPHIN is shown to yield much better performance than existing distance-based techniques such as ORCA (the most efficient detection technique in terms of temporal cost) and SNIF (the most efficient one in terms of space and I/O cost).

2.4.4 Density-based Techniques

There are two outstanding definitions of density-based outliers which can be found in [25, 67]. Density-based methods (LOF [25], LOCI [67]) in general assign to each data point a factor describing the relative density of that data point's neighborhood. Similar to distance-based approach, density-based detection also involves in the computation of data points' nearest neighbors. However, the measurement of a data point p to its nearest neighbors is then compared to its neighbors' same measurement. The purpose of doing so is to overcome different effects of dense and sparse clusters on points' neighborhood in detecting outliers. However, that comes with a tradeoff in which the computational cost becomes even more expensive than that of distance-based techniques. In spite of that, once again, because of their applicability for large and high-dimensional data, such kind of methods still attract much attention from the research community.

We here reuse a popular example that is usually used to highlight the advantage of density-based approach (c.f., Figure 2.3). It is first proposed in [25]. Assume the distance from every object p_3 in C_1 to its nearest neighbor is greater than the distance from p_2 to its nearest neighbor in C_2. If a distance-based definition like the one proposed by Ng et al. [49] is used, there will be no values of P and r such that p_2 will be an outlier while every object in C_1 is not.

Breunig et al. [25] propose the first density-based detection technique. The outlier score used, called Local Outlier Factor (LOF), is a measure of difference in neighborhood density of a point p and the same measurement of other points in its local neighborhood. Definitions 2.6, 2.7, 2.8, 2.9 and 2.10 describe the concept of LOF. As shown in [55], LOF is able to capture local outliers. For data points that belong to a cluster, their LOFs are approximately equal to 1, while for each outlier the corresponding value should be much higher. Experimental results obtained in [55] demonstrate that LOF outperforms other detection techniques in most cases. All the computations of LOF depend on $MinPts$, which is used for computing the neighborhood density for each data point. The choice of $MinPts$, however, is not simple. According to [25], LOF does not change monotonically as $MinPts$ increases. A method for estimating the range of $MinPts$ is also discussed in the same article.

Definition 2.6 [k-distance of p] *The k-distance of p, denoted as k-distance(p) is defined as the distance $d(p, o)$ between p and o such that*

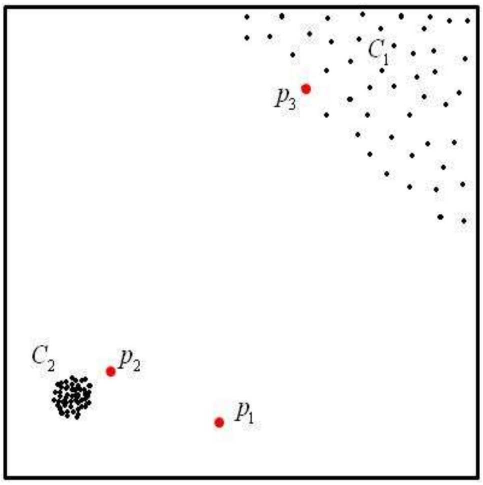


Figure 2.3: Example showing a case where distance-based outlier definitions, e.g. the one in [49], do not work.

- *For at least k data points $o' \in DS \setminus p$ it holds that $d(p, o') \leq d(p, o)$*
- *For at most $(k-1)$ data points $o' \in DS \setminus p$ it holds that $d(p, o') < d(p, o)$*

Definition 2.7 [k-distance of p's neighborhood] *The k-distance of p's neighborhood contains every object whose distance from p is not greater than the k-distance, is denoted as*

$$N_k(p) = \{q \in DS \setminus p | D(p, q) \leq k - distance(p)\}$$

Definition 2.8 [Reachability distance of p w.r.t. o] *The reachability distance of data point p with respect to o is defined as*

$$reach - distk(p, o) = max\{k - distance(o), d(p, o)\}$$

Definition 2.9 [Local reachability density of p] *The local reachability density of a data point p is the inverse of the average reachability distance from the k nearest neighbors of p*

$$Lrd_k(p) = 1/[\frac{\sum\limits_{o \in N_k(p)} reach - distk(p, o)}{|N_k(p)|}]$$

Definition 2.10 [Local outlier factor of p]

$$LOF_k(p) = (\frac{\sum\limits_{o \in N_k(p)} Lrd_k(o)}{Lrd_k(p)})/|N_k(p)|$$

While LOF performs better in terms of accuracy than other techniques, its computational complexity is, however, very expensive since the neighborhood measurement needs to be done not only for each data point p itself but also for other points in p's local neighborhood. Motivated by that, Jin et al. [46] propose a pruning technique to reduce the LOF computation. The main idea of this solution is to compress the data into *micro-clusters* and estimate an upper bound and lower bound on LOF values for each cluster. The clusters are then pruned based on their estimated bounds and the remaining clusters are then used to detect the top n outliers. More specifically, a heap $OutCluster$ of top n clusters with largest lower bounds is maintained. We denote $Min(OutCluster)$ as the minimum lower bound value of all clusters inside the heap. During pruning process, a micro cluster MC will be deleted if its upper bound of LOF is smaller than the current $Min(OutCluster)$. Finally, assume there are N_c micro clusters left. For each data point p in those clusters, its true LOF value is computed. The top n points with largest LOF values will be output as the final result. The pruning technique utilized here is similar to the one proposed in [68].

Papadimitriou et al. [67] introduce new definition of density-based outliers. Instead of using the k nearest neighbors of a data point p in computing its outlier score, the r-neighborhood of p (points that are within distance r from p) is employed. The outlier score of each data point, called MDEF, is used to compare against the normalized deviation of its neighborhood's scores. The 3-σ scheme is employed in the related experiments. That removes the need of using any static cutoff or any score ranking. Besides the new definition of density-based outlier, the paper also introduces one form of outlier abstraction, called LOCI plot. This plot describes how a data point's neighborhood changes as the neighborhood radius changes. It is shown to be meaningful enough in capturing the general view about how a normal data point as well as how an outlier behaves. Although the notion of MDEF does not involve in any k nearest neighbors searching, the computational cost is nevertheless still very expensive. This is an inherent characteristic of density-based detection techniques. To overcome the drawback, an approximate version of the original algorithm is proposed named AMDEF. It aims to approximate all the factors used in the computation of outlier score (as described in the above definitions) by employing the cell-based data partitioning scheme. Using the approximation of outlier score is also another way to reduce the time complexity. This technique again shows how important the partition-based strategy is in outlier detection.

2.4.5 Evolutionary-based Technique

Both of distance-based and density-based techniques involve in the computation of distances from each data point to its neighborhood. However, for high-dimensional dataset the concept of locality as well as neighbors becomes less meaningful [21]. This is because in multi-dimensional space, data distribution becomes very sparse. Therefore, distances between data points tend to cluster around a specific value. In particular, the distance of one data point to its nearest neighbor approaches that to its farthest neighbor. This causes the notion of nearest neighbors in computing the outlier score to become less meaningful. Another problem associated with high-dimensional data is that it is difficult to define the distance function. Full-dimensional distance function may hide the deviation of some specific attribute's value. Aggarwal et al. [7] propose an evolutionary-based technique to overcome the mentioned problem. Basically, the technique performs a grid *discretization* of the data by dividing each attribute's range of values into $\varnothing$ *equidepth* sub-ranges. The combination of sub-ranges from different attributes will form a sub-projection of the dataset. If a sub-projection is formed by d_t sub-ranges of d_t corresponding attributes, it is called a d_t-dimensional sub-projection. The underlying algorithm receives d_t which is the number of dimensions of sub-projections, and m which is the number of sub-projections to return as two parameters. It then mines m sub-projections of dimensionality up to d_t having smallest densities. Points belonging to one of those m returned sub-projections are flagged as outlier.

Suppose the number of dataset dimensions is dim, the total number of d_t-dimensional sub-projections is equal to $C^{d_t}_{dim} \cdot \varnothing^{d_t}$, which is very large. For such a large searching space, a brute force algorithm definitely will be impractical. As an alternative, an evolutionary algorithm is proposed. Each sub-projection $Proj$ is transformed into a string of length dim. At a specific i^{th} attribute, if $Proj$ contains sub-range j^{th} $(1 \leq j \leq \varnothing)$ of that attribute, then the character at position i in $Proj$'s string will receive value j. Otherwise the character will receive value '*'. With the presentation of sub-projections as strings, the crossover and mutation activities can be carried out. The execution time is empirically shown to be slightly worse than linear. The results are analyzed and some meaningful outliers on tested real world datasets, e.g. Breast Cancer [24], are also identified. However, this technique suffers an inherent drawback associated with evolutionary algorithms which is its sensitivity to the selection of initial population size, crossover and mutation probabilities [45]. This issue nonetheless has not yet been addressed.

2.5 Summary of Related Works and Their Limitations

Overall, statistics-based techniques are though simple in principles, inapplicable for data with more than three dimensions. Clustering-based approaches lack formal notions for outliers and consider anomalies only by-products of the clustering process. This limits their capabilities in providing intuition on the outlier-ness of the results.

Distance-based techniques are excellent alternatives for detecting outliers due to their intuitive notions of anomalies as well as their great efficiency on large datasets. Density-based techniques on the other hand steps further in improving the detection accuracy. They nevertheless suffer high execution time than distance-based approaches.

Existing works on distance-based detection using the outlier notion introduced in [15] have so far improved much the issue of high execution time. However, their performance are still contingent on the type of underlying datasets. This motivated us to develop pruning strategies helping to further reduce the temporal cost significantly and consistently on different datasets (c.f., Chapter 4).

All aforementioned approaches but evolutionary-based one do not handle well high-dimensional data due to the curse of dimensionality. In particular, they work with the full set of dimensions and hence, lose information residing in subspaces. Evolutionary-based method though being able to handle such data, has its performance significantly impacted by inappropriate choices of parameters while an effective guidance is unavailable. To better resolve the issue, we design a new technique (c.f., Chapter 3) which is able to combine individual detectors' performance on heterogeneous subspaces to yield highly accurate outcome.

Chapter 3

Ensemble Outlier Detection on Subspaces

Despite the importance of detecting outliers, defining outliers in fact is a nontrivial task which is normally application-dependent. On the other hand, detection techniques are constructed around the chosen definitions. As a consequence, available detection techniques vary significantly in terms of accuracy, performance and issues of the detection problem which they address. In this chapter, we propose a unified framework for combining different outlier detection algorithms. Unlike existing work, our approach combines non-compatible techniques of different types to improve the outlier detection accuracy compared to other ensemble and individual approaches. Through extensive empirical studies, our framework is shown to be very effective in detecting outliers in the real-world context.

3.1 Problem Formulation and Technique Descriptions

There are many ways in practice to define what outliers exactly are, e.g., r-neighborhood Distance-based Outlier [49], k^{th} Nearest Neighbor Distance-based Outlier [68] (a.k.a. k-NN) and Cumulative Neighborhood [11]. Since detection methods are usually constructed around specific outlier notions, their detection qualities vary significantly among datasets. For example, a recent study in [55] shows that the Nearest-Neighbor (NN) method performs well when outliers are located in sparse regions whereas LOF performs well when outliers are located in dense regions of normal data.

Existing techniques usually compute distances (*in full feature space*) of every data sample to its neighborhood to determine whether it is an outlier or not [7, 25, 49, 67]. This causes two side-effects. First, for high-dimensional datasets the concept of locality as well as neighbors becomes less meaningful [21]. Second, not all features are relevant for outlier mining. More specifically, popular distance functions like Euclidean and Mahalanobis are extremely sensitive to noisy features [55]. Despite the presence of the curse of dimensionality, it is difficult in practice to choose a relevant subset of features for the learning purpose [7, 43, 56].

While the nature of data is unpredictable, there is a need for an efficient technique to combine different outlier detection techniques to overcome the drawback of each single method and yield higher detection accuracy. The motivation here is similar to the advent of ensemble classifiers in the machine learning area [43, 52]. With the feasibility of ensemble learning and subspace mining demonstrated, the natural progression would be to combine them both. Lazarevic et al. [56] propose the first solution for *semi-supervised* ensemble outlier detection in feature subspace. That work assumes the existence of outlier scores where a *combine* function can be applied directly. However, this is not practically true since different detection methods can produce outlier scores of different *scales*. For example, it can be recognized that the scores produced using k^{th} Nearest Neighbor Distance-based Outlier [68] are smaller in scale than those using Cumulative Neighborhood [11]. Furthermore, as pointed out in Section 3.2.3, different detection techniques also produce different *types* of score vectors. In particular, some vectors are real-valued while others are binary-valued. This leads to the need of a unified notion of outlier score and an efficient technique to specifically deal with scores' heterogeneity. The availability of such notion would facilitate the task of combination.

Consider a dataset DS with N data samples in dim dimensions. While most of the data samples in DS are normal, some are outliers, and our task is to detect these outliers. While few outliers can be found when all dimensions are taken into account, most of them can only be identified when looking at some subsets of features. In addition, some features of DS are noisy, and cause the full distance computation to be inaccurate if they are included. Given a set of base outlier detection technique(s), our goal is to build an efficient method to combine the results obtained from them while overcoming their individual drawbacks when applying on DS. The ensemble framework should: (a) alleviate of the curse of dimensionality and noisy features, (b) efficiently combine outlier score vectors of base techniques having different *scales* and different *characteristics*, and (c) provide higher detection quality than each individual base technique used in the ensemble (when applied on full feature space).

In order to address this problem, we present the *He*terogeneous *D*etector *E*nsemble on *Random S*ubspaces (HeDES) framework. The advantage of using HeDES lies in its ability to incorporate various heuristics for combining different types of score vectors. The main contributions of this chapter can be summarized as follows:

- We introduce a unified notion of outlier score function and show how existing outlier definitions can be represented using it. We demonstrate how to identify different types of outlier scores in literature by using this new notion of outlier score function.
- We propose a generalized framework for ensemble outlier detection in feature subspaces - HeDES. Unlike the existing simple framework in [56], HeDES is able to combine different techniques producing outlier scores of different scales or even different types of scores (e.g., real-valued v/s. binary-valued).
- We demonstrate through extensive empirical studies that the HeDES framework can outperform state-of-the-art detection techniques and is therefore suitable for outlier detection in real-world applications.

The rest of this chapter is organized as follows. Details of our approach are provided in Section 3.2 and empirical comparison with other current-best approaches is discussed in Section 3.3. Finally, the chapter is summarized in Section 3.4 with directions for future work.

3.2 Methodology

The HeDES framework is a *generalized* framework for mining outliers in subspaces using ensemble of outlier detection techniques (henceforth termed detectors). In the following, we present the details of constructing the ensemble and explain how it is applied in HeDES.

3.2.1 Ensemble Construction

The process of constructing the ensemble of detectors is displayed in Algorithm 2. In each of the total R rounds, we first sample a detector T from the pool of techniques considered ($\mathcal{T}$) on a *round-robin* basis. Practically, R should be chosen as a multiple of the pool size. Next, we form a subspace S where T will operate by randomly choosing N_f features from the full feature space. Here, N_f is sampled from the uniformly distributed range $[\lfloor dim/2 \rfloor, dim-1]$.

The pair (T, S) is then added to the ensemble. By sampling N_f from the range $[\lfloor dim/2 \rfloor, dim-1]$ instead of fixing it to $\lfloor dim/2 \rfloor$ like in [43], we increase the possibility of generating different subsets of features for each detector in the ensemble. Since the detection capability of each detector relies on its own notion of dissimilarity measure, this increases the chance that they generalize their prediction in ways different to each other. Hence, the above process of constructing the ensemble takes advantage of high-dimensional feature space and weakens the curse of dimensionality.

After identifying all the detectors to be used in the ensemble, we adjust their weights by running the ensemble against an *unlabeled* training set. The intuition behind this weight-adjust is that some detection techniques are more powerful than others on some certain types of data. For example, recent study by Lazarevic et al. [55] shows that the Nearest-Neighbor (NN) method outperforms LOF when outliers are located in sparse regions whereas LOF yields higher performance than NN when outliers are located in dense regions of normal data. Even though the detectors in the ensemble are applied on the same dataset during testing, the subspaces where they operate are homogeneous. Furthermore, subspace distributions are different whereas detectors' prediction performance is dependent on their respective subspace. Thus, our argument on detectors' superiority over the others in some certain data still holds in our ensemble learning. Since the nature of subspaces is unpredictable, assigning fixed weights for detectors is not a good solution. Intuitively, had we known which detectors would work better, we would give higher weights to them. In the absence of this knowledge, a possible strategy is to use the result of detectors on a separate validation dataset, or even their performance on the training dataset, as an estimate of their future performance.

Algorithm 2: Constructing HeDES

```
for $i = 1$ to $R$ do
    Choose a detector $T_i \in \mathcal{T}$
    Randomly sample $N_f$ from $[\lfloor dim/2 \rfloor, dim - 1]$
    Randomly sample a subset of features $S_i$ of size $N_f$ from the feature set of $DS$
    Add $(T_i, S_i)$ into the ensemble
Apply the ensemble to the synthetic training dataset
Adjust the weight of each detector in the ensemble
```

Our approach, similar to AdaBoost [35], employs the latter strategy. However, since the training set is unlabeled, a direct weight-adjust is not straightforward. To overcome this problem, we construct a labeled synthetic training dataset from the original (unlabeled) one by applying the technique presented in [4]. In brief, the synthetic set is comprised of normal data drawn from the original one and artificially generated outliers. The artificial outliers here are created by using a uniform distribution U that is defined within a bounded subspace whose minimum and maximum are limited to be 10% beyond the observed minimum and maximum, respectively. Let the original training set be S_{tr}, we construct the set of artificial outliers S_{out} of size $|S_{tr}|$ according to U on the bounded domain. The synthetic training set is then set to be $S_{tr} \cup S_{out}$. More details are given in [4]. The use of this set helps us estimate the performance of each detector in the ensemble and adjust its weight correspondingly despite the lack of knowledge on anomalous behavior. Since outlier detectors in the ensemble are unsupervised, they

Algorithm 3: Mining Outliers with HeDES

```
Normalize $DS$
foreach detector type $j$ do
    $TVS_j = \varnothing$
for $i = 1$ to $R$ do
    Choose the detector $(T_i, S_i)$ from the ensemble
    $j$ = type of $T_i$
    $RVS_i$ = apply $T_i$ to $DS$ projected on $S_i$
    $TVS_j = TVS_j \cup \{RVS_i\}$
foreach detector type $j$ do
    $VS_j = SUBCOMBINE(TVS_j)$
$VS_{FINAL} = COMBINE(VS_1, VS_2, \ldots)$
```

are less susceptible to the overfitting problem. In other words, the weights trained are loosely coupled with the synthetic training set. Furthermore, this artificial data generation has been shown in [4] to be successful in training highly accurate classifiers. Thus, the weights obtained in the training phase are likely to have very high generalization capability on unseen test data. By using the weight-adjusted scheme, the effect of detection techniques that are not as relevant as the others can be reduced. This becomes even more critical when irrelevant techniques may lead to a significantly wrong assignment of outlier score (c.f., Section 3.3).

3.2.2 HeDES Framework

Our proposed approach, HeDES, is described in Algorithm 3, and functions as follows. The testing dataset is passed through the ensemble. For every pair (T, S) in the ensemble, we apply T to DS projected on subspace S and obtain a raw vector score. This raw vector score is stored together with other vector scores generated by the same detector type j in TVS_j. After finishing R rounds, each set of vector scores (vectors in the same set are of the same type) are combined separately using SUBCOMBINE function to yield a vector score VS_j. Finally, the COMBINE function is invoked using all the VS's obtained to produce the final vector score VS_{FINAL}. The interpretation (combination) of VS and VS_{FINAL} depends on the specific combine functions utilized which are explored in detail in Section 3.2.4. Note that the two most important components in this framework are: (a) the outlier score function, and (b) the (SUB)COMBINE functions. The main difference between the simple subspace ensemble framework in [56] and our generalized framework lies in the multi-staged combine function which allows much more flexible integration among the heterogeneous types of outlier detection techniques. It is highlighted that similar to other ensemble classifiers [35, 43], ensemble outlier detection method is a *parallel learning* algorithm [56]. Since each round of running is independent of the other, a parallel implementation can be employed for faster learning.

3.2.3 Outlier Score Function

Assume a metric distance function D exists on DS, using which we can measure the dissimilarity between two arbitrary data samples in any arbitrary subspace. A general approach that has been used by most of the existing outlier detection methods is to assign an *outlier score* (based on the distance function) to each individual data point, and then design the detection process based on this score [7, 25, 49]. The use of the outlier score is analogous to the mapping of multi-dimensional datasets to $\mathbb{R}$ space (the set of real numbers). In other words, we can define the outlier score function (F_{out}) which maps each data sample in DS to a unique value in $\mathbb{R}$. Intuitively, to create an outlier score function, we first identify a set of measurements based on some specified criteria, then define a mechanism g for combining them, and finally generate a function (F_{out}) based on g. Most the existing techniques utilize only a single measurement, i.e., g becomes a *uni-variable* function that is related directly to the only measurement taken into account. With reference to the k-NN [68], let the measurement considered be the distance from a data pattern p to its k^{th} nearest neighbor (D^k), then a possible choice of F_{out} is $F_{out} = g(D^k) = D^k$.

3.2.3.1 Outlier score function classification

Among existing approaches to outlier detection problem, we can classify F_{out} into *global* and *local* score functions [63]. An outlier score function is called *global* when the value it assigns to a data sample $p \in DS$ can be used to compare globally with other data samples. More specifically, for two arbitrary data samples p_1 and p_2 in DS, $F_{out}(p_1)$ and $F_{out}(p_2)$ can be compared with each other, and if $F_{out}(p_1) > F_{out}(p_2)$, p_1 has a larger possibility than p_2 to be an outlier. The definitions proposed in [11, 25, 68] straightforwardly adhere to this category. On the other hand, the definition in [49] can be converted to this category by taking the inverse of the number of neighbors within distance r of each data point. In contrast, a *local* outlier score function assigns to each data sample p, a score that can only be used to compare within some local neighborhood. Example of such a function is proposed in [67], where the local comparison space is the set of data samples lying within the *circle* centered by p and the *radius* is user-defined. The choice of a global or local outlier score function clearly affects later stages of the algorithm design process.

3.2.3.2 A classification of detection techniques using F_{out}

Using the notion of F_{out} defined above, existing outlier detection techniques can be classified into two types: (a) Threshold-based where a local F_{out} is usually used, and (b) Ranking-based where a global F_{out} is employed, (c.f., Definitions 3.1 and 3.2, respectively). According to this classification, the methods proposed in [7, 11, 68] using global score functions are classified as Ranking-based. On the other hand, LOCI with local score function is classified as Threshold-based. Although the technique in [49] utilizes a global F_{out}, it is classified as Threshold-based by letting $F_{out}(p) = \frac{1}{|S(p)|}$ and choosing $t = \frac{1}{1-P}$. In this case, a data sample $p \in DS$ is an outlier if $F_{out}(p) > t$, i.e., $F_{out}(p) > \frac{1}{1-P}$. Note that the threshold t in LOCI is dynamic, whereas that in [49] is static (dependant on the pre-defined variable P).

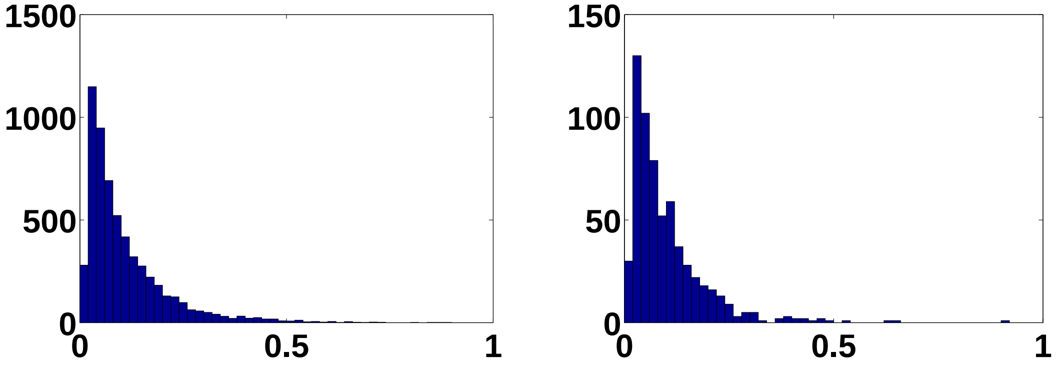
Definition 3.1 [Threshold-based] *Given a (dynamic or static) threshold t, a data sample p is an outlier of DS if $F_{out}(p) > t$.*

Definition 3.2 [Ranking-based or Top-n-outlier] *Given a positive integer n, a data sample p is an n^{th} outlier of DS if no more than $n-1$ other points in DS have a higher value of F_{out} than p. An algorithm based on this definition outputs the top n outliers.*

When F_{out} is global, a Ranking-based technique is normally preferred since the assigned score values of data samples can be compared globally to produce the top points with largest scores. The resultant score vector is then real-valued and identical to the values that F_{out} assigns to data samples. On the other hand, if F_{out} is a local one, a Threshold-based approach becomes a reasonable choice. As a consequence, the score vector obtained contains only binary values (0 for non-outliers and 1 for outliers) since the scores produced by F_{out} are already *discretized* through a threshold-based test. Therefore, score vectors produced by different detection techniques are heterogeneous and need to be processed carefully to facilitate the COMBINE process.

3.2.3.3 Issue of converting F_{out} to the posterior probabilities

Assume by applying an outlier detector T with outlier score function F_{out} onto DS, we obtain the score vector: $RVS = \{F_{out}(p_1), F_{out}(p_2), \ldots, F_{out}(p_N)\}$. The problem of outlier detection is equivalent to a binary classification problem with two classes: O (outlier class) and M (normal class). One important question which has not been addressed well by the research community is how to compute the posterior probability $P(O|F_{out}(p_i))$ using the knowledge on RVS. Gao et al. [36] propose two methods attempting to solve this problem. The first method bases on the assumption that the posterior probabilities follow a logistic sigmoid function and the



3.1.a: Score distribution for normal samples

3.1.b: Score distribution for outliers

Figure 3.1: Outlier score distribution of Satimage dataset using LOF as the detector.

normal and anomalous samples have similar forms of outlier score distribution (same covariance matrix). It then tries to learn the function's parameters using RVS. The second learner on the other hand models the likelihood probability distributions $P(F_{out}(p_i)|O)$ and $P(F_{out}(p_i)|M)$ as a Gaussian and an exponential distribution, respectively. The posterior probabilities are then computed using Bayes theorem. Among the two methods, mixture modeling is more suitable for ensemble learning as demonstrated in [36].

The main intuition leading to this mixture model is derived from the empirical studies using k-NN as the score function. However, the argument used in [36] does not hold for density-based approaches, such as LOF, where density of a data sample is compared (divided) to that of its neighbors. Because of limited space, we omit the demonstration here. Our empirical studies (c.f., Section 3.3) point out that processing the outlier scores directly (like in HeDES and Feature Bagging) instead of converting to posterior probabilities will yield better detection results.

We verify our claim by running experiments on the Satimage dataset (c.f., Section 3.3) using LOF as the outlier detector and compute the likelihood probability distribution accordingly. The results are shown in Figure 3.1. Here the outlier scores are normalized using the min-max normalization, and the interval $[0, 1]$ is divided into 50 equi-width bins. As can be seen, the outlier score distribution of outlier class does not adhere to the assumed Gaussian distribution at all. Apart from the Satimage datasets, we also tested on several other datasets and arrived at the same conclusion. In fact, our empirical studies (c.f., Section 3.3) point out that processing the outlier scores directly (like in HeDES and Feature Bagging) instead of converting to posterior probabilities will yield better detection results.

3.2.4 COMBINE Functions

Lazarevic et al. [56] introduce two combine functions (Cumulative Sum and Breadth First) which have been successfully used in ensemble-based outlier mining. Here, we present three novel combine functions which are *Weighted Sum*, *Weighted Majority Voting* and *OR Voting*. Unlike Breadth First, these functions are invariant to the order of the detectors. Since accuracy is the most critical factor in ensemble learning, this property becomes an advantage of our approach. Among them, the first two functions are shown to be very efficient in ensemble classification and have been widely employed in many practical applications [75, 35]. The intuition for utilizing weighted combine functions were also discussed in details above. Weighted Majority Voting is known to excel in combining class labels assigned by different classifiers in the ensemble. On the other hand, Weighted Sum in classification is normally applied on posterior probabilities [43]. Conversely, in HeDES, it is used to combine *normalized* outlier scores produced by different detectors of the ensemble. Finally, Or Voting is a natural combine function for integrating heterogeneous types of output scores as demonstrated later. It is important to note here that exploring all possible combine functions is not a focus in this work. Nevertheless, our chosen combine functions are still able to encompass almost all available types of outlier scores in the field.

Although HeDES provides an easy extension to score vectors of various types (depending on the purpose of learners), in our approach score vectors are either real-valued or binary-valued. An natural approach (Ensemble Voting) to combine different types of score vectors is to simply normalize and *discretize* the real-valued score vectors (convert all score vectors to the same type), and thereafter integrate all the binary-valued score vectors (inclusive of the discretized real-valued score vectors) using Weighted Majority Voting. However, such a natural approach is not sufficient and does not produce good results (c.f., Section 3.3). The set of input score vectors to the (SUB)COMBINE function is classified into two groups in which the first group contains score vectors (TVS_R) resulting from applying Ranking-based techniques, whereas the second group contains score vectors (TVS_T) of Threshold-based ones. Our strategy is to apply some combine function on TVS_R and TVS_T separately to obtain VS_R and VS_T. Finally, a special combine function is used to integrate VS_R and VS_T to produce the final score vector VS_{FINAL}. It is noted that the problem of combining results of Ranking-based and Threshold-based techniques here is very similar to the problem of combining detection results of categorical and continuous features in mixed-attribute datasets as addressed in [66]. In both cases, we process real values and binary/categorical values separately. Eventually, a heuristic is used to integrate the results obtained. This is the base intuition for our Or Voting combine function.

3.2.4.1 Processing outlier score vectors

Because of the different nature between Ranking-based and Threshold-based techniques, outlier score vectors produced by them need different treatments. Assume the data samples in DS are $p_1, p_2, \ldots, p_N$. A detection technique T using a specific score function F_{out} is applied to identify outliers in DS. We denote T's resultant score vector as $RVS = \{F_{out}(p_1), F_{out}(p_2), \ldots, F_{out}(p_N)\}$.

If T is a Ranking-based technique: Vectors of different Ranking-based techniques may have different scales [68]. Hence, to apply combine functions, real-valued vectors need to have equivalent scale. In other words, normalization is necessary. In HeDES, RVS is normalized using the standardization technique. One of the most important characteristics of this normalization technique is its ability to maintain the detectability of extreme values after performing normalization [40]. As argued in [56], this facilitates combining real-valued vectors since a data sample receiving a high score value by one detector, after summing up its score with those produced by other detectors, may still have large values and be flagged as outliers. We define the *normalized* value of $F_{out}(p_j)$ in RVS as: $Score_{norm}(p_j) = \frac{F_{out}(p_j) - m}{s}$ where $m = \frac{1}{N}(\sum_{i=1}^{N} F_{out}(p_i))$ and $s = \frac{1}{N}(\sum_{i=1}^{N} |F_{out}(p_i) - m|)$. By applying normalization, the range of outlier score becomes independent of the technique used. Since all normalized vectors score have comparable scale, it is feasible to integrate them.

If T is a Threshold-based technique: We preserve RVS as it is. This is because each individual element in RVS already indicates the posterior probability of being outlier for data points. Thus, if an ensemble employs techniques from both Ranking-based and Threshold-based, we need a special combine function. Since *Cumulative Sum* and *Breadth First* functions ignore the score vectors' heterogeneity, they are not suitable for use.

3.2.4.2 Weighted Sum

This function is used for vectors in TVS_R. Let us denote the *weight* of the detector $T_i \in \mathcal{T}$ at round i with score vector RVS_i as W_i. The final score vector of all vectors in TVS_R is defined as: $VS_R = \sum_i W_i \times RVS_i$. Weighted Sum is in fact a modified version of Cumulative Sum proposed in [56]. However, the weight-based strategy helps boost the performance of more efficient detectors. This cannot be obtained in equi-weight schemes.

3.2.4.3 Weighted Majority Voting

This combine function is used for processing vectors in TVS_T. Although similar to most of the existing ensemble classifiers [43, 35], the problem here is much simpler since we are only interested in two classes of data: normal (class M) and outlier (class O). Since all vectors in TVS_T only contain binary values, they are suitable for Weighted Majority Voting. As in the case of Weighted Sum, the weight of each vector is determined by the performance of the corresponding detection technique on training datasets.

3.2.4.4 OR Voting

This function is used for combining VS_R and VS_T. However, its input vectors must contain only binary values. Therefore, we perform a *discretization* process on VS_R where its top values are converted to 1, and the rest are converted to 0. Under this scheme, we have: $VS_{FINAL} = VS_R \vee VS_T$ where "$\vee$" is the usual Boolean operator.

3.2.4.5 Interpretation of VS_{FINAL}

If the pool of detection techniques $\mathcal{T}$ contains only Ranking-based techniques, we then flag those data samples having highest scores in VS_{FINAL} as outliers. In case $\mathcal{T}$ contains only Threshold-based techniques, outliers are those points having score in VS_{FINAL} equal to 1. Finally, if $\mathcal{T}$ contains both Ranking-based and Threshold-based methods, outliers are those whose scores equal to 1 in VS_{FINAL}. Thus, the flagging mechanism for "mixed" $\mathcal{T}$ is similar to that of an ensemble containing only Threshold-based methods. This is because by applying the OR function, the real-valued vector VS_R is already converted to a binary-valued one. Similar to [56], the number of outliers to flag for Ranking-based methods depends on the specific dataset used.

3.2.5 Further Discussion

In HeDES, the feature subspaces are chosen randomly which may affect the quality and utility of subspaces formed. We were aware of this issue and did investigate it during our study. However, it is noted that there is no systematic way to rank the relevance of subspaces for outlier detection. That is because outliers exhibit non-monotonicty property. More specifically, if a data point p does not show any anomalous behavior in some subspace S, it may still be an outlier in some lower-dimensional projection(s) of S (and this is also the reason why we find outliers in subspaces). On the other hand, if p is a normal data point in all projections of S, it can still be an outlier in S. This property prevents the selection of subspaces where outliers show their anomalous behavior and leads to the unavoidable exploration of all subspaces to mine full result set (which is very expensive). The issue was also mentioned in [7, 8, 56].

Dataset	**Number of classes**	**Number of attributes**	**Number of instances**	**Outlier class v/s. Normal**
Ann-thyroid	3	21	3428	class 1, 2 v/s. 3
Lymphography	4	18	148	merged class 2 & 4 v/s. rest
Satimage	7	36	6435	smallest class v/s. rest
Shuttle	7	9	14500	class 2, 3, 5, 6, 7 v/s. 1
KDD Cup 1999	2	42	60839	class U2R v/s. normal
Breast Cancer	2	32	569	class 2 v/s. 1
Segment	7	19	2310	each class v/s. rest
Letter	26	16	6238	each class v/s. rest

Table 3.1: Characteristics of datasets used for measuring accuracy of techniques.

Since there is no efficient method to tell us exactly in which subspaces outliers are present while the number of possible subspaces is exponential to the total number of dimensions, researchers propose two types of approaches to tackle the issue: (a) utilizing combinations of randomly chosen subspaces [56], and (b) mine all subspaces of dimensionality up to some threshold [7, 8]. The former has its origin in ensemble classification [43] where its effectiveness was demonstrated. It was then employed in [56] where the results obtained were also very promising. We also have one paper currently under review exploring the latter approach. Nevertheless, to the best of our knowledge, there is not existent any more informed approach for selecting subspaces. Thus, subspace sampling as utilized in HeDES is a reasonable choice.

3.3 Experimental Results

To verify the effectiveness of the proposed combination framework, we conducted the experiments on several real datasets which are taken from UCI Machine Repository [1]. These datasets are used widely in outlier detection as well as in rare class mining [56], and are summarized in Table 3.1.

The setup procedure (converting datasets into binary-class sets, etc.) employed here follows exactly that of Feature Bagging. In the field of outlier detection, ROC curve (as well as AUC) is an important metric used to evaluate detection quality. Similar to [36, 55, 56, 11], AUC (area under the ROC curve) was chosen as performance benchmark in this work because of its proved relevance for outlier detection [56, 55].

In each experiment, we report how AUC changes when the number of rounds R is varied for KDD Cup 1999 dataset. This dataset is chosen as the representative since it has the largest number of instances as well as attributes among all the datasets considered, and hence is a good representative. For other sets, the results are similar and average AUC with $R = 10$ is

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html

presented (setting R to 10 was suggested in [56, 36]). For every dataset, each reported result is a 95% confidence interval of the AUC obtained by averaging the outcomes of running the algorithms 10 times on each of its generated binary-class sets.

In our empirical studies, two different base detectors are considered: LOF [25] and LOCI [67], and are tested using full feature space. The former is known to be one of the best Ranking-based techniques [55] while the latter is a well-known Threshold-based technique [67]. By choosing these high quality base detectors, we are able to highlight the improvement of HeDES in detection accuracy. For LOF, the parameter $MinPts$ was set to 20. For LOCI, we chose $n_{min} = 20$, $n_{max} = 50$, $\alpha = 1/2$, and $k_\alpha = 3$. Those values were derived from the corresponding works. Apart from the two base techniques, we compared our approach with other ensemble approaches including: Feature Bagging [56], Active Outlier [4], Mixture Model [36], and Ensemble Voting (c.f., Section 3.2.4). Feature Bagging uses two combine functions: Cumulative Sum and Breadth First. For each dataset under consideration, we choose to display the highest AUC value among the two for Feature Bagging. Active Outlier constructs an ensemble after t rounds of training, i.e. the ensemble contains t detectors. Here, t was set to R for fair comparison. Since Active Outlier does not use any base detector, its performance remains the same regardless of which base detector is chosen for other ensemble techniques.

3.3.1 Experiment on Ranking-based Technique

This experiment aims to investigate the performance of the our proposed combine function, Weighted Sum, when applied to the Ranking-based technique. We compared our method against LOF, Feature Bagging (FB), Mixture Model (MM), and Active Outlier (AO). The results are shown in Figure 3.2 and Table 3.2. It can be observed that Weighted Sum strategy yields very good results in all test cases. Even in the case where the base technique, LOF, performs no better than random guessing due to high dimensionality of the dataset (Satimage), our approach is still able to bring very good improvement. The results also indicate that using full feature space in outlier detection may yield low accuracy, especially when the number of features is large and it is likely that some features are noisy. The performance of Mixture Model over the datasets used is worse than Active Outlier and Feature Bagging. This agrees with our argument about the applicability of Mixture Model on other notions of outliers. In particular, the outlier score proposed in LOF is density-based whereas k-NN is distance-based. Extensive studies in the field have pointed out the significant differences between these two notions. These in addition to the results obtained show that the assumption made in Mixture Model is not flexible enough to encompass the scores produced by LOF. For all ensemble techniques considered (including our approach), AUC value increases as the number of detectors included in the ensemble increases. However, Weighted Sum and Feature Bagging tend to work better than AO. This can be attributed to the fact that ensemble learning by subspace sampling produces more efficient learners than data sub-sampling one [43].

3.3.2 Experiment on Threshold-based Technique

In this experiment, we study the effect of our proposed combine function, Weighted Majority Voting (WMV), for Threshold-based techniques. Thus, LOCI is selected as the base detector. Our approach's performance is assessed against LOCI, Feature Bagging (FB, also utilizes LOCI), and Active Outlier (AO). Mixture Model is omitted here since the posterior probabilities can be derived directly from the binary-valued scores. In fact, the results achieved by Mixture Model under this setting are the same as that of Feature Bagging. From Figure 3.2 and Table 3.3, it can be seen that Weighted Majority Voting yields the best or nearly best results in all cases (the margin with respect to the best one is negligible). For Feature Bagging, neither Cumulative Sum nor Breadth First works well in combining vectors of Threshold-based techniques. This indicates that specialized schemes are required. With the results achieved in this test, Weighted Majority Voting is shown to be a promising candidate.

Dataset	**LOF**	**FB**	**MM**	**AO**	**WS**
Ann-thyroid (class 1 v/s. 3)	0.869	0.869 ± 0.015	0.855 ± 0.021	0.856 ± 0.023	**0.892 ± 0.005**
Ann-thyroid (class 2 v/s. 3)	0.761	0.769 ± 0.003	0.759 ± 0.007	0.753 ± 0.009	**0.798 ± 0.008**
Lymphography	0.924	0.967 ± 0.009	0.921 ± 0.001	0.843 ± 0.041	**0.984 ± 0.004**
Satimage	0.510	0.558 ± 0.031	0.562 ± 0.025	0.646 ± 0.024	**0.703 ± 0.022**
Shuttle	0.825	0.839 ± 0.004	0.724 ± 0.017	0.843 ± 0.006	**0.861 ± 0.002**
Breast Cancer	0.805	0.825 ± 0.022	0.758 ± 0.012	0.822 ± 0.015	**0.866 ± 0.017**
Segment	0.820	0.847 ± 0.017	0.798 ± 0.005	0.836 ± 0.002	**0.882 ± 0.003**
Letter	0.816	0.821 ± 0.003	0.722 ± 0.014	0.824 ± 0.002	**0.848 ± 0.001**

Table 3.2: Ranking-based technique: AUC values of LOF, Feature Bagging, Mixture Model, Active Outlier, and Weighted Sum with $R = 10$.

Overall, we can observe that ensemble outlier detection (Feature Bagging, Weighted Majority Voting, Active Outlier) results in good improvements over the base technique. We again observe the same pattern as in the previous experiment: the accuracy of ensemble techniques grows as the number of detectors increases and that of Active Outlier is dominated by our approach's and Feature Bagging's.

Dataset	**LOCI**	**FB**	**AO**	**WMV**
Ann-thyroid (class 1 v/s. 3)	0.871	**0.873 ± 0.003**	0.856 ± 0.023	0.872 ± 0.021
Ann-thyroid (class 2 v/s. 3)	0.747	0.754 ± 0.026	0.753 ± 0.009	**0.812 ± 0.015**
Lymphography	0.892	0.932 ± 0.007	0.843 ± 0.041	**0.987 ± 0.003**
Satimage	0.529	0.535 ± 0.022	0.646 ± 0.024	**0.654 ± 0.024**
Shuttle	0.822	0.856 ± 0.011	0.843 ± 0.006	**0.873 ± 0.004**
Breast Cancer	0.801	0.827 ± 0.002	0.822 ± 0.015	**0.842 ± 0.001**
Segment	0.835	**0.852 ± 0.002**	0.836 ± 0.002	0.850 ± 0.014
Letter	0.811	0.834 ± 0.016	0.824 ± 0.002	**0.872 ± 0.004**

Table 3.3: Threshold-based technique: *AUC* values of LOCI, Feature Bagging, Active Outlier, and Weighted Majority Voting with $R = 10$.

3.3.3 Experiment on Ranking-based & Threshold-based Techniques

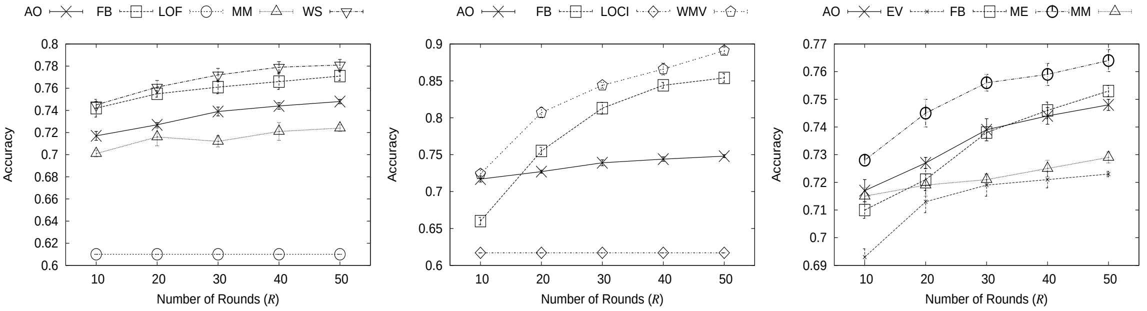
So far in our empirical studies, the ensemble contains either only Ranking-based (LOF) or only Threshold-based (LOCI) detection techniques. We now investigate our last proposed combine strategy, the OR Voting, in an ensemble where both types of techniques are considered. Therefore, in this experiment, both LOF (Ranking-based) and LOCI (Threshold-based) are employed. We call our method under this setting Mixed Ensemble (ME). More specifically, we use Weighted Sum for Ranking-based technique whereas with Threshold-based technique, we apply Weighted Majority Voting. The results from each group are combined using the OR Voting. Our proposed approach is compared against Feature Bagging (FB), Mixture Model (MM), Active Outlier (AO) and the natural combination approach (Ensemble Voting, a.k.a. EV). Ensemble Voting, similar to ensemble classifier using weighted majority voting (e.g., AdaBoost), is shown to yield very high accuracy in the classification problem [35]. However, through this experiment we point out that it is not very applicable for ensemble outlier detection. For Cumulative Sum of Feature Bagging, we simply sum up all score vectors after performing normalization. The AUC values of all methods are presented in Figure 3.2 and Table 3.4. Our approach (Mixed Ensemble) once again performs very well compared to other techniques. The results also show that when an ensemble contains both Ranking-based and Threshold-based techniques, natural sum-up scheme of Cumulative Sum as well as usual ensemble learning based on Weighted Majority Voting does not help much. Instead, we need special combine functions to deal specifically with different types of score vectors.

3.4 Summary

In this chapter, the problem of ensemble outlier detection in high-dimensional datasets were studied in detail. A formal notion of outlier score which helps to identify different types of

Dataset	**FB**	**MM**	**AO**	**EV**	**ME**
Ann-thyroid 1	0.870 ± 0.015	0.813 ± 0.013	0.856 ± 0.023	0.832 ± 0.012	**0.883 ± 0.020**
Ann-thyroid 2	0.768 ± 0.031	0.684 ± 0.001	0.753 ± 0.009	0.754 ± 0.012	**0.792 ± 0.004**
Lymphography	**0.955 ± 0.033**	0.735 ± 0.002	0.843 ± 0.041	0.901 ± 0.235	0.952 ± 0.014
Satimage	0.531 ± 0.003	0.517 ± 0.043	0.646 ± 0.024	0.544 ± 0.007	**0.780 ± 0.005**
Shuttle	0.853 ± 0.028	0.729 ± 0.013	0.843 ± 0.006	0.827 ± 0.024	**0.871 ± 0.016**
Breast Cancer	0.824 ± 0.013	0.755 ± 0.023	0.822 ± 0.015	0.837 ± 0.017	**0.864 ± 0.015**
Segment	0.845 ± 0.007	0.792 ± 0.016	0.836 ± 0.002	0.840 ± 0.004	**0.852 ± 0.006**
Letter	0.841 ± 0.004	0.785 ± 0.011	0.824 ± 0.002	0.836 ± 0.003	**0.877 ± 0.018**

Table 3.4: Ranking-based & Threshold-based techniques: *AUC* values of Feature Bagging, Mixture Model, Active Outlier, Ensemble Voting, and Mixed Ensemble with $R = 10$.



3.2.a: Ranking-based

3.2.b: Threshold-based

3.2.c: Ranking-based & Threshold-based

Figure 3.2: *AUC* values of all competing approaches on the KDD Cup 1999 dataset.

outlier score vectors was introduced. Using the new notion, we presented a heterogeneous detector ensemble on random subspaces (HeDES) framework using different relevant combine functions to tackle the problem of heterogeneity of techniques. Extensive empirical studies on several popular real-life datasets show that our approach can outperform contemporary techniques in the field in terms of detection accuracy.

Chapter 4

The MIRO approach

In this chapter, we address the problem of efficiently detecting distance-based outliers in large datasets. Distance-based outlier detection in general requires substantial amount of computational costs (both I/O and CPU) for mining anomalies. As a preliminary study, we aim at designing a detection method yielding lower CPU overhead compared to other outstanding techniques in the literature.

We employ the outlier function proposed in [15], although the ideas in MIRO can also be adapted to other functions. The intuition and quality of detection results of the chosen outlier definition are based on solid work in [15, 20]. This definition is also used in other popular techniques on outlier detection, e.g. the one in [37]. Therefore, in this paper we do not again demonstrate how well MIRO does in terms of actually discovering abnormalities in real data. Instead, we focus on showing its superiority in terms of CPU cost.

Let us denote the set of k nearest neighbors of a data point p in DS as kNN_p. We can now define the outlier score function F_{out} as follows.

Definition 4.1 [Outlier Score Function] *The dissimilarity of a point p with respect to its k nearest neighbors is known by its cumulative neighborhood distance. This is defined as the total distance from p to its k nearest neighbors in DS. In other words, we have:* $F_{out}(p) = \sum_{m \in kNN_p} D(p, m)$.

This definition has been proved by Angiulli et al. [15] to be more intuitive than the definition used in [68]. Given two positive integers k and n, our task is to mine top n outliers that have the largest outlier scores based on the chosen F_{out}. For ease of reference, symbols used in the chapter are presented in Table 4.1.

Symbol	Definition
DS	The dataset
N	Number of points in the dataset
dim	Dimensionality of the data space
$D(p_1, p_2)$	Distance function between points p_1 and p_2
kNN_p	set of k nearest neighbors of a data point p
n	Number of outliers to be mined
F_{out}	Outlier Score Function

Table 4.1: Definitions of symbols

4.1 Problem Formulation and Technique Descriptions

Distance-based outliers which have been popularly defined as:

- Data points from which there are fewer than p points that are within distance r [51].
- Top n data points whose distance to their corresponding k^{th} nearest neighbor are largest [68].
- Top n data points whose total distance to their corresponding k nearest neighbors are largest [15, 11].

As these definitions indicate, a significant amount of distance computations need to be performed in order to verify whether a data point is an outlier or not. This leads to high execution times and has motivated many attempts to produce efficient algorithms to mine outliers. Among them, outstanding work by Bay et al. [20] and Ghoting et al. [37] aim to reduce execution time by utilizing a simple pruning nested-loop algorithm.

Reducing CPU cost of detection techniques in general generates many benefits for various applications where the speed of detecting deviations plays a critical role (e.g. fraud detection, intrusion detection). One of such application is outlier detection in streaming environment [65]. In such scenarios, storing data into disks and doing post-processing is infeasible. Instead, data objects are received at a fast rate and an algorithm with strict time bound is required to continuously monitor and catch abnormal ones while ensuring no jamming for the subsequent data traffic. As a consequence, CPU cost of processing becomes the main issue. To further illustrate our point, let us consider a system in which data arrives in batch and each batch of data is stored in a buffer memory. It is assumed that the buffer size is large enough to accommodate each batch but if so many batches are stored at the same time, buffer will overflow. Such scenario is very popular in applications dealing with data streams [39]. The task of the system is to identify abnormal records in each arrival batch. The buffer will be automatically flushed when this monitoring process is done. However, if the detection techniques takes so much time relatively to the speed of arrival of batches, we may lose data because of the problem of buffer overflows. Therefore, developing a fast detection algorithm becomes a necessary needs since it leads to higher throughput for the system. Additionally, the higher throughput will also yield higher detection accuracy since data loss is avoided.

Our MIRO approach operates in two phases. In the first phase, we partition the data into clusters, and make an early estimate on the lower bound of outlier scores. This phase prunes clusters that cannot have outliers, and the second phase then processes the remaining clusters using the traditional block nested-loop algorithm. Here two pruning rules are utilized: (a) first triangular inequality on the data point's outlier score is used, and then (b) the outlier score is compared with the minimum score required to be an outlier. The second check is similar to that of ORCA [20]. However, while ORCA starts with a cutoff of 0, in MIRO the initial cutoff is obtained from the first phase, and hence converges faster. Though the pruning rules seem simple, their combined effect is strong and efficiently reduces the search space. The main contributions of this chapter can be summarized as follows:

- We analyze the problem of outlier detection from the outlier score perspective and introduce the concepts of global and local outlier score functions. This gives a summary classification of all existing detection techniques.

- We demonstrate high improvement in execution time by using multiple pruning rules in two phases, compared with outstanding existing nested-loop distance-based methods, ORCA [20] and RBRP [37].

- We illustrate the effectiveness of our pruning rules on the overall detection process and give a detailed theoretical analysis on how those rules lead to the superior performance of MIRO. With extremely low CPU cost, MIRO is very suitable for detecting outliers in streaming environments as well as other real-time applications.

The rest of this chapter is organized as follows. We present our MIRO approach in Section 4.2, and theoretically analyze its complexity in Section 4.3. Then we empirically compare our approach with other current-best approaches using real-world datasets in Section 4.4. Finally, we provide a brief summary of the proposed technique in Section 4.5.

4.2 The MIRO Detection Approach

Our approach operates in two phases and employs three pruning rules. In the first phase, we partition DS into clusters, and compute upper and lower bounds of the outlier score for each cluster. Based on these bounds, some clusters are pruned, and the remaining candidates are sent for final processing in the traditional block nested-loop algorithm. Here two pruning rules are utilized: (a) first triangular inequality on the data point's outlier score is used (R_1), and then (b) the outlier score is compared with the minimum score required to be an outlier (R_2). The second check is similar to that of ORCA, however in MIRO the initial cutoff is obtained from the first phase (instead of using 0 as in ORCA), and hence converges faster. The additional overhead of the first phase is offset by the reduction in cost of the second phase. While preprocessing by clustering has been proposed in RBRP, our preprocessing phase incorporates the pruning of unnecessary clusters while RBRP's does not. Additionally, the use of the simple triangular inequality in the second phase and the precomputation of the initial cutoff of outlier score before this phase commences, generates the distinct advantages of MIRO's nested-loop compared to that of ORCA. The detailed process is described below.

4.2.1 Cluster-based Pruning

In this phase, we first cluster the dataset DS (using Algorithm 4) and subsequently identify upper and lower bounds of the outlier score for each resultant cluster (using Algorithm 5). Algorithm 4 is in fact based on the clustering algorithm of RBRP, however we have made some modifications. We denote the *expected number of data points per cluster* as n_c. By changing n_c, we can control the degree of homogeneity of clusters, i.e., points that are close to each other in space are likely assigned to the same cluster. It is noted that in our approach, n_c has the same role as the parameter *BinSize* of RBRP. Compared to RBRP, the cost of clustering is saved for those resultant clusters y having $1 < |y|/nc \leq M$, since a) they are re-clustered only once with the number of clusters being $\lfloor \frac{|y|}{n_c} \rfloor \leq M$ and b) the time complexity of K-Means algorithm is proportional to the number of clusters produced. Hence, our clustering algorithm takes less time than that of RBRP.

Let C be the set of clusters obtained as a result of applying Algorithm 4 on DS with predetermined values of M and it. For each cluster $C_i \in C$, let $|C_i|$ denote its cardinality (or the number of data points allocated to C_i), o_{C_i} its centroid, and r_{C_i} its radius. l_{C_i}, u_{C_i} are the estimated lower and upper bounds of the outlier scores of all data points in C_i respectively. These bounds are only estimations since the true bounds can only be known when the true scores of member data points are identified. A data point p by itself is also a cluster C_i with $o_{C_i} = p$, $r_{C_i} = 0$, $l_{C_i} = u_{C_i} = F_{out}(p)$.

Algorithm 4: CLUSTER

```
Input: $M$: the number of clusters, $it$: the number of iterations, $DS$: the dataset to be
       clustered
Output: $B$: the set of clusters
Set $Y = KMeans(M, it, DS)$
foreach cluster $y \in Y$ do
    if $\frac{|y|}{n_c} > M$ then
        $Cluster(M, it, y)$
    else if $\frac{|y|}{n_c} > 1$ then
        Set $Y' = KMeans(\lfloor\frac{|y|}{n_c}\rfloor, it, y)$
        foreach cluster $y' \in Y'$ do
            Add $y'$ to $B$
    else
        Add $y$ to $B$
```

Definition 4.2 [DISTANCE BETWEEN CLUSTERS]
The minimum distance between clusters C_i and C_j is

$$minDis(C_i, C_j) = max\{D(o_{C_i}, o_{C_j}) - r_{C_i} - r_{C_j}, 0\},$$

and maximum distance between clusters C_i and C_j is

$$maxDis(C_i, C_j) = D(o_{C_i}, o_{C_j}) + r_{C_i} + r_{C_j}.$$

Given a cluster $C_i \in C$, we now need to find clusters that potentially contain k nearest neighbors for every point in C_i. So we first find a set of clusters, Min_{C_i}, *closest* to C_i in terms of $minDis()$, containing at least k data points, i.e., $Min_{C_i} \subseteq C \setminus C_i$, s.t. $minDis(C_j, C_i) \leq minDis(C_k, C_i)\ \forall C_j \in Min_{C_i}, C_k \in C \setminus \{C_i \bigcup Min_{C_i}\}$, the total number of data points in $Min_{C_i} \geq k$.

Similarly, we identify a set of clusters, Max_{C_i}, *closest* to C_i in terms of $maxDis()$, which also contains at least k data points in total.

Consider a data point $p \in C_i$. To compute the lower bound of its outlier score, we have to find the *closest* clusters to p in terms of $minDis()$. In order to do this we consider all clusters closest to C_i as well as other data points in C_i (as clusters). So we choose $Min_p = Min_{C_i} \bigcup C_i \setminus p$. In order to estimate the cumulative distance from p to its k nearest neighbors, we order Min_p and choose the top z clusters $M_1 \ldots M_z$ s.t. $\sum_{i=1}^{z-1} M_i < k \leq \sum_{i=1}^{z} M_i$. Now the lower bound of the outlier score of p can be computed as $l_p = \sum_{i=1}^{z-1} |M_i| \cdot minDis(p, M_i) + (k - \sum_{i=1}^{z-1} |M_i|) \cdot minDis(p, M_z)$.

Algorithm 5: PruneClusters

```
$l_{C_i}, u_{C_i}$    estimateBounds $\forall_i C_i \in C$
Identify $C_o$, $l_{C_o}$
Prune $C_i | u_{C_i} < l_{C_o}$
Return $l_{C_o}, C$
```

Similarly we can compute the upper bound of p's outlier score, $u_p = \sum_{i=1}^{z-1} |M_i| \cdot maxDis(p, M_i) + (k - \sum_{i=1}^{z-1} |M_i|) \cdot maxDis(p, M_z)$, where $\{M_1 \ldots M_z\}$ are the top z clusters in Max_p defined as $Max_{C_i} \bigcup C_i \setminus p$.

Definition 4.3 [Bounds of a cluster's outlier score] *The upper and lower bounds of a cluster's outlier score in terms of its contained points are given as:* $u_{C_i} = max\{u_p, p \in C_i\}$ *and* $l_{C_i} = min\{l_p, p \in C_i\}$*, respectively.*

We now use a simple heuristic to prune clusters that do not contain outliers: pick clusters with the largest lower bounds of outlier scores, until we have a total of at least n data points. Let the last cluster picked be C_o. Clusters whose upper bounds of outlier scores are smaller than l_{C_o} cannot contain outliers, and are therefore pruned. This heuristic constitutes the first pruning phase and is presented in Algorithm 5. The value l_{C_o} is passed as an initial seed to the second pruning phase for faster pruning. While the above heuristic correctly prunes clusters containing data points which are all non-outliers, it may allow clusters containing some non-outliers. This happens for all clusters C_i, where $l_{C_i} \leq l_{C_o} \leq u_{C_i}$. This is undesirable, since not all data points in these clusters are potential outliers. In order to resolve this issue, we propose another heuristic called P_{points} which prunes all points $p \in C_i, u_p < l_{C_o}$. Time complexity of MIRO with and without P_{points} is discussed in Section 4.3.1.

4.2.2 Nested-loop Algorithm

After the lower bound on the outlier score is obtained from the first phase, we process the remaining clusters using the traditional nested-loop algorithm similar to ORCA. In the second phase of MIRO (Algorithm 6) we employ two pruning rules (R_1 in line 9 and R_2 in line 13 of Algorithm 6). Similar to [20], we check if the outlier score of the data point is smaller than the current cutoff c on the outlier score (rule R_2). However, while ORCA initializes c as 0, in our second phase, we converge faster by choosing c from the first clustering phase (with or without P_{points}).

Let us consider an arbitrary data point q. If $c > kD(p, q) + F_{out}(q)$, then by our definition of outlier score and using triangular inequality, we can show that $c > \sum_{m \in kNN_q} D(p, m) \geq F_{out}(p)$,

Algorithm 6: FinalProcessing

```
Set $c, C$    $PruneClusters()$
Set $TopOut$    $\varnothing$
foreach remaining cluster $C_i \in C$ do
    Set $A$    $C_i \cup \{\bigcup_{C_1 \in Min_{C_i}} C_1\} \cup \{\bigcup_{C_2 \in Max_{C_i}} C_2\}$
    foreach data point $p \in C_i$ do
        foreach cluster $C_j \in A$ do
            foreach data point $q \in C_j$ do
                if $q \neq p$ then
                    if $(c - F_{out}(q))/k > D(p, q)$ then
                        Mark $p$ as non-outlier
                        Process next data point in $C_i$
                    Update $p$'s $k$ nearest neighbors using $q$
                    if $F_{out}(p) < c$ then
                        Mark $p$ as non-outlier
                        Process next data point in $C_i$
        if $p$ is outlier then
            Update $TopOut$ with $p$
            if $Min(TopOut) > c$ then
                Set $c$    $Min(TopOut)$
```

i.e., $c > F_{out}(p)$. Therefore p is not an outlier and can be pruned. Despite its simplicity, this pruning rule is extremely efficient in the final processing phase as shown in Section 4.4. By using the combination of two pruning rules, the execution time is further reduced, creating a huge advantage over ORCA and RBRP. It is also noted that by reserving Min_{C_i} and Max_{C_i} for each remaining cluster C_i, we are able to limit the search space for each data point $p \in C_i$. More specifically, to process p, in the worst case we only have to scan $C_i \cup \{\bigcup_{C_1 \in Min_{C_i}} C_1\} \cup \{\bigcup_{C_2 \in Max_{C_i}} C_2\}$. The search space is therefore much smaller than the original dataset DS.

4.3 Theoretical Analysis

In addition to the notations stated in Table 4.1, we define the following new terms for analysis: (a) p_1 is the probability that a cluster will be pruned during the first phase, and (b) p_2 is the probability that a data point will be pruned by rule R_1 before it is scanned with the $(k+1)^{th}$ data point among the remaining ones. It is also noted that in practice, $n_c \leq k$ and $n \ll N$. In the following discussion, we present detailed time and space complexity analysis for MIRO.

4.3.1 Time Complexity of MIRO

The execution time cost of the first phase without P_{points} includes (a) the cost of clustering ($S_{cluster}$), (b) the cost of computing upper and lower bounds outlier score for all clusters (S_{bounds}), and (c) the pruning cost ($S_{pruning}$). The expected clustering cost is $O(N \cdot logN)$ according to [37]. Now, for a cluster C_i, we need to identify Min_{C_i} and Max_{C_i}. Since the mean size of each cluster is n_c, on average we have $|Min_{C_i}| = |Max_{C_i}| = \lceil k/n_c \rceil$. A naïve approach sorts all clusters and extracts $\lceil k/n_c \rceil$ clusters for Min_{C_i}/Max_{C_i}, at a cost of $O(\frac{N}{n_c} \cdot log(\frac{N}{n_c}))$. However, we note that only $\lceil k/n_c \rceil$ clusters need to be reserved for Min_{C_i} as well as Max_{C_i}. Therefore a better approach is that for each cluster C_j, we compute the minimum/maximum distance from C_j to C_i and insert the result into the corresponding set. This approach leads to an total cost of $O(\frac{1}{2} \cdot \lceil \frac{k}{n_c} \rceil \cdot \frac{N}{n_c} \cdot (\frac{N}{n_c} - 1))$ over all clusters, which can be simplified to $O(\frac{N^2}{{n_c}^2})$. To estimate the cost of computing upper and lower bounds of the outlier score for each cluster C_i, we compute the cost of measuring the same bounds for each individual data point $p \in C_i$. To obtain p's bounds, we also need to extract $n_c + \lceil \frac{k}{n_c} - 1 \rceil$ clusters (including zero-radius ones) from a set of $n_c + \lceil \frac{k}{n_c} \rceil$ clusters. Since the number of items extracted is nearly no different from the total set of items, we apply the naïve sorting approach discussed above. As a consequence, the total cost incurred is $O((n_c + \lceil \frac{k}{n_c} \rceil) \cdot log(n_c + \lceil \frac{k}{n_c} \rceil))$, i.e. $O(n_c \cdot log(n_c))$. Hence, the cost of computing C_i's bounds $= O({n_c}^2 \cdot log(n_c))$. Therefore, $S_{bounds} = O(\frac{N}{n_c} \cdot {n_c}^2 \cdot log(n_c)) + O(\frac{N^2}{{n_c}^2}) = O(N \cdot n_c \cdot log(n_c)) + O(\frac{N^2}{{n_c}^2})$. To prune the clusters, we need to compute l_{C_o} and scan the whole set of clusters to check their corresponding upper bounds. To compute l_{C_o}, we need to extract $\lceil n/n_c \rceil$ clusters with largest lower bounds from a set of N/n_c clusters. In other words, $S_{pruning} = O(\lceil \frac{n}{n_c} \rceil \cdot \frac{N}{n_c}) + O(\frac{N}{n_c})$. Overall, the approximate overhead incurred by the first phase is:

$S_{phase1} = S_{cluster} + S_{bounds} + S_{pruning} = O(N \cdot logN) + O(N \cdot n_c \cdot log(n_c)) + O(\frac{N^2}{{n_c}^2}) + O(\lceil \frac{n}{n_c} \rceil \cdot \frac{N}{n_c}) + O(\frac{N}{n_c}) = O(N \cdot logN) + O(N \cdot n_c \cdot log(n_c)) + O(\frac{N^2}{{n_c}^2}) + O((\lceil \frac{n}{n_c} \rceil + 1) \cdot \frac{N}{n_c})$.

After the first phase, the number of remaining clusters is $(1 - p_1) \cdot \frac{N}{n_c}$, which implies that the total number of remaining data points is $n_c \cdot (1 - p_1) \cdot \frac{N}{n_c} = (1 - p_1) \cdot N$. Among them, the total number of data points pruned out by the rule R_1 with no more than k distance computations is $p_2 \cdot (1 - p_1) \cdot N$. On the other hand, for each of the data points left, we need to scan the entire cluster C_i as well as Min_{C_i} and Max_{C_i} in the worst case, i.e., the corresponding cost is $O(n_c + 2 \cdot n_c \cdot \lceil k/n_c \rceil)$, which simplifies to $O(3 \cdot n_c + 2 \cdot k)$. Hence the execution time of the second phase in the worst case can be expressed as:

$S_{phase2} = O(k \cdot p_2 \cdot (1 - p_1) \cdot N + (3 \cdot n_c + 2 \cdot k) \cdot (1 - p_2) \cdot (1 - p_1) \cdot N) = O((3 \cdot n_c \cdot (1 - p_2) + k \cdot (2 - p_2)) \cdot (1 - p_1) \cdot N)$.

Hence, the approximate cost of the whole algorithm is:

$S_{phase1} + S_{phase2} = O(N \cdot logN) + O(N \cdot n_c \cdot log(n_c)) + O(\frac{N^2}{{n_c}^2}) + O((\lceil \frac{n}{n_c} \rceil + 1) \cdot \frac{N}{n_c}) + O((3 \cdot n_c \cdot (1 - p_2) + k \cdot (2 - p_2)) \cdot (1 - p_1) \cdot N)$.

We can also reclassify the whole detection process into a more detailed sequence of operations: (a) clustering, (b) identifying neighboring clusters for all clusters, (c) computing the bounds for clusters (we consider the process for each cluster as a operation, so we have N/n_c operations), (d) pruning clusters (N/n_c operations on average) and (e) final processing step ($(1 - p_1) \cdot N$ operations on average). Among them, the cost of the operations (a) and (b) are loglinear and quadratic w.r.t. N, respectively. On the other hand, each of the remaining operations incurs costs independent of N. Furthermore, when p_1 has large values, the execution time of the second phase becomes very small which compensates the overhead incurred by the first phase. In addition, when p_2 receives a large value, a larger portion of the remaining data points after the first phase require no more than k distance computations to be identified as normal records, and a smaller number of these remaining points require more than k distance computations. This fact leads to another reduction of execution time. Besides, the pre-computation of cutoff c helps contribute to further reduction of the execution time. Therefore, practically each of the operation performed in item (e) is nearly constant. By applying the accounting method of amortized analysis, we expect the expensive cost of operations (a) and (b) would be compensated by the remaining inexpensive ones, i.e. the *amortized running time* of each individual operation is inexpensive and non-quadratic w.r.t. N. In the experiments carried out in Section 4.4, we always have max(p_1, p_2)$\geq$ 0.7 which leads to the practical linear execution time w.r.t N. It is also noted that based on our analysis, this quadratic overhead w.r.t. N is common for techniques that utilize similar partition-based strategy such as the one in [68], which though using less pruning rules than MIRO, still reports linear execution time performance w.r.t N.

4.3.2 Time Complexity of MIRO with P_{points}

In the above analysis, we assume that the P_{points} heuristic (c.f. Section 4.2.1) is not used for the first phase. In contrast, if this heuristic is considered, we prune all points whose upper bound of outlier score is less than the cutoff obtained by the clustering phase, so $S_{pruning}$ has to be recomputed. Particularly, after applying l_{C_o} for pruning out clusters, we perform an additional scan on the the set of clusters left. The mean number of clusters to scan is therefore $(1 - p_1) \cdot \frac{N}{n_c}$, and the expected cost for scanning each cluster is n_c. Consequently, the additional cost is $O((1 - p_1) \cdot \frac{N}{n_c} \cdot n_c) = O((1 - p_1) \cdot N)$. The execution time of the first phase then becomes:

$S_{phase1} = O(N \cdot logN) + O(N \cdot n_c \cdot log(n_c)) + O(\frac{N^2}{{n_c}^2}) + O((\lceil \frac{n}{n_c} \rceil + 1) \cdot \frac{N}{n_c}) + O((1 - p_1) \cdot N)$.

From the above expression, it can be observed that the cost of S_{phase1} does not change theoretically whether P_{points} is used or not. But P_{points} is only effective if it does indeed help to prune out more data points after the first phase. We will examine that in Section 4.4.

4.3.3 Space Complexity of MIRO

As mentioned earlier, minimizing I/O cost is neither a focus of techniques in [20, 37, 15] nor of MIRO. Hence, in general MIRO uses space for: (a) storing the data points, and (b) storing the clusters created. Furthermore, the spatial cost for storing each cluster C_i can be simplified to the cost of storing its major components which include: (a) its member data points, and (b) Min_{C_i} as well as Max_{C_i}. This is simplified by space-efficient hash indexes, therefore each C_i takes $O(n_c + 2 \cdot \lceil \frac{k}{n_c} \rceil)$ space on average. Hence, the space complexity of MIRO is $O(N) + O(\frac{N}{n_c} \cdot (n_c + 2 \cdot \lceil \frac{k}{n_c} \rceil))$, which can be simplified to $O(N)$.

4.3.4 Analysis of Parameters Used

4.3.4.1 Cluster size

For a fixed dataset size, as the average cluster size n_c decreases, the total number of clusters will increase. Since the size of each cluster C_i becomes smaller, in order to compute the bounds of C_i, we need to include more clusters in Min_{C_i} as well as Max_{C_i}. In other words, more clusters are required for computing C_i's bounds. That increases S_{bounds} and leads to the increase in the overall execution time of our algorithm. In the extreme case, when $n_c = 1$, the first phase degrades to scanning the entire dataset, i.e., the total execution time becomes a normal nested-loop algorithm and the execution time saved during the second phase becomes insufficient to compensate this overhead. On the other hand, as n_c increases, there are lesser clusters than before. Since the size of each cluster becomes larger, we need to consider fewer clusters in the process of computing clusters' bounds on the outlier score. But that does not directly lead to a decrease in cost of computing bounds since we need to process more data points per cluster. Furthermore, as n_c increases and exceeds k, the lower bound score l_{C_o} becomes smaller since we only need to use data points in a cluster C_i to compute its bounds (the assumption here is that in general a cluster contains data that are relatively homogeneous). That means less clusters are pruned after the first phase hence the execution time will increase. Overall, we should choose a reasonable value of n_c such that the average number of data points per cluster is neither too small nor too large compared to k. More specifically, we need to identify a threshold for n_c such that as n_c increases above as well as decreases below this threshold, the execution time of MIRO will increase. Consequently, picking this threshold to be n_c will be a wise choice. From the above analysis, we conclude that the impact of n_c over the overall performance of MIRO is complex and identification of reasonable values for n_c by analytical methods is practically infeasible. Through empirical study carried out in Section 4.4, we show that $k/5$ is a possible candidate value.

4.3.4.2 Number of nearest neighbors

As the number of nearest neighbors taken into account for the computation of outlier score, k, increases, the value that F_{out} assigns to each individual data point p in DS will increase correspondingly. This in turn leads to an increase in the lower bound l_{C_o}, and hence more clusters may be pruned by the first phase of MIRO. However, as demonstrated before, an increase of k results in having to consider more clusters when computing outlier score bounds for an arbitrary cluster. Therefore, the cost of computing cluster's bounds will increase. The increase of k creates a two-fold effect: (a) a decrease in execution time since more data points are pruned, and (b) an increase in execution time due to the increase in the cost of computing clusters' bounds. Our experimental result in Section 4.4 shows that MIRO's execution time increases as k increases, i.e., the latter factor outperforms the former one.

4.4 Empirical Results and Analyses

In order to assess the effectiveness of our proposed technique, we performed extensive experiments on six real and high-dimensional datasets. For each set of input parameters that affect the performance of the corresponding algorithm, we ran the experiment for ten times. The results presented are from average outcomes obtained from multiple runs. Datasets used for evaluation include CorelHistogram, ColorMoments, CoocTexture, Covertype, Server [2] and Landsat [3], and their brief characteristics are presented in Table 4.2. All of them are original datasets except for Server which is extracted from KDD Cup 1999 data [2], using the procedure provided in [78]. These datasets are used widely by popular techniques of the field [15, 20, 37, 78]. All experiments were conducted on a Pentium 4 computer with a 3.4 GHz processor and 1GB RAM. It is noted that we set $M = 10$ and $it = 5$ throughout all experiments. In this section, we present the corresponding empirical results. In particular, we demonstrate:

- The efficiency of MIRO in reducing the execution time of the traditional nested-loop algorithm. We measure the scalability of MIRO's execution time against the dataset size (N) as well as the number of nearest neighbors (k) used. In the latter case, we present MIRO's performance with and without P_{points}. The result is then compared with ORCA and RBRP to highlight the merit of our method.

Dataset	No. of attributes	No. of instances
CorelHistogram	32	68,040
ColorMoments	9	68,040
CoocTexture	16	68,040
Covertype	54	581,012
Landsat	60	275,465
Server	5	500,000

Table 4.2: Characteristics of datasets

- The pruning power of MIRO, in both phases of processing, with and without P_{points}. In addition, we also assess the effect of k on the pruning quality. The sensitivity of MIRO's execution time with respect to the cluster size (n_c) is also presented.

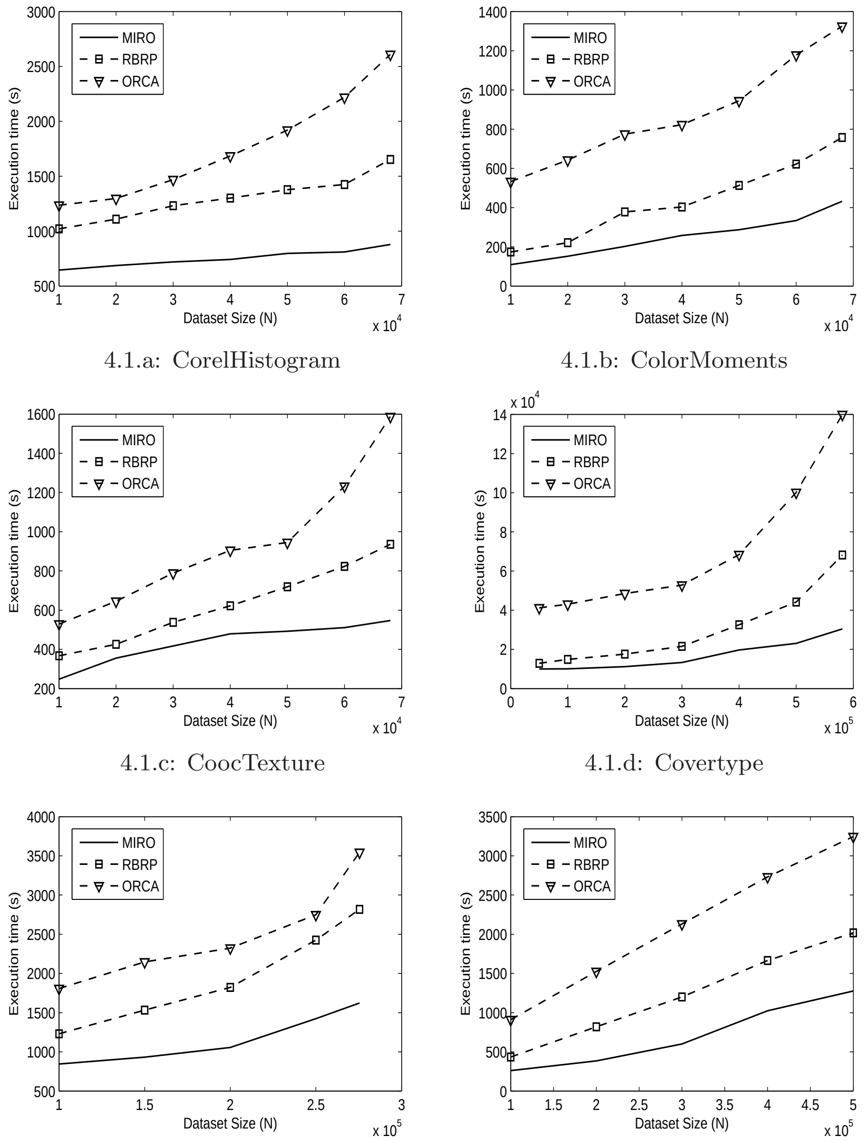
4.4.1 Execution time v/s. N

First we evaluate the scalability of execution time of three distance-based outlier detection techniques MIRO, RBRP and ORCA w.r.t the dataset size N. In this experiment, we chose the number of outliers mined $n = 30$, number of nearest neighbors $k = 50$, set the size of each cluster $n_c = 20$, and varied N. We chose the implementation of MIRO without P_{points} since the efficiency of P_{points} is highlighted in a later part of this section. We observe from the result (Figure 4.1) that MIRO scales better than RBRP and ORCA on all datasets, although its theoretical asymptotic time complexity is quadratic in N. This agrees with the amortized analysis in Section 4.3.1.

In order to analyze the cause of MIRO's efficiency, we also compare the execution time with and without the first phase. Table 4.3 presents the speedup of the execution time on the original dataset size. It can be seen that in all cases, MIRO achieves a speedup from 2 to more than 4 times which is good enough to serve our proposed technique's main purpose (reducing execution time). This clearly illustrates that the clustering strategy benefits the pruning phase.

4.4.2 Execution time and MIRO's pruning power v/s. k

We now analyze the effect of the number of nearest neighbors (k) on execution time. This experiment is conducted on the entire datasets, and $n = 30$, $n_c = 20$ as in the previous case. The results (Figure 4.2) show that the execution time for every technique increases with k, but MIRO scales better (with and without P_{points}) compared to RBRP and ORCA. The reason is once again attributed to the effective pruning power of MIRO in both phases of processing. It



4.1.a: CorelHistogram

4.1.b: ColorMoments

4.1.c: CoocTexture

4.1.d: Covertype

4.1.e: Landsat

4.1.f: Server

Figure 4.1: Execution time vs. the dataset size N.

Dataset	**Speedup**
CorelHistogram	2.92
ColorMoments	3.03
CoocTexture	2.85
Covertype	4.41
Landsat	2.14
Server	2.42

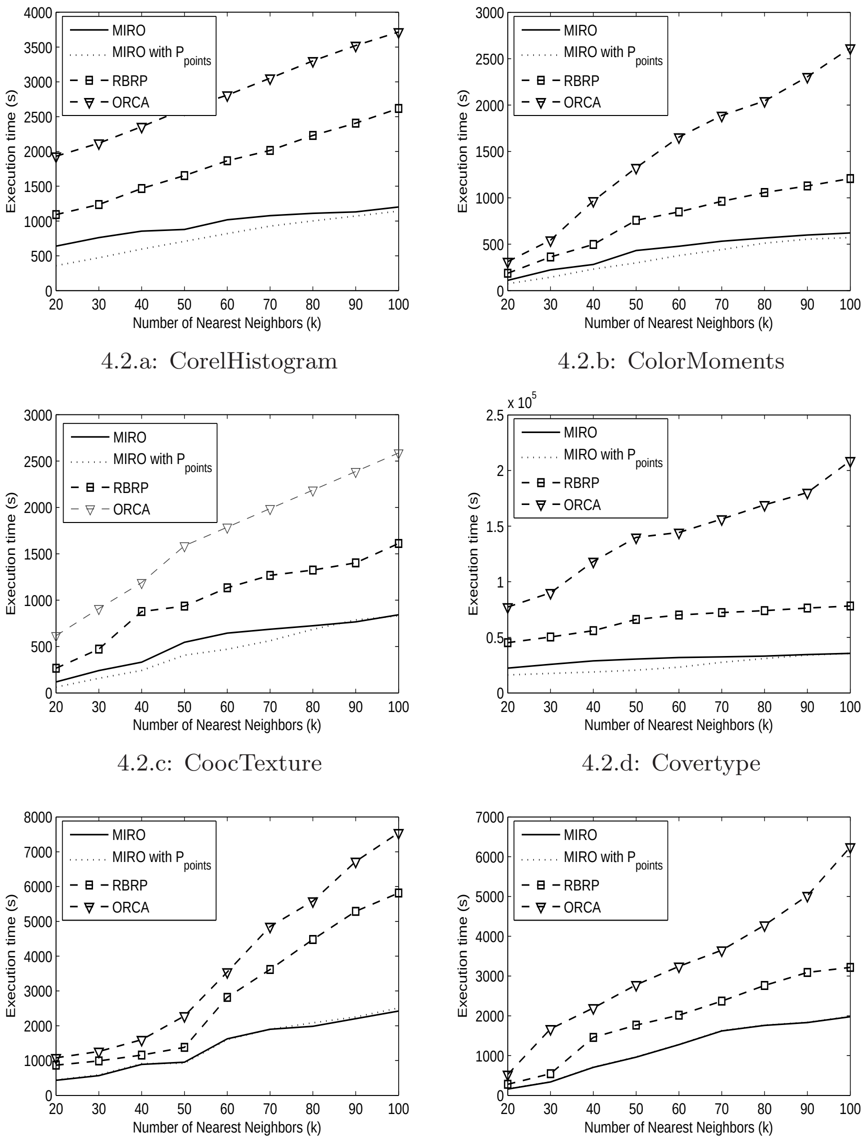
Table 4.3: Benefit of using the first phase of clustering

is also clear that by using P_{points}, we are able to obtain better or equal performance in term of execution time. This observation is further analyzed later when we discuss the effect of k on MIRO's pruning power.

Figure 4.3 presents two pruning probabilities in one plot for each dataset: the probability of pruning a cluster in the first phase (p_1), and the probability that a data point will be pruned out by rule R_1 before it is scanned with the $(k+1)^{th}$ data point among the remaining ones (p_2), as the number of nearest neighbors is varied. In all cases, very high values of p_1 and/or p_2 are achieved, with p_1 increasing when P_{points} is utilized. As explained in Section 4.3, a high value of p_1 implies that we need to process less data points in the nested-loop phase, while a high value of p_2 shows that majority of the data records left in the second phase need no more than k distance computations per record. While we do not obtain high values for both p_1 and p_2 at the same time, we observe that in every case at least one of them receives a value greater than 0.7. This reflects a very high efficiency in pruning and explains why MIRO takes lesser execution time compared to RBRP and ORCA. In addition, the value of p_1 tends to increase as k increases (except in the case of Landsat dataset), which means more clusters will be pruned after the first phase when k receives higher value. This agrees with the discussion in Section 4.3.4. Furthermore, when p_1 without P_{points} already has relatively large value, applying P_{points} does not help much in increasing the pruning power of the first phase. This point is reflected by the tendency of p_1 with and without P_{points} to converge towards each other as p_1 increases. We also observe that when the pruning effect without using P_{points} is low, i.e., when p_1 is low, there will be a significant improvement in execution time if P_{points} is employed instead. This can be attributed to the fact that adjoining clusters' lower and upper outlier score bounds are too interleaved with each other which creates redundancy if we include the whole of each candidate cluster in the final processing step. In contrast, if the value of p_1 is already high, which means l_{C_o} has been identified wisely, using P_{points} may not improve MIRO's performance by much, although the pruning effect obtained is still equal or better. The reason is that increase in pruning power in such cases is not enough to compensate the additional time spent to run P_{points}. However, it is noted that when p_1 receives a higher value, the cost of executing P_{points}, which is $O((1-p_1)\cdot N)$, becomes lower. Therefore, it can be concluded that applying P_{points} does not degrade performance by much, but may lead to significantly better performance.

4.4.3 Execution time v/s. n_c

We now study the effect of the average cluster size (n_c) on the execution time of MIRO. In this experiment, we set $n = 30$. By varying k, we run MIRO with $n_c \geq 1$ and $\leq k$ and take note



4.2.a: CorelHistogram

4.2.b: ColorMoments

4.2.c: CoocTexture

4.2.d: Covertype

4.2.e: Landsat

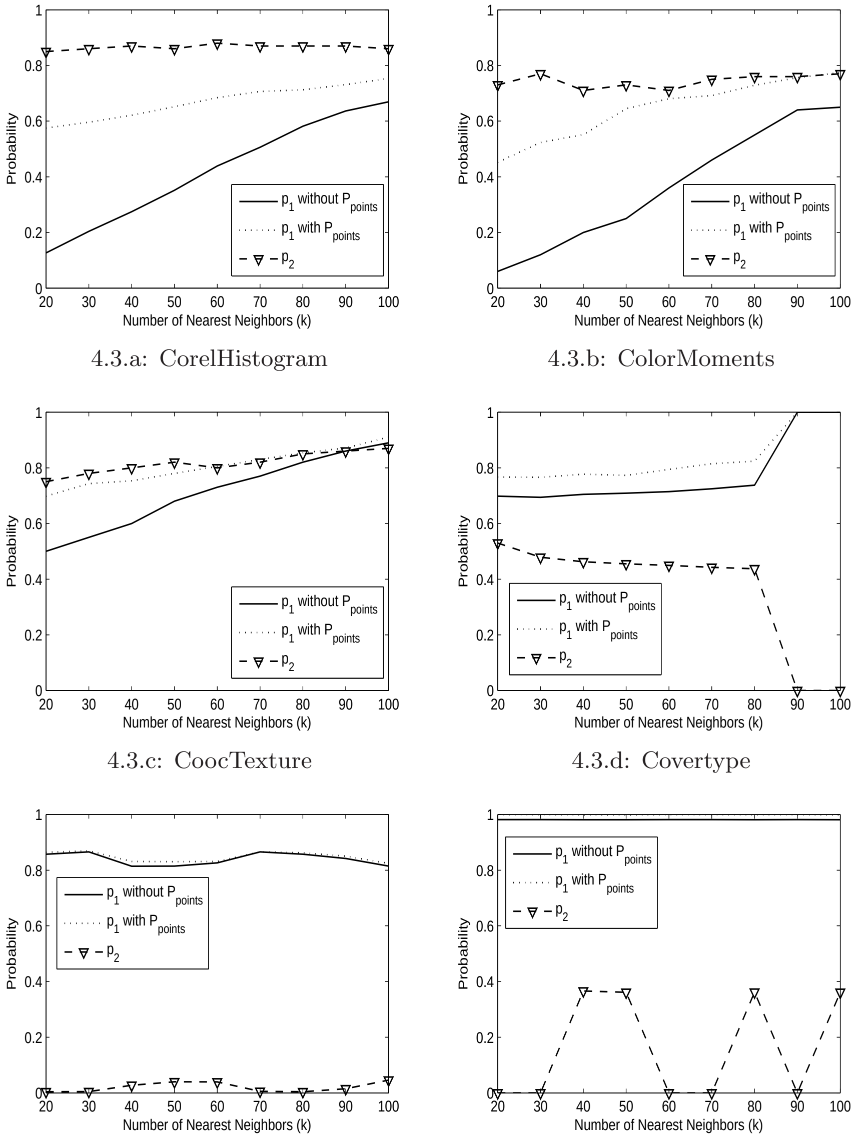
4.2.f: Server

Figure 4.2: Execution time vs. the number of nearest neighbors k.

the value of n_c which yields smallest CPU cost. The result obtained suggests that n_c should be $k/5$. As a representative, we only show the execution time of MIRO on all datasets for $k = 50$ (Figure 4.4). In this figure, the left y-axis corresponding to the Covertype dataset, and the right y-axis for the rest. We can see that MIRO's execution time is nearly linear with $n_c \geq 10$. With $n_c = 9$, the execution time begins increasing (this increase is slightly small for Covertype dataset) and becomes worst at $n_c = 1$ where the performance downgrades to that of a normal nested-loop algorithm. These findings agree with our discussion in Section 4.3.4. Figure 4.4 confirms that with $k = 50$, the best choice of n_c is $k/5 = 10$. A good selection of n_c helps to balance the tradeoff between the time spent on computing clusters' bounds, as well as the pruning effect of the first phase of MIRO. In practice, we can also determine n_c by performing a training process on a subset of the original dataset with $n_c = k/5$ as the initial seed.

4.5 Summary

This chapter presents a new combination of several pruning strategies to produce an efficient distance-based outlier detection technique. The proposed technique, MIRO, consists of two pruning phases of processing which lead to amortized efficiency. During the first phase, a partition-based technique is employed to extract candidate clusters for the later processing step. Furthermore, an additional benefit of the first phase is that we are able to compute an initial value of the outlier cutoff threshold which is utilized in the nested-loop phase. In the second phase of MIRO, two pruning rules are employed to further reduce the overall temporal cost. Extensive empirical studies demonstrate that MIRO can outperform outstanding related techniques in the field.



4.3.a: CorelHistogram

4.3.b: ColorMoments

4.3.c: CoocTexture

4.3.d: Covertype

4.3.e: Landsat

4.3.f: Server

Figure 4.3: MIRO's pruning power vs. the number of nearest neighbors k.

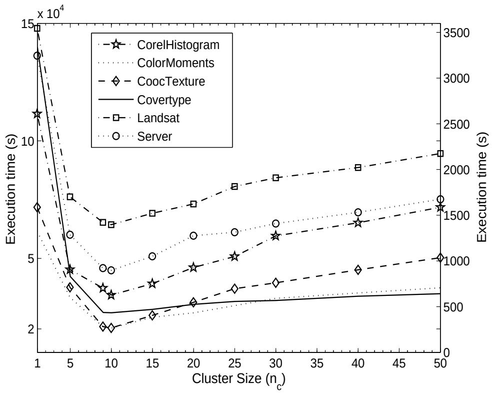


Figure 4.4: Execution Time of MIRO vs. the average cluster size n_c.

Chapter 5

Future Work

We would like to explore the following research directions which we firmly believe to be very important for outlier detection research.

5.1 Detecting Outliers in Concept-Drift Environment

A concept can be seen as a mapping between a set of (functions on) independent variables and a dependent variable. In many cases, it is assumed that this mapping is constant and does not change with time. However in many domains, e.g. data streams, this is not necessarily true. When this mapping in the phenomenon of interest changes over time, concept-drift is said to have occurred. Being able to detect concept-drift means being able to track the changes and update the concept mapping (model) accordingly. As an example, consider an education subsidy policy based on educational levels of various ethnic groups. A recent study in the U.S. [33] shows that from 1990 to 2005, there was a general increase in the percentage of adults (aged 25 and over), who had completed high school. The study also found that while the percentages of White, Black, Hispanic, Asian/Pacific Islander, and American Indian/Alaska Native adults with bachelor's degrees have increased, the distribution in each individual group is different. During this period, the gap between White and Black adults who had completed high school has narrowed down from 15% to 9%, while there was no significant change in the White-Hispanic high school completion gap (31% in 1990 and 32% in 2005). It is also observed that the percentages of Blacks and Whites who completed higher (tertiary) degrees have increased, while those of other races/ethnicities were similar in the same period. This study demonstrates a few keys points: (a) there is a drift in the education levels of U.S. citizens, and (b) the drift differs among ethnic groups. By capturing the knowledge of this concept-drift, it may be possible to devise more effective schemes to increase overall national education quality. For example, the study indicates that the high school educational policies for Blacks have been

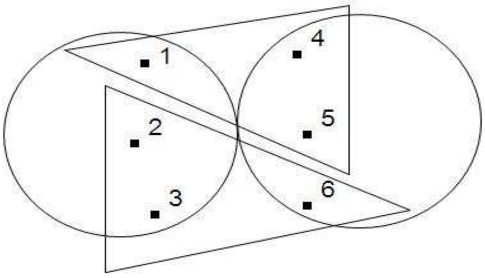


Figure 5.1: Incremental clustering example.

effective, but Hispanics may need more incentives. The study also shows that further graduate education subsidies for Natives, Hispanics and Asians would be more valuable, rather than an across-the-board option.

Besides concept-drift, model building methods on data streams also have other problems. As mentioned in [45], incremental clustering methods share a common property which is also a drawback: they are order-dependent. An approach is order-independent if it generates the same result regardless of the order in which data are presented, otherwise it is said to be order-dependent. In other words, the result generated by an order-dependent technique depends on the arrangement of new coming data points. If that arrangement is changed, we might obtain a different result. We use the example shown in Figure 5.1 for illustrating this property. If the order of incoming data is 1, 2, 3, 4, 5 and 6, the two clusters created by the underlying clustering algorithm are as denoted by the two circles. On the other hand, if the order is 1, 4, 5, 2, 3, and 6, the two triangles outline the resultant clusters.

Let assume we want to build a technique to detect outliers in data streams that contain concept-drift. Because of drift, it is dangerous if no outlier concept updating strategy is available. In other words, we may miss some important knowledge which is initially not available in the monitored streams. It is therefore more reasonable if we have a mechanism to take advantage of outliers detected to improve the designed model (including updating concepts about outliers) instead of simply discarding them. Despite the importance of detecting outliers in concept-drift environment, there are currently very few related articles in the field whereas there are still several open issues. Particularly, a technique that is capable of detecting outliers in a concept-drift environment has to specifically handle the following points:

- **Adaptation to concept-drift:** Concept-drift causes the underlying model to become outdated, since new concepts appear in data. The technique therefore should be able to detect these changes and perform necessary updates to cope with the current trend in data.

- **Memory constraints:** In a streaming environment, data grow with time and it is impossible to store them on disk. The designed technique should contain a mechanism to extract and store only the relevant characteristics of historical data for future learning. Furthermore, assume the detected algorithm is A. For each incoming data point p, the number of times p is scanned by A should be minimized. In the ideal case, A is expected to be a single-pass algorithm, i.e. each incoming data point is scanned only one time. This requirement once again stems from the limited memory constraints.

- **Resistance to order-dependent effect:** The order-dependent effect stems from the nature of data streams themselves, and hence, is very difficult to eliminate completely. Designing a scheme for reducing that effect which leads to more stable model for the learning process is therefore desired.

Besides the issues mentioned above, it is also noted that outlier score of each incoming data point can only be computed using incomplete knowledge about the data stream [66]. This fact is natural and the outlier score here hence is just temporary score. However, as long as the score is able to reflect the current trend in the data stream, the model can be considered as functioning successfully. Furthermore, by nature, the updating outlier concept should not exclude the process of updating concepts of normal data except for the case when knowledge about normal samples is not important for our learning purpose. In other words, the current trend of data should be captured so that the result of prediction becomes more accurate. Therefore, a designed technique for dealing with such kind of problem will converge very closely to an algorithm used to deal with concept-drift. More specifically, there may be a convergence between concept-drift research and detecting outliers in streaming environment research.

5.2 Visualization of Detection Results

When the purpose of detecting outliers is just for cleaning the dataset to perform other data mining tasks (e.g. clustering), the descriptions of outliers detected are usually not required. On the other hand, when we want to know the characteristics of outliers detected to make some critical decisions, being able to visualize the detection results will make the learning process far more efficient and effective than simply presenting the anomalous points per se. However, creating such visualization is nontrivial. The first visualization tool is introduced by Papadimitriou et al. [67], called LOCI plot. The LOCI plot summarizes information about the points in its vicinity, determining clusters, micro-clusters, their diameters and their inter-cluster distances. The modified definition of LOCI plot is provided in Definition 5.1.

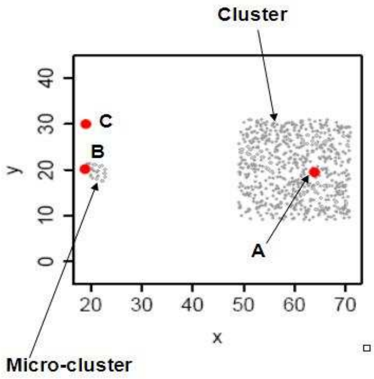


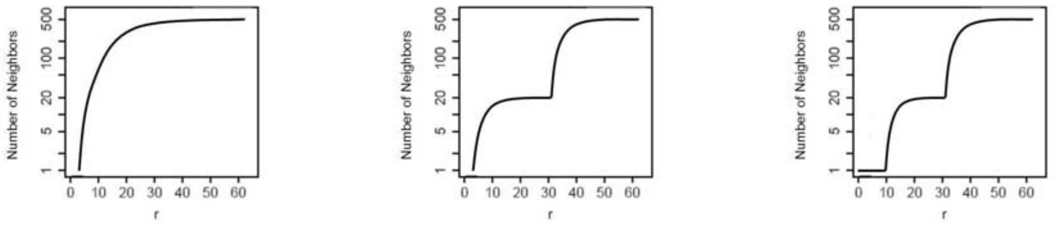
Figure 5.2: Synthetic dataset DS_{syn}.

Definition 5.1 [LOCI plot] *For any data point p in the dataset, the plot of the density of r-neighborhood of p against r is called its LOCI plot.*

One may find that the above definition is slightly different from the original one but the idea is still the same: LOCI plot is a plot that describes the relationship between the neighborhood of a data point and the radius of that neighborhood. Figure 5.3 shows an example of the LOCI plot (proposed in [67]) of three data points A, B and C of the synthetic dataset DS_{syn} (Figure 5.2, also proposed in proposed in [67]). DS_{syn} contains a big cluster, a micro-cluster and an outlier (C). For A which belongs to the big cluster, its LOCI plot does not contain any sudden change as the radius of neighborhood r increases. In contrast, the LOCI plot of data point B belonging to the micro-cluster contains a "jump" at $r = 30$. This phenomenon reflects the fact that we will encounter more data points belonging to the big cluster at $r = 30$. Similarly, the LOCI plot of outlier C contains two sudden "jumps" at $r = 10$ and $r = 30$ respectively. Once again, this can be attributed to the fact that the minimum distance between C and the micro-cluster is 10 while it is 30 for the big cluster. If such plot can be constructed during runtime, we will then be able to select the suitable value for the neighborhood radius. More specifically, the maximum value of r should be less than or equal to the value where a jump occurs in the plot (for Figure 5.3, $r \leq 10$). Furthermore, by examining the plot, we are able to explore more about the intuition of flagging a data record as outlier. In the remaining of this section we discuss about the benefits of outlier visualization tools and suggest possible ways to construct such representation.

5.2.1 Benefits of Result Visualization

First, detection results themselves are less human intuitive if the users have no idea about the behavior of other data records in the dataset. They do not know why a data point should be an



5.3.a: LOCI plot of pattern A

5.3.b: LOCI plot of pattern B

5.3.c: LOCI plot of pattern C

Figure 5.3: An example of LOCI plot.

outlier and why others are not. On the other hand, data points will be nearly the same without being embedded into a specific context. In that case, an intuitive representation is required to provide users a general idea about the data's *outlier-ness*. Furthermore, visualization is useful for decision making. For instance, when it is still not clear which threshold(s) to use in extracting outliers as well as how many outliers we should flag, then a plot of outlier score against some parameter(s) we need to consider or a plot of outlier score for the entire dataset may help. Finally, even when the visualization tools cannot be constructed at runtime (e.g. in fast streaming data), they are still beneficial for the learning process in which knowledge on outliers are studied to overcome the problem of concept-drift and hence, improving the accuracy of detection techniques in later stages of detection. Being able to visualize the detection results brings an entirely new perspective to outlier detection, exploration, and identification of novel and exciting knowledge.

5.2.2 Possible Forms of Visualization

In this part, we present some possible schemes that can be used as an visualization tool. The details are as follows:

- **A plot of data points neighborhood against some parameter(s):** LOCI plot [67] is a typical example for such kind of plots. They help us in determining which parameter should be used in the detecting process. In Figure 5.3, at radius $r = 10$ there is a jump of neighborhood of the anomalous data point. Hence r should be chosen to be ≤ 10 so that the designed technique is still able to detect that outlier. As mentioned above, this tool can be not only constructed during training phase but also used at runtime to assist in choosing suitable values for the key parameters affecting the algorithm performance. Therefore, better decision can be made by using such visual representation.

- **A plot of outlier score for the entire dataset:** A plot of outlier score for the entire dataset will provide us an insight about what threshold values we should use for the outlier score (if the detection technique falls into Threshold-based) or how many outliers we should detect (if the detection technique falls into Ranking-based). A too high or too low threshold can therefore be avoided. If Threshold-based is used, this plot can be used not only in training phase but also in testing phase. For instance, we add the outlier scores found so far to the plot and hence, we can dynamically choose a suitable cutoff value for outlier scores for the rest of the dataset. In case Ranking-based is used, we can estimate in testing phase how many percents of the dataset outliers are present. Using this percentage, in runtime we can choose the number of outliers we need to detect. An example of such plot is shown in Figure 5.4 where the underlying score function used is LOF. There are three obvious outliers with outstandingly largest LOF values while the other three (denoted as *potential outliers*) are just slightly larger than the rest. Hence, setting the number of outliers to mine to be 3 is more reasonable and intuitive than 6 in this case.

- **Summarized descriptions of both normal and abnormal data points:** Similar to clustering, normal and abnormal groups of data points can be described using conjunctive logical expressions, expressing in the form of decision tree [45]. This form of representation can be used to create a 'signature' to characterize outliers. However it may become lengthy if the outliers detected cannot be grouped together. Fawcett et al. [32] introduce the first rule-based system for detecting fraudulent activity or news story monitoring. The rule-based module may be either a classifier learning classification rules from both normal and abnormal training data or a recognizer trained on normal data only and learning rules to pinpoint changes that identify fraudulent activity. The learned rules create profiling monitors for each rule modeling the behavior of a single entity, such as a phone account. When the activity of the entity deviates from the expected, the output from the profiling monitor reflects this. In fact, the proposed rule-based systems is very similar to decision trees as they both test a series of conditions before producing a conclusion (class) about data records.

- **A ROC plot:** As mentioned in Section 3.2, this plot can be constructed easily when underlying datasets are well labeled. ROC plot can be constructed in training phase to test the quality of the detection technique. Apart from that, users can also choose the combination of (detection rate, false alarm rate) which reflects their acceptable trade-off. For example, based on ROC curve in Figure 2.1, users may decide to choose the tuple (detection rate = 0.7, false alarm rate = 0.3) as the level of tolerance during testing phase.

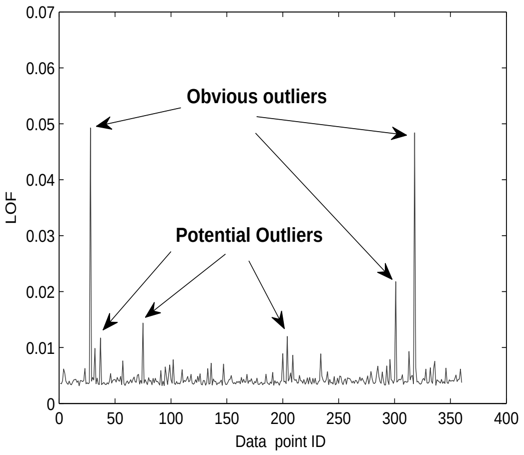
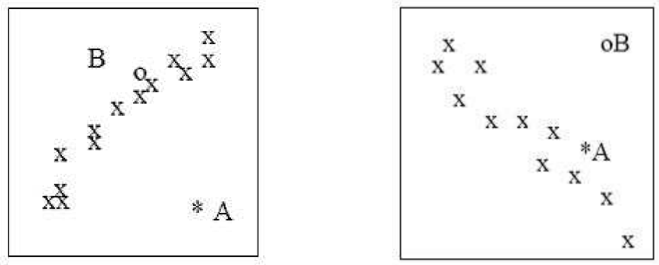


Figure 5.4: An example of outlier score plot.

So far in this section, we have introduced some possible forms of outlier visualization. To the best of our knowledge, LOCI is the only technique that provide such detailed graphical and interactive representation. Obviously, the future of detection result visualization is limited only by our imaginations. More research is required to build and assess the effectiveness of such tools.

5.3 Dimension Reduction in Outlier Detection

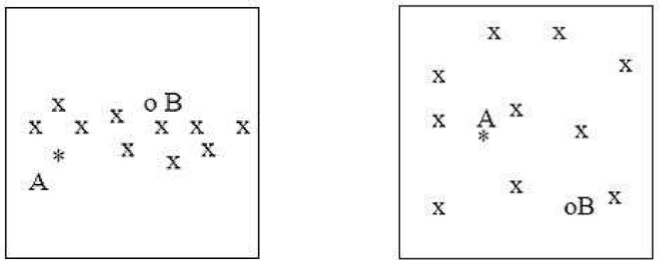
Similar to other data mining applications, choosing a suitable subset of features to apply in the detection process is very important. For high dimensional dataset the concept of locality as well as neighbors becomes less meaningful [21]. This is because in high dimensional space, data distribution becomes very sparse. Therefore, the distance between data points tend to cluster around a specific value. It has been shown recently that by examining the data in their subspaces, we are able to develop more effective techniques for clustering [74, 75] as well as similarity search in multi-dimensional datasets [5, 9]. Furthermore, in high dimensional datatsets, it is more difficult to detect points that are outliers because of the averaging behavior of the noisy and irrelevant dimensions. Another problem caused by the curse of dimensionality is that the considered dataset may contain some noisy attributes which may deteriorate the impact of the distance function used, especially for those distance functions that are very sensitive to noise (e.g. Euclidian distance). Pruning out some specific attributes prior to the detection process is also not straightforward. In outlier detection, we are interested in abnormal points; by discarding some certain features we may loose some interesting knowledge. We use



5.5.a: Projection where A is outlier

5.5.b: Projection where B is outlier

Figure 5.5: Projections where A and B yield abnormal behavior.



5.6.a: Dense Projection

5.6.b: Sparse Projection

Figure 5.6: Projections where A and B behave normally.

the examples in Figure 5.5 and 5.6 (proposed in [7]) to illustrate our argument. According to Figure 5.5, it can be seen that in the two projections, data points A and B yield abnormal behavior, i.e. they are outliers. However, in the other two projections (Figure 5.6), A and B have normal behavior. Careless elimination of dimensions where anomalous behavior is uncovered leads to loss of critical knowledge. Thus, choosing a combination of features to mine is a very complicated process.

Despite of this issue's importance, there is less effort done so far. To the best of our knowledge, there are only two existing articles by Aggarwal et al. [7] (EvoSearch) and Lazarevic et al. [56] (FeatureBagging) which address this problem. Both the proposed approaches first fix the number of dimensions d_t to use in the detection process and then mine outliers using the chosen attributes and hence, do not perform any test of significance on each individual feature. However, there are some differences between them. Particularly, while the former technique utilizes a evolutionary approach to mine outliers over the entire search space of d_t-projections, FeatureBagging on the other hand mines outliers using a predetermined set of attributes. That makes FeatureBagging's mechanism of selecting relevant attributes less systematic and intuitive. As pointed out in Section 2.4, EvoSearch is sensitive to the selection of initial population size, crossover and mutation probabilities. Further more, the fix of the number of dimensions to mine is also not intuitive since outliers may present in projections differing in the number of dimensions. Assume the dimensionality of the considered dataset is *dim*. A test on all 1-dimensional, 2-dimensional, . . . , to *dim*-dimensional projections is desired. This is however too expensive. We already have research about *frequent itemset mining* in which combinations of items with high frequency is mined. The problem in high-dimensional outlier detection on the other hand can be seen as *infrequent itemset mining* where each item is a dataset attribute. More work is needed to discover the possibility of applying already well-established theories of frequent itemset mining in high-dimensional outlier detection problem.

Chapter 6

Conclusion

In this report, we have provided a comprehensive study on the existing work in outlier detection. In addition, we have analyzed the limitations of the related approaches which serve as a guide for the design of our solution.

As of now, we have developed an efficient algorithm, HeDES, that is able to combine the detection results of different techniques to improve the overall accuracy. Similar to Feature Bagging [56], HeDES builds different detectors in the ensemble by choosing different random subspaces. That helps to avoid the repetition of the same detection errors by detectors. At the same time, it somehow overcomes the curse of dimensionality. Unlike Feature Bagging, HeDES does not assume the availability of outlier scores produced by different algorithms. Instead, it works based on a careful classification of outlier scores as well as different treatments for different types of scores. Empirical comparison with all existing techniques has pointed out that HeDES is able to outperform all of them in terms of detection accuracy.

As an achievement of our research, we have also addressed the issue of reducing the computational complexity in distance-based outlier detection. Our proposed solution, MIRO, operates in two phases: clustering and nested-loop. During clustering phase, data are grouped into clusters using a divisive clustering algorithm. Thus, data points which are near to each other are likely to lie in the same cluster. The upper and lower outlier score bounds of each cluster are estimated and pruning is performed based on those bounds. As a result of cluster pruning, only a few clusters are left before we enter the nested-loop phase. In the nested-loop phase, two pruning rules based on triangular inequalities are employed to early terminate the neighborhood search of normal data points. Computational cost is therefore further reduced. Comprehensive studies on real datasets have confirmed the efficiency of MIRO compared to outstanding techniques in the field. For future work, we want to tackle the issue of reducing I/O cost using the same notion of distance-based outliers. For the notion proposed by Ng et al. [49], a technique that perform excellently in both worlds (CPU and I/O costs) can be found in [13]. However, it is inapplicable to the outlier definition that we are interested in. That motivates us to explore such a solution which if available will bring great benefits to distance-based outlier detection community.

In the future, we would like to explore solutions for addressing the issues of concept drift, dimension reduction, and detection result visualization. By solving them, we will be able to extend the applicability of anomaly detection in different contexts, and hence, broadens its application domains.

Chapter 7

List of Author's Publications

Accepted Papers

- Hoang Vu Nguyen, Vivekanand Gopalkrishnan, Praneeth Namburi. Online Outlier Detection Based on Relative Neighbourhood Dissimilarity. In *Proceedings of the 9th International Conference on Web Information Systems Engineering*, pages 50–61, 2008.

- Hoang Vu Nguyen, Vivekanand Gopalkrishnan. Efficient Pruning Schemes for Distance-Based Outlier Detection. In *Proceedings of the 13th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 160–175, 2009.

- Hoang Vu Nguyen, Vivekanand Gopalkrishnan. On Scheduling Data Loading and View Maintenance in Soft Real-time Data Warehouses. In *Proceedings of the 15th International Conference on Management of Data*, 2009.

- Hoang Vu Nguyen, Hock Hee Ang, Vivekanand Gopalkrishnan. Mining Outliers with Ensemble of Heterogeneous Detectors on Random Subspaces. In *Proceedings of the 15th International Conference on Database Systems for Advanced Applications*, pages 368–383, 2010.

- Hoang Vu Nguyen, Vivekanand Gopalkrishnan. Feature Extraction for Outlier Detection in High-Dimensional Spaces. In *Proceedings of the 4th International Workshop on Feature Selection in Data Mining*, pages 64–73, 2010.

Work Under Submission

- Hoang Vu Nguyen, Vivekanand Gopalkrishnan. On Efficient Distance-Based Outlier Detection in High-Dimensional Spaces.

- Hoang Vu Nguyen, Vivekanand Gopalkrishnan, Ira Assent. An Unbiased Distance-Based Outlier Detection Approach for High-Dimensional Data.

References

[1] Bloomberg news. `http://www.bloomberg.com`.

[2] UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[3] Vision research lab. `http://vision.ece.ucsb.edu`.

[4] N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 504–509, 2006.

[5] C. C. Aggarwal. Re-designing distance functions and distance-based applications for high dimensional data. *ACM SIGMOD Record*, 30(1):13–18, 2001.

[6] C. C. Aggarwal. On abnormality detection in spuriously populated data streams. In *Proceedings of the 5th SIAM International Conference on Data Mining*, 2005.

[7] C. C. Aggarwal and P. S. Yu. An effective and efficient algorithm for high-dimensional outlier detection. *International Journal on Very Large Data Bases*, 14(2):211–221, 2005.

[8] C. C. Aggarwal and P. S. Yu. Outlier detection with uncertain data. In *Proceedings of the 8th SIAM International Conference on Data Mining*, pages 483–493, 2008.

[9] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[10] D. W. Aha and R. L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the AAAI Workshop on Case-Based Reasoning*, pages 106–112, 1994.

[11] F. Angiulli, S. Basta, and C. Pizzuti. Distance-based detection and prediction of outliers. *IEEE Transactions on Knowledge and Data Engineering*, 18(2):145–160, 2006.

[12] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 811–820, 2007.

[13] F. Angiulli and F. Fassetti. Very efficient mining of distance-based outliers. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 791–800, 2007.

[14] F. Angiulli and F. Fassetti. DOLPHIN: An efficient algorithm for mining distance-based outliers in very large datasets. *ACM Transactions on Knowledge Discovery from Data*, 3(1), Article 4, 2009.

[15] F. Angiulli and C. Pizzuti. Outlier mining in large high-dimensional data sets. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):203–215, 2005.

[16] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 164–169, 1996.

[17] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 260–264, 2000.

[18] D. Barbará, C. Domeniconi, and J. P. Rogers. Detecting outliers using transduction and statistical testing. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 55–64, 2006.

[19] V. Barnett and T. Lewis. *Outliers in Statistical Data, 3rd edn.* John Wiley and Sons, 1994.

[20] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 29–38, 2003.

[21] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is “nearest neighbor” meaningful?. In *Proceedings of the 7th International Conference on Database Theory*, pages 217–235, 1999.

[22] C. M. Bishop. Novelty detection and neural network validation. *IEE Proceedings on Vision, Image and Signal Processing*, 141(4):217–222, 1994.

[23] C. Böhm, K. Haegler, N. S. Müller, and C. Plant. Coco: coding cost for parameter-free outlier detection. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 149–158, 2009.

[24] T. Bozkaya and Z. M. Özsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems*, 24(3):361–404, 1999.

[25] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

[26] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), Article 15, 2009.

[27] E. Chávez, G. Navarro, R. A. Baeza-Yates, and J. L. Marroqun. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[28] K. Das, J. Schneider, and D. B. Neill. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–176, 2008.

[29] D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of the International Conference on Intelligent Systems*, pages 82–87, 1996.

[30] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[31] T. Fawcett and F. J. Provost. Adaptive fraud detection. *Journal of Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[32] T. Fawcett and F. J. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, 1999.

[33] N. C. for Education Statistics. Status and Trends in the Education of Racial and Ethnic Minorities. `http://nces.ed.gov/pubs2007/minoritytrends/ind_1_1.asp`, 2007.

[34] J. Francis, D. Addison, S. Wermter, and J. MacIntyre. Effectiveness of feature extraction in neural network architectures for novelty detection. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, pages 976–981, 1999.

[35] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[36] J. Gao and P.-N. Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 212–221, 2006.

[37] A. Ghoting, S. Parthasarathy, and M. E. Otey. Fast mining of distance-based outliers in high dimensional datasets. In *Proceedings of the 6th SIAM International Conference on Data Mining*, 2006.

[38] F. E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[39] 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*, 15(3):515–528, 2003.

[40] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[41] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. In *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180, 2002.

[42] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[43] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[44] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[45] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[46] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 293–298, 2001.

[47] G. H. John. Robust decision trees: Removing outliers from databases. In *Proceedings of the 1st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 174–179, 1995.

[48] M. V. Joshi, R. C. Agarwal, and V. Kumar. Mining needle in a haystack: Classifying rare classes via two-phase rule induction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 91–102, 2001.

[49] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases*, pages 392–403, 1998.

[50] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 211–222, 1999.

[51] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *International Journal on Very Large Data Bases*, 8(3-4):237–253, 2000.

[52] E. B. Kong and T. G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning*, pages 313–321, 1995.

[53] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 444–452, 2008.

[54] J. Laurikkala, M. Juhola, and E. Kentala. Informal identification of outliers in medical data. In *Proceedings of the Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pages 20–24, 2000.

[55] A. Lazarevic, L. Ertz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003.

[56] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 157–166, 2005.

[57] W. Lee, S. J. Stolfo, and K. W. Mok. Mining audit data to build intrusion detection models. In *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 66–72, 1998.

[58] M. Markou and M. Singh. Novelty detection: a review. *Signal Processing*, 83(12):2481 – 2497, 2003.

[59] P. McBurney and Y. Ohsawa, editors. *Chance Discovery.* Advanced Information Processing. Springer, 2003.

[60] B. Morin and H. Debar. Correlation of intrusion symptoms: An application of chronicles. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, pages 94–112, 2003.

[61] E. Müller, I. Assent, U. Steinhausen, and T. Seidl. Outrank: ranking outliers in high dimensional data. In *Proceedings of the 24th IEEE International Conference on Data Engineering Workshops*, pages 600–603, 2008.

[62] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 144–155, 1994.

[63] H. V. Nguyen and V. Gopalkrishnan. Efficient pruning schemes for distance-based outlier detection. In *Proceedings of the 13th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 160–175, 2009.

[64] H. V. Nguyen and V. Gopalkrishnan. Feature extraction for outlier detection in high-dimensional spaces. In *Proceedings of the 4th International Workshop on Feature Selection in Data Mining*, pages 64–73, 2010.

[65] H. V. Nguyen, V. Gopalkrishnan, and P. Namburi. Online outlier detection based on relative neighbourhood dissimilarity. In *Proceedings of the 9th International Conference on Web Information Systems Engineering*, pages 50–61, 2008.

[66] M. E. Otey, A. Ghoting, and S. Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Journal of Data Mining and Knowledge Discovery*, 12(2-3):203–228, 2006.

[67] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proceedings of the 19th IEEE International Conference on Data Engineering*, pages 315–324, 2003.

[68] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 427–438, 2000.

[69] S. J. Roberts. Novelty detection using extreme value statistics. *IEE Proceedings on Vision, Image and Signal Processing*, 146(3):124–129, 1998.

[70] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection, 3rd edn.* John Wiley and Sons, 1996.

[71] H. Sagan. *Space Filling Curves.* Springer-Verlag, 1994.

[72] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A wavelet based clustering approach for spatial data in very large databases. *International Journal on Very Large Data Bases*, 8(3-4):289–304, 2000.

[73] D. B. Skalak and E. L. Rissland. Inductive learning in a mixed paradigm setting. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 840–847, 1990.

[74] A. Strehl and J. Ghosh. Cluster ensembles a knowledge reuse framework for combining partitionings. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 93–102, 2002.

[75] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.

[76] E. Suzuki and J. M. Zytkow. Unified algorithm for undirected discovery of exception rules. In *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 169–180, 2000.

[77] J. Tang, Z. Chen, A. W. C. Fu, and D. Cheung. A robust outlier detection scheme in large data sets. In *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1–8, 2002.

[78] Y. Tao, X. Xiao, and S. Zhou. Mining distance-based outliers from large databases in any metric space. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 394–403, 2006.

[79] P. H. S. Torr and D. W. Murray. Outlier detection and motion segmentation. In *Proceedings of 6th SPIE Conference on Sensor Fusion*, pages 432–443, 1993.

[80] W. Wang, J. Yang, and R. R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, 1997.

[81] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Journal of Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.