Data Pruning Based Outlier Detection

Rajendra Pamula

Data Pruning Based Outlier Detection

Thesis submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

by

Rajendra Pamula

Under the supervision of

Prof. Jatindra Kumar Deka

&

Prof. Sukumar Nandi

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
Guwahati 781039, India
April, 2015

Dedicated to
my wife,
my son,
my parents
and
my sisters

Declaration

I certify that

a. The work contained in this thesis is original and has been done by myself under the general supervision of my supervisors.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

d. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT Guwahati

Date:

Rajendra Pamula
Research Scholar
Department of Computer Science
and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, INDIA 781039

Certificate

This is to certify that the thesis entitled **Data Pruning Based Outlier Detection** being submitted by **Rajendra Pamula** to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, is a record of bonafide research work under our supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

Place: IIT Guwahati
Date:

Prof. Sukumar Nandi
Department of Computer Science
and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039

Prof. Jatindra Kr. Deka
Department of Computer Science
and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039

Acknowledgements

I would like to express my sincere gratitude to my supervisors Prof. Jatindra Kr Deka and Prof. Sukumar Nandi for guiding my research work and providing me valuable support. They enabled me to develop a keen interest in understanding the field outlier detection and data mining and gave me much encouragement in carrying out my research work. The numerous discussions I had with them contributed towards learning my research topic and the continuous support and cooperation I recieved is quite exemplry. I shall be indebted to both of them now and always.

I thank all the members of the doctoral committee Prof. Sajit Gopalan, Prof. Diganta Goswami and Prof. Ratnajit Bhattacharya for their valuable comments and suggestions to improve the quality of the work and their critical comments towards refinement of my work.

Next, I would like to thank my host organization, Indian School of Mines for allowing me to carry out my research work at IIT Guwahati by granting me leave of absence and I thank all my colleagues for their continuous support.

My sincere gratitude goes to all the staff members of the Department of Computer Science and Engineering especially Bhrigu, Nanu and Raktajit for helping me out on many occasions.

I would like to express my heartfelt thanks to Bidyut, with whom I shared office space in the department and discussed many more apart from studies, I shall be indebted for the companionship of my fellow research scholars Tejmani sir, Nitya sir, Suresh, Satya, Pallav, Mohanty, Biswanath, Kataniah, Mamta, Godfrey, Hitesh, Agile, Suresh, Maushumi madam, Amrita madam, Neminath, Arup sir, Lipika, Paromita, Aurelia, Alka, Suddhasil, Ashok sir, Shiva, Saitanya, Subhas, Reddy, Nani, Santhi Raju, Nagesh, Ravi, Ramesh, Sekhar, Tharun with whom I have made friendship in my research timespan. They have helped me widen my perspective in life and lifted up my spirits on

many occasions. I shall always remember our numerous tea sessions where we brainstormed on a varied issues/topics.

I want to thank Prof. Roy Paily and his family, Sanjog brother and his family Dr. Prakash and his family, Hokip sir and his family and Bhagath and his family for wonderful fellowship and prayers.

I also want to thank Anjana aunty and his family for their love and care.

To my family and relatives, I have no words to express my gratitude for your continuous support, love and prayers. I offer my deep heartfelt gratitude to my dear wife Anitha and my son Eion for their love and support. I also want to offer deep heartfelt gratitude and respect to my dear father Bapuji and mother Devaki Devi for always encouraging me to think big and helping me to realize my dreams; I also want to express my deep gratitude to my sisters Sunita and Vazrani for their moral encouragement during the tenure of my research work. I also want to express my gratitude for my dear onces Munnu, Kinnu, Pintu and Tippu for their love.

Place: IIT Guwahati

Date: **Rajendra Pamula**

Abstract

Due to the advancement of the data storage and processing capabilities of computers, most of the real life applications are shifted to digital domains and many of them are data intensive. In general, most of the applications deal with similar type of data items, but due to variety of reasons some data points are present in the data set which are deviating from the normal behaviors of common data points. Such type of data points are referred as outliers and in general the number of outliers in a data set is less in number. Identifying the outliers from a reasonably big data set is a challenging task. Several methods have been proposed in the literature to identify the outliers, but most of the methods are computation intensive. Due to the diverse nature of data sets, a particular outlier detection method may not be effective for all types of data set.

The main focus of this work is to develop algorithms for outlier detection with an emphasis to reduce the number of computations. The number of computations can be reduced if the data set is reduced by removing some data points which are obviously not outliers. The number of computations again depends on the number of attributes of data points. While detecting outliers it may be possible to work with less number of attributes by considering only one attributes from a set of similar or correlated attributes.

The objective of this work is to reduce the number of computations while detecting outliers and study the suitability of the method for a particular class of data set. Our methods are based on the clustering techniques and divide the whole data set into several clusters at the beginning. Depending on the nature of the clusters we propose methods to reduce the size of the data sets, and then apply outliers detection method to find the outliers.

We propose three methods based on the characteristics of the clusters to identify the clusters that may not contain outliers and such clusters are pruned from the data set.

We also propose a method to identify the inlier points from each cluster and prune those points from clusters. We use the principle of data summarization and propose a method that involves both cluster pruning and point pruning. For high dimensional data set, we propose a method that involves attributes pruning to reduce the number of computations while detecting outliers.

Once we perform the pruning step, a reduced data set is resulted and then outlier detection techniques are used to detect the outliers. For each method we demonstrate the effectiveness of our proposed methods by performing experiments.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Knowledge extraction from data is commonly referred to as data mining. Extractions of knowledge vary from application to application. Outlier detection in data mining is the identification of items, events or observations/patterns which do not conform to an expected behavior. An outlier is a data point which is significantly different from remaining data. Outliers arise due to mechanical faults, changes in system behavior, fraudulent behavior, human error, instrument error. Their detection is important before they escalate with potentially catastrophic consequences. Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. Thus, finding the outliers and analysing its behaviour is an interesting data mining task, which is referred as outlier detection [2]. Indeed, for many applications, the rare events (outliers) are often more interesting than the normal ones (inliers). Although rare events are by definition infrequent and their significance is high as compared to other events; so the detection of outliers is becoming important. Outlier detection can identify errors and can be removed from the data set makes data set pure for processing. Outlier detection is related to, but distinct from noise. Noise can be defined as a phenomenon in data that is not of interest to the analyst, but acts as a hindrance for data analysis.

1.1 Outlier Detection

Outlier is a point that deviates from other points and outlier detection is a technique to discover the outliers. Most data mining methods discard outliers as noise or exception. However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones. Many outlier detection methods have been emerged/developed for different areas/applications. These methods can be

classified from different perspectives. One important view on which outlier detection can be classified is based on the distance among the data points. Detecting outliers from various applications using different techniques is a non-trivial task. A simple and straight forward method to detect an outlier is to identify the data points which do not belong to normal region. Depending on the application domain, outliers are of specific interest. Outlier detection methods always use some parameters based on the characteristics of data points to define the 'degree of outlierness'. Many techniques have been employed for detecting outliers and in literature it is found that several terminologies have been used by different authors. For example, Chandola *et. al.* [3] described their various approaches as outlier detection. Chandola *et. al.* [3] used the term anomalies and authors of the papers [4–6] used the term novelty detection. Similarly the term discordant observations [3, 7], exceptions [3, 8, 9], surprises [3], peculiarities [3, 10] are the other terms used by different authors in their works. On the other-hand, there is no single formal definition for an outlier, but conceptually:

> An outlier is an observation (or a set of observations) that differs so much from other observations (or sets) as to arouse suspicion that it has been generated by a different mechanism. [11]

Barnett and Lewis [12] indicate that an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs. Similarly, Johnson [13] defines an outlier as an observation in a data set which appears to be inconsistent with the remainder of that set of data

Formally outlier detection can be described as: Given a set of N data points or data objects and m the expected number of outliers, find the top-m objects in reference to some dissimilarity measures that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data points.

In order to understand the need of outlier detection, we need to understand different aspects of outlier detection. For example, as to formulate specific problem, which has to be determined by all factors such as nature of data, description/information about the data, application domain requirements and to describe data in terms of global and local outliers. The global outliers are the observations with the largest distances, whereas an observation that deviates greatly from its neighbors with respect to its local density is considered to be a local outlier. The density is measured by the length of the κ-nearest neighbor distances of its neighbors. Even though a local outlier may not deviate from all the other observations, it, indeed, may signal interesting information. Data visualization

techniques are one way for detecting outliers. Human eyes are very fast and effective at noticing data inconsistencies. However, data visualization methods are weak in detecting outliers in data with many categorical attributes or in data of high dimensionality, since human eyes are good at visualizing numeric data of only two to three dimensions.

There are several factors that affect the performance or utility of the outlier detection methods. Some of them are listed below:

Nature of input data: The nature of input data [3, 14] is the key aspect of any outlier detection technique. The input data is described in terms of set of data points(also referred as objects, records, instances, vectors, patterns, events, cases, samples, observations, entities) and each data point can be described using a set of attributes (also referred to as variables, characteristics, features, fields, or dimensions). The attributes can be of different types such as binary, categorical, or continuous. Each data instance might consist of only one attribute (uni-variate) or multiple attributes (multivariate). In case of multivariate data instances, all attributes might be of same type or might be a mixture of different data types. The nature of the data and the nature of the attributes determine the applicability of the outlier detection techniques.

Type of Outliers: The nature of the desired outliers (output) is also the key aspect of any outlier detection technique. Outliers can be classified into following three categories [3]:

- Point Outliers: If an individual data point is deviating from the rest of the data points, then the data point is termed as point outlier.
- Contextual Outliers: If a data point is observed to be an outlier in a specific context, but not otherwise, then it is termed contextual outlier (also referred to as conditional outliers [15]).
- Collective Outliers: If a set of related data points appears together and behave as outliers with respect to the entire data set, it is termed a collective outliers.

Output: Output is the way to present/describe the nature of the result for any outlier detection technique. In general there are two different ways to present the output:

- Outlier Score: A score is assigned to each data point, which is the degree of outlierness of a data point.

- Labels: A binary classification technique is used to present whether a data point is normal or outlier.

In majority of the applications, outliers are present in the data set due to variety of reasons. The major challenges for outlier detection can be attributed to the following points.

- It is very difficult to indicate the clear boundary between the outliers and inliers.
- Different application domains assume different perspective to distinguish an outlier from the inliers.
- It is very difficult to predict how many number of outliers are present in a data set.
- The task of defining the action of a normal points from abnormal points is very difficult.

Over the time, a variety of outlier detection techniques have been emerged from research communities. Many of these techniques are specific to certain application domains, while others are more generic. Methods for outlier detection can be categorized into following approaches [3].

Statistical outlier detection techniques: These techniques generally take the assumption that outliers lie in the low probability regions and normal point lie in high probability regions of a stochastic model [12,16]. A statistical model is fitted to a given data and a test is applied on the data point to find out whether it belongs to the model are not. The data points with low probability obtained from the model, based on the applied test statistics, are declared as outliers [11, 17]. There are two basic types of procedures for detecting outliers: (1) Block procedures: In this case, either all of the suspect objects are treated as outliers or all of them are accepted as consistent. (2) Consecutive (or sequential) procedures: An example of such a procedure is the inside-out procedure. Its main idea is to consider an object to be tested first which is least likely to be an outlier. If it is found to be an outlier, then all of the more extreme values are also considered as outliers; otherwise, the next most extreme object is tested, and so on. This procedure tends to be more effective than block procedures.

Nearest neighbor based outlier detection techniques: The distance or similarity measure between two data points is used in nearest neighbor based outlier detection.

The distance (or similarity) between two data points can be computed for different types of data in different ways. These methods identify the data points as outliers based on two different aspects; (1) distance between the neighbors and (2) density of the neighbors. From the literature nearest neighbors based outlier detection techniques are broadly classified into two categories namely, distance based techniques, use the distance of the data point to its κ-nearest neighbors as the outlier score [1, 18–23] and density based techniques, use relative density of the data point as outlier score [23, 24].

Classification based outlier detection techniques: These techniques use classification algorithms to build a model to test whether a data point is an outlier or not. Learning a model from a set of training data and classifying a test data point using the learnt model is called classification [3, 14, 25]. Classification based outlier detection techniques can be grouped into two categories namely one-class outlier detection techniques and multi-class outlier detection techniques. In One-class techniques all the training data points have only one class label which are the normal points. If the classifier does not classify the data point then it is an outlier. In multi-class techniques normal points belong to any one of the class where as outlier belong to none of the class.

Clustering based outlier detection techniques: These techniques generally take two important assumptions:

1. Normal data points lie close to their closest cluster centriod, while the outliers lie far away from their closest cluster centriod.
2. Normal data points belong to large and dense clusters, while outliers lie in small or sparse clusters.

Applying an outlier score on the data point to find out the degree of outlier. Most of the outlier detection methods use some quantitative parameters to identify a data point as outlier. These measures are either distance based or density based. The outlier score of a data point is measured as its distance to its k-th nearest neighbor [26] or KNN-distance [27]. If the distance measure is more then the point is an outlier. Knorr *et. al.* [1] use the outlier score as the number of neighbors that are not more than d distant away from the data point and if the number of neighbors is small then the point is an outlier. Similarly, Local Distance Based Outlier Factor ($ldof$) [21], Local Outlier Factor

(LOF) [23], Outlier Detection Using In degree Number ($ODIN$) [28], Multi granularity Deviation Factor ($MDEF$) [9], etc. are some of the measures for outlier score.

Clustering and classification are both fundamental tasks in Data Mining. Classification is used mostly as a supervised learning method, clustering for unsupervised learning. The goal of clustering is descriptive and goal of classification is predictive [29]. The objective of clustering is to find a set of groups which is driven by the intrinsic properties of data points, but in classification, the grouping of data points are driven by some extrinsic properties.

Clustering methods can be divided into different categories, like partitioning methods, hierarchical methods, density based methods, grid based methods, fuzzy and model based methods. Similarly classification methods can be divided into different categories, like parametric and non-parametric. Parametric methods: Linear discriminant analysis, Quadratic discriminant analysis, Maximum entropy classifier. Non-parametric methods: Decision trees, Kernel estimation and k-nearest-neighbor algorithms, Naive Bayes classifier, Neural networks, Support Vector Machines, and Gene expression programming.

In our proposed methods of outlier detection, we perform pre-processing on given data set by partitioning it into several clusters. We used simple and most popular clustering algorithm (k-means) as a initial pre-processing for outlier detection. k-means is a distance based partitional clustering algorithm. It is popular due to the following reasons:

- Its time complexity is $O(i * k * N)$, where N is the number of patterns, k is the number of clusters, and i is the number of iterations taken by the algorithm to converge. Typically, k and i are fixed in advance and so the algorithm has linear time complexity in the size of the data set [30].

- It is order-independent; that means for the same set of initial seed points, it generates the same clusters, irrespective of the order of data points presented to the algorithm. However, the k-means algorithm is sensitive to initial seed selection and even in the best case, it can produce only hyper-spherical clusters.

The key advantage of distance based techniques is that they are purely data driven and do not make any assumptions. The key disadvantage of distance based technique is that it greatly relies on a distance measure defined between a pair of data points. So it involves lot of distance computations. In order to reduce the distance computations we have used clustering to group the data points. Generally the chances of having outliers in dense clusters are less and removing those clusters may not remove any outliers. Our outlier detection methods identify the outliers from remaining points. Several outliers

scores are reported in the literature to identify the outliers. The outlier score local distance based outlier factor ($ldof$) [21] is a good measure for scattered real-world data sets. After removing some of the clusters from given data set, the remaining data set becomes sparse one, and so $ldof$ is used in our methods as outlier score.

1.2 Objective of the thesis

Several methods for outlier detection have been published by different researchers. The basic ideas of all these methods are to calculate an outlier score for data points. Depending on the value of the outlier score, declare the data point as outlier or inlier. The calculation of the outlier score is computation intensive. The main objective of this work is to reduce the number of computations. The reduction of the number of computations can be achieved by reducing the number of data points. In our work, we divide the entire data set into several clusters using k-means clustering algorithm. Depending on the behaviors of clusters it is possible to prune an entire cluster or prune some points from a cluster. Also the computation can be reduced by selecting some features of a data set. As the number of points in a data set can be reduced by pruning, the outlier score calculations for all remaining data points in terms of distance computations are reduced. To reduce the computations while detecting outliers we propose three different ways of pruning data points (for which chances of being an outlier is less) from the data set. Also we put forward another method of features pruning to reduce the number of computations.

- **Cluster Pruning:** Removing a cluster of points is called cluster pruning. A clustering algorithm is used to obtain the clusters. These clusters are analyzed based on different parameters, and identify the clusters that can be pruned. A complete set of data points in the cluster is pruned.
- **Point Pruning:** Removing data points is called point pruning. Clusters are found using clustering algorithm and the points are pruned from each of the clusters with respect to the distance of a point from the centriod of the cluster.
- **Data Summarization based Pruning:** Summarization of a cluster and removing the points based on summarization parameters is called data summarization based pruning. Clusters are formed using clustering algorithm and obtain the features for each cluster. These features are used to select the clusters for pruning.
- **Feature Pruning:** Removing features with similar characteristics is called feature pruning. Correlation factor is calculated for all pairs of features. The features are

pruned which are having similar correlation values. The unpruned distinct features are used for outlier detection.

1.3 Contributions of the Thesis

In this work, we propose three cluster based pruning methods, one point pruning, one data summarization based pruning and one feature pruning method to reduce the size of the data set and eventually reduce the number of computation while detecting outliers. We use the parameter $ldof$ [21] in our work to determine the outlierness of a data point. The contributions of our work are summarized as follows:

Cluster based pruning for Outlier Detection: Three new cluster based pruning methods are proposed in our work. All the methods have three stages. In first stage k-means clustering method is applied to a given data set to obtain k number of clusters. In the second stage, which is termed as pruning stage, the clusters are analyzed so as to prune some clusters. In the third stage an outlier score is calculated for the unpruned data points to report the top-m outliers.

In the first method, outlier score of the centriods of the clusters are calculated. Using these outliers scores, we obtain the clusters that are pruned from data set. The basic assumption for this method is that, the clusters with low outlier score have less chances of containing outliers. Removing certain number of clusters reduces the size of the data set. In the first method we have reduced the data points by pruning half of the clusters. In second and third methods we use some characteristics of the clusters to identify the clusters to be pruned. In the second method we use the radius of each cluster to take the decision about cluster pruning. In the third method we calculate the distance of each cluster's centriod from the overall centriod of the data set and use this distance to find the candidate clusters to be pruned. From the limited number of points remain after pruning, we calculate the outlier score of these points and report the top-m points as outliers from the data set. The results produce by these methods are analyzed experimentally. Proposed methods perform better than the existing method [21].

Point Pruning based Outlier Detection Method: A new point pruning method is proposed to perform better pruning than the cluster pruning methods. In cluster pruning entire cluster is pruned, whereas in point pruning we identify some data points from each clusters that to be pruned. We form the clusters using k-means

clustering algorithm. The radius of the clusters are used in this method to decide whether a data point to be pruned or not. The radius is the average distance of all the data points from the centriod of the cluster. Based on the distance of a point from the centriod, we check for the points which are inliers, that are the points which lie close to the centriod and fall below the radius of the cluster. Pruning the inliers points from the clusters reduces the data points and eventually brings down the distance computations. For the remaining unpruned points we calculate the outlier score and report the top-m points as outliers. We present the experimental results for showing the efficiency of this scheme.

Effective data summarization based pruning for Outlier Detection: A new data summarization scheme is proposed to prune both the clusters and the data points depending on the parameters obtained from the clustering method. Initially we cluster the data set using a leaders clustering method. A leader with summarized information of its followers is referred as *Data Entity*. The distance between the leader and centriod of the cluster is used to take the decision whether the cluster has to be pruned are not. If a cluster is not getting pruned we take the distance of the data points from the leader to check whether the point is a candidate outlier are not and accordingly we prune some data points. For all the pruned clusters, we include a representative data point for each of such pruned clusters in the remaining data set. After pruning the clusters and points from the clusters, we calculate the outlier score for the rest of the points and report the top-m as outliers depending on their outlier scores.

Correlation based Pruning Method for Outlier Detection: A new correlation based attribute pruning method is proposed to identify the attributes to be pruned. A well known Pearson Correlation Coefficient is used to find the correlation value between the attributes. The attributes which are having very high correlation values are grouped together. Only one attribute is selected from each group, remaining attributes are pruned from the group. All features selected from different groups are used for outlier detection. An outlier score is calculated for all the points with only selected attributes. Due to the reduction in the number of attributes, the time required for distance computation is also reduced. Finally we report the top-m points with outlier score as outliers.

1.4 Organization of the Thesis

The thesis is organized into seven chapters. A summary of the contents of the chapters is as follows:

Chapter 2: This chapter presents a brief survey and related concepts of previous work on outlier detection methods that forms the background of our work.

Chapter 3: This chapter introduces the idea of pruning to reduce the computations while detecting outliers and presents three new cluster pruning methods for outlier detection.

Chapter 4: This chapter addresses the problems of cluster pruning and a new point pruning method is introduced to prune the data points from different clusters while detecting outliers.

Chapter 5: In this chapter we define a new method for data summarization called *Data Entity*. Using the data entity, we propose a new outlier detection method that involves both cluster pruning and data pruning.

Chapter 6: In this chapter we propose a feature correlation based pruning method, which can identify the correlated features. An outlier detection method is proposed using only one feature among the correlated features.

Chapter 7: This chapter summarizes the overall contributions of the thesis with several interesting directions for future research.

Chapter 2

Background and Related Works

2.1 Introduction

It is observed that in many applications there exist data points that do not comply with the general behavior or model of the data. Such data points, which are inconsistent with the remaining set of data, are called outliers [3, 31]. Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. This, however, could result in the loss of important hidden information. In other words, the outliers may be of particular interest, such as in the case of fraud detection. Hence forth, outlier detection and analysis is an interesting data mining task, referred to as outlier detection. Outlier detection can be described as follows [3, 21, 31]: For a given set of N data points, finding the top-m (m expected number of outliers) objects that are considerably dissimilar, exceptional or inconsistent with respect to the remaining data points is treated as outlier detection. The outlier detection problem can be viewed as two sub-problems:

- Defining the objects/data points that are inconsistent in a given data set.
- Defining efficient algorithms to detect outliers.

The problem of defining outliers is non-trivial. When multidimensional data are analyzed, not any particular one dimension but a combination of dimension values may be extreme. For non-numeric (i.e., categorical) data, the definition of outliers requires special consideration. Similarly data visualization methods are weak in detecting outliers in data with many categorical attributes or in data of high dimensionality, since human eyes are not good at visualizing numeric data beyond two to three dimensions. Even clustering algorithms may not identify outliers [32]. Clustering finds groups of strongly

related objects. Outlier detection finds objects that are not strongly related to other objects. Some clustering algorithms discard outliers as noise. The clustering algorithms can be modified to include outlier detection as a by-product of their execution. An object is a cluster-based outlier if the object does not belong strongly to any cluster. In detecting outliers, small clusters that are far from other clusters are considered to contain outliers. These approaches are sensitive to the number of clusters selected. It requires thresholds for the minimum cluster size and the distance between a small cluster (with outliers) and other clusters. If a cluster is smaller than the minimum size, it is regarded as a cluster of outliers. Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their similarity. As a data mining function, cluster analysis can be used as a stand-alone tool to gain insight into the distribution of data, to observe the characteristics of each cluster, and to focus on a particular set of clusters for further analysis. Alternatively, it may serve as a preprocessing step for other algorithms, such as characterization, outlier detection, attribute subset selection, and classification, which would then operate on the detected clusters and the selected attributes or features. Many algorithms are designed to cluster interval-based (numerical) data. However, applications may require clustering other types of data, such as binary [33, 34], categorical [35, 36], and ordinal data [34], or mixtures of these data types. So clustering algorithms are used as a preprocessing step to obtain certain number of clusters. These clusters are examined to use them for further analysis. For example, the clusters are used to compress the data based on the data summarization. But to form these clusters we need a measure to group the points into clusters. That is the similarity or dissimilarity measure. In the next section we discuss various dissimilarity measures for the different types of attributes.

2.2 Dissimilarity Measures

The types of data that often occur in cluster analysis and how to preprocess them for such an analysis. Suppose a data set to be clustered contains N objects, which may represent persons, houses, documents, countries, and so on. Many of the clustering algorithms typically operate on either of the following two data structures.

Data matrix (or object-by-variable structure) [2,37]: This represents N objects, such as persons, with δ variables (also called measurements or attributes), such as age, height, weight, gender, and so on. The structure is in the form of a relational table, or N-by-δ matrix (N objects $\times$ δ variables).

Table 2.1: A contingency table for binary variables.

		Object j		
		1	0	sum
Object i	1	q	r	$q+r$
	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

Dissimilarity matrix (or object-by-object structure) [2,38,39]: This stores a collection of proximities that are available for all pairs of N objects. It is often represented by an N-by-N table, where $d(i,j)$ is the proximities between the objects i and j and measure the difference or dissimilarity between objects i and j. In general, $d(i,j)$ is a non negative number that is close to 0 when objects i and j are highly similar or near each other. Similarly dissimilarity becomes higher as they differ more or move far from each other. Many clustering algorithms operate on a dissimilarity matrix. If the data are presented in the form of a data matrix, it can first be transformed into a dissimilarity matrix before applying such clustering algorithms.

Calculating the dissimilarity between different variables like interval-scaled variables, binary variables, categorical, ordinal, and ratio-scaled variables; or combinations of these variables are different.

- A binary variable has only two states 0 or 1, where 1 represent presence of variable and 0 represent absence of the variable. The results of any method can mislead if the binary variables are treated as interval-scaled variables. Therefore, methods specific to binary data are necessary for computing dissimilarities. There are two types of binary variables, symmetric and asymmetric binary variables. A symmetric binary variable is a variable that has equal valuable states and carry the same weight; that is, there is no preference on which outcome should be coded as 0 or 1. Dissimilarity measure that is based on symmetric binary variables is called symmetric binary dissimilarity. If all binary variables are thought of as having the same weight, the contingency table is viewed as 2-by-2 matrix as shown in Table 2.1, where q is the number of variables that equals 1 for both objects i and j, r is the number of variables that equals 1 for object i but that are 0 for object j, s is the number of variables that equals 0 for object i but equal 1 for object j, and t is the number of variables that equal 0 for both objects i and j. The total number of variables is p, where $p = q+r+s+t$. The dissimilarity (or distance) measure for a symmetric binary

variables can be assessed between objects i and j as shown in Equation (2.1).

$$d(i,j) = \frac{r+s}{q+r+s+t} \tag{2.1}$$

A binary variable is asymmetric if the outcomes of the states are not equally important, such as the positive and negative outcomes of a disease test. Generally, the most important outcome, which is usually the rarest one, is coded by 1 (e.g., Cancer positive) and the other by 0 (e.g., Cancer negative). Given two asymmetric binary variables, a positive match (code 1) is considered more significant than that of a negative match (code 0). The dissimilarity based on such variables is called asymmetric binary dissimilarity, where the number of negative matches, t, is considered unimportant and thus is ignored in the computation, as shown in Equation (2.2).

$$d(i,j) = \frac{r+s}{q+r+s} \tag{2.2}$$

Complementarily, we can measure the distance between two binary variables based on the notion of similarity instead of dissimilarity. For example, the asymmetric binary similarity between the objects i and j, or $sim(i,j)$, can be computed as shown in Equation (2.3)

$$sim(i,j) = \frac{q}{q+r+s} = 1 - d(i,j) \tag{2.3}$$

The coefficient $sim(i,j)$ is called the Jaccard coefficient [40].

- A categorical variable is a generalization of the binary variable in that it can take on more than two states. For example, map color is a categorical variable that may have, say, five states: red, yellow, green, pink, and blue. Let the number of states of a categorical variable be s_t. The states can be denoted by letters, symbols, or a set of integers, such as 1, 2, . . . , s_t. These integers are used just for data handling and do not represent any specific ordering. The dissimilarity between two categorical objects i and j can be computed based on the ratio of mismatches. Where s_t is the number of matches (i.e., the number of variables for which i and j are having the same state), and p is the total number of variables. Weights can be assigned to increase the effect of s_t or to assign greater weight to the matches in variables having

a larger number of states. Equation (2.4) shows the dissimilarity measure between the objects i and j.

$$d(i,j) = \frac{p - s_t}{p} \tag{2.4}$$

- A discrete ordinal variable resembles a categorical variable, except that the s_t states of the ordinal value are ordered in a meaningful sequence. A continuous ordinal variable looks like a set of continuous data of an unknown scale. Ordinal variables may also be obtained from the discretization of interval-scaled quantities by splitting the value range into a finite number of classes. The values of an ordinal variable can be mapped to ranks. The treatment of ordinal variables is quite similar to that of interval-scaled variables when computing the dissimilarity between objects. Suppose that t is a variable has s_t states from a set of ordinal variables describing N objects. The dissimilarity computation with respect to t involves the following steps:

  1. The value of t for the i^{th} object is x_{it} , and t has s_t ordered states, representing the ranking 1, ... ,s_t . Each x_{it} is replaced by its corresponding rank, $r_{it} \in \{1, ...s_t\}$.
  2. Since each ordinal variable can have a different number of states, it is often necessary to map the range of each variable onto [0.0, 1.0] so that each variable has equal weight. This can be achieved by replacing the rank r_{it} of the i^{th} object in the t^{th} variable by

$$z_{it} = \frac{r_{it} - 1}{s_t - 1} \tag{2.5}$$

  3. Dissimilarity can then be computed using any of the distance measures as described in the interval scaled variables.

- A ratio-scaled variable makes a positive measurement on a nonlinear scale, such as an exponential scale, approximately following the formula

$$Ae^{Bq} \text{ or } Ae^{-Bq} \tag{2.6}$$

where A and B are positive constants, and q typically represents time. Common examples include the growth of a bacteria population or the decay of a radioactive element. There are three methods to handle ratio-scaled variables for computing the dissimilarity between objects

1. Treat ratio-scaled variables like interval-scaled variables. This, however, is not usually a good choice since it is likely that the scale may be distorted.
2. Logarithmic transformation is applied to a ratio-scaled variable t having value x_{it} for object i by using the formula $y_{it} = log(x_{it})$. The y_{it} values can be treated as interval- valued.
3. Treat x_{it} as continuous ordinal data and treat their ranks as interval-valued. The latter two methods are the most effective, although the choice of method used may depend on the given application.

- Interval-scaled variables are continuous measurements of a roughly linear scale. Typical examples include weight and height, latitude and longitude coordinates and weather temperature. The measurement unit used can affect the analysis. For example, changing measurement units from meters to inches for height, or from kilograms to pounds for weight, may lead to a very different results. To help avoid dependence on the choice of measurement units, the data should be standardized. Standardizing measurements attempts to give all variables an equal weight. This is particularly useful when given no prior knowledge of the data. However, in some applications, users may intentionally want to give more weight to a certain set of variables than to others. After standardization, or without standardization in certain applications, the dissimilarity (or similarity) between the objects is typically computed based on the distance between each pair of objects. The most popular distance measure is Euclidean distance. Distance between any two data points is often used as the dissimilarity measure in many methods. Suppose there are three different points a, b, and c. If the distance between (a,b) is higher than the distance between (a,c), then we can say that dissimilarity between (a,b) is higher than dissimilarity between (a,c).

  Euclidean distance [41, 42] is widely used dissimilarity measure by machine learning and data mining communities. Euclidean distance between a pair $d(x, y)$ of δ-dimensional points (objects) can be expressed as follow.

$$d(x, y) = \sqrt{\sum_{i=1}^{\delta} (x_i - y_i)^2} \tag{2.7}$$

  Another well-known dissimilarity measure is Manhattan (or city block) distance, defined as

$$d(x, y) = \sum_{i=1}^{\delta} |x_i - y_i| \tag{2.8}$$

Generalization of Euclidean and Manhattan distance is known as Minkowski distance [41, 42].

$$L_p = \left(\sum_{i=1}^{\delta} |x_i - y_i|^p \right)^{1/p} \tag{2.9}$$

This is referred to as L_p norm. If $p = 1$, we call it as L_1 norm or city block distance. Similarly, Euclidean distance is called L_2 norm. Euclidean distance has some well known properties, which are given below. Let d be the Euclidean distance between a pair of objects in $\mathcal{D}$. Then d satisfies certain properties. For three points $x, y, z \in D$,

- **Non-negativity:** $d(x, y) \geq 0$
- **Reflexivity:** $d(x, y) = 0$, if $x = y$
- **Symmetry:** $d(x, y) = d(y, x)$
- **Triangle inequality:** $d(x, y) + d(y, z) \geq d(x, z)$

A distance function that satisfy above all properties is called *metric* distance [43]. However, Euclidean distance cannot be used when features have different scales and they are correlated among themselves. In this scenario, *Mahalanobis distance* [44] is very useful. It considers distribution of entire data set. Mahalanobis distance between x and y

$$d_M(x, y) = (x - y) \sum^{-1} (x - y)^T \tag{2.10}$$

where $\sum$ is the covariance matrix of the data set. Mahalanobis distance has time complexity of $O(N\delta^2)$, where N is the size of the data set and δ is the dimension of the data set. Therefore, this method is not suitable in large data set. Euclidean distance takes $O(\delta)$ times to compute distance between a pair of objects. Euclidean distance is independent of underlying data set. Outlier detection methods proposed in this thesis use Euclidean distance as the dissimilarity measure.

Based on different similarity/dissimilarity measures many clustering methods have been developed in various application domains, which are discussed in next section.

2.3 Types of Clustering Methods

The essence of many fields such as statistics, object recognition, information retrieval, machine learning, data mining, psychology and other social sciences is to find meaningful groups of objects. Cluster analysis has been played an important role in these fields [45]. Therefore, many clustering methods have been emerged for automatically finding meaningful groups or clusters. It is very difficult to give crisp categorization of clustering methods because methods in one category may share some characteristic from other categories. One can categorize these clustering methods from different perspectives [2, 14, 45, 46] and a brief description is presented for each category.

- **Partitional and Hierarchical:** Clustering methods can be divided on the basis of clustering results *viz., partitional clustering* and *hierarchical clustering* methods. Partitional clustering methods produce a single clustering (flat clustering) while hierarchical clustering methods produce nested clustering of a data set. Most of the partitional clustering methods need the desired number of clusters as the prior knowledge. Hierarchical or taxonomic structures of data are discovered through hierarchical clustering methods.

- **Distance Based and Density Based:** Clustering methods use optimization technique to produce clustering results. Some clustering methods uses global criteria, which is the function of distance between objects in a data set. These methods are called distance based method. On the other hand, density based clustering method optimize local criteria, which is based on density distribution of the data set. In this approach, a dense region is considered as a cluster in the feature space. Both distance and density based methods can produce flat as well as nested clusterings of data.

- **Hard and Soft :** Clustering methods, which assign object to a cluster is called hard clustering. Here, cluster boundary is crisp (hard or rigid) and clusters are well separated. However, object may be required to place in more than one clusters simultaneously in many applications. Here, clusters are overlapped, i.e. some objects belong to more than one clusters. A different type of clustering method is proposed to handle these applications. This is called soft clustering approach.

- **Complete and Partial:** Complete clustering methods assign each and every object to a cluster. Partial clustering could not assign some of the objects to any clusters because these objects do not belong to any meaningful groups in the data.

In the next subsections, we review widely used clustering methods with emphasizing on the methods which can detect clusters in nearly linear time.

2.3.1 k-means clustering

The k-means [47, 48] is a distance based partitional clustering method. The k-means is very popular clustering method and it is used in many application domains. It is an iterative method. The k-means assumes that the desired number of clusters, k is a user specified parameter. Initially, method selects k objects randomly from data set and treats them initial centriods of k clusters. Centriod of a cluster is the center point of a cluster. It needs not be object in the given data set. Each object of the data set is then assigned to the closest centriods. Patterns assigned to a centriod form a new cluster and centriod of each of the k clusters are recalculated. The step of assignments to and calculation of centriod is repeated until all centriods remain unchanged in consecutive iterations.

The k-means has time complexity of $O(i * k * N)$, where i is the number of iterations, N is the size of the data set. Since i is a small number as compared to N and $k << N$, the method runs linearly in the size of the data set. The space requirement of the method is $O(N)$.

However, it has advantages and disadvantages. It is relatively efficient because the time complexity of the method is linear if i, and k are small. It gives best result when data set is distinct or well separated from each other. It can find only convexed shaped clusters. It cannot detect noise point or outliers. It is applicable to only numeric data set (vector space)and with different initial points, it produces different clustering results.

There are many improvements of this basic k-means method [49–51]. k-medoids is a variation of k-means method. Here, a medoid represents a cluster. A medoid is a cluster point, which locates close to center of the cluster. It can be applied to non-numeric data set. However, it is computationally more expensive than the k-means method. Other variation of k-means method are CLARA [52], CLARANS [53].

2.3.2 Leaders clustering

There exists a category of clustering method called sequential algorithm. These methods are fast and create a single clustering of the data set [54]. Leaders clustering method [45, 55–57] is one of this type. It is a distance based clustering method. It scans the data set only once. Leaders clustering method has also been used in preclustering phase in data mining applications [58].

For a given threshold distance τ, it produces a single clustering of a data set $\mathcal{D}$. Each cluster is represented by object called *leader*. It produces a set of leaders $\mathcal{L}$ as follows. First object x_1 in the data set is considered as the first leader in $\mathcal{L}$. For rest of the object $x \in \mathcal{D} \setminus \{x_1\}$, if there is a leader $l \in \mathcal{L}$ such that $||x - l|| <= \tau$, then x is assigned to the leader l. In this case, x is considered as a *follower* of the leader l. If no such leader is found in the leader set, then x is added as a new leader to $\mathcal{L}$. Main advantages of this method is that it scans a data set once, it can be applicable to any data set where notion of distance is defined and it is an incremental algorithm. However, it has the following drawbacks. It can find only convex shaped clusters in data. Leaders method produces different clustering results for two different scanning orders of the same data set,i.e., clustering results are ordered dependent. Similarity (distance) between objects (followers) of different clusters (leaders) may be more (less) than corresponding leaders. Let x and y be two followers of two distinct leaders l_x and l_y, respectively. Distance between two leaders l_x and l_y is always more than τ, however distance between x and l_y or y and l_x or x and y may be less than τ.

The time complexity of leaders clustering is $O(mN)$, where $m = \mid \mathcal{L} \mid$. The space complexity is $O(m)$, if only leaders are stored; otherwise $O(N)$.

Based on these clustering methods, how to reduce the distance computations from the data set in order to find the outlier detection. The reduction of distance computations can be achieved by data pruning. In the next Section, we discuss how data pruning is applied in different areas.

2.4 Data Pruning

In classification based data mining techniques, the general belief for better generalization is to have more training data. But in contrast it is proved that the learning algorithm performs better off when some training data are discarded/pruned/removed. In other words, the quality of the examples are important. Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results. In data reduction, the cluster representations of the data are used to replace the actual data. The effectiveness of this technique depends on the nature of the data. It is much more effective for data that can be organized into distinct clusters than for smeared data. In the literature there are different strategies for data reduction some of them are

described as follows:

1. Numerosity reduction, where the data is reduced by replacing or estimating alternative smaller forms of data representations. In parametric models data are represented by model parameters instead of storing the actual data. Non-parametric methods do not assume any model and data are represented as clustering, sampling, and histograms.

2. Attribute subset selection, where irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.

2.4.1 Numerosity reduction

Techniques of numerosity reduction can indeed be applied to reduce the data volume by choosing alternative, smaller forms of data representation. These techniques may be parametric or non-parametric. For parametric methods, a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data. (Outliers may also be stored.) Log-linear models, which estimate discrete multidimensional probability distributions, are an example. Non-parametric methods for storing reduced representations of the data include histograms, clustering, and sampling.

Clustering techniques consider data objects as objects. They partition the objects into groups or clusters, so that objects within a cluster are similar to one another and dissimilar to objects in other clusters. Similarity is commonly defined in terms of how close the objects are in space, based on a distance function. The quality of a cluster may be represented by its diameter, the maximum distance between any two objects in the cluster.

In data reduction, the cluster representations of the data are used to replace the actual data. The effectiveness of this technique depends on the nature of the data. It is much more effective for data that can be organized into distinct clusters than for smeared data. In database systems, multidimensional index trees are primarily used for providing fast data access. They can also be used for hierarchical data reduction, providing a multi-resolution clustering of the data. This can be used to provide approximate answers to queries. An index tree recursively partitions the multidimensional space for a given set of data objects, with the root node representing the entire space. Such trees are typically balanced, consisting of internal and leaf nodes. Each parent node contains keys and pointers to child nodes that, collectively, represent the space represented by the parent

node. Each leaf node contains pointers to the data objects they represent (or to the actual objects).

An index tree can therefore store aggregate and detail data at varying levels of resolution or abstraction. It provides a hierarchy of clustering of the data set, where each cluster has a label that holds for the data contained in the cluster. If we consider each child of a parent node as a bucket, then an index tree can be considered as a hierarchical histogram. The use of multidimensional index trees as a form of data reduction relies on an ordering of the attribute values in each dimension. Two-dimensional or multidimensional index trees include R-trees, quad-trees, and their variations. They are well suited for handling both sparse and skewed data.

Sampling can be used as a data reduction technique because it allows a large data set to be represented by a much smaller random sample (or subset) of the data. Suppose that a large data set, $\mathcal{D}$, contains N objects. The most common ways that we could sample $\mathcal{D}$ for data reduction, as illustrated as follows:

1. Simple random sample without replacement (SRSWOR) of size s: This is created by drawing s of the N objects from $\mathcal{D}$ ($s < N$), where the probability of drawing any object in $\mathcal{D}$ is $1/N$, that is, all objects are equally likely to be sampled. Simple random sample with replacement (SRSWR) of size s: This is similar to SRSWOR, except that each time a object is drawn from $\mathcal{D}$, it is objected and then replaced. That is, after object is drawn, it is placed back in $\mathcal{D}$ so that it may be drawn again.

2. Cluster sample: If the objects in $\mathcal{D}$ are grouped into M mutually disjoint clusters, then an SRS of s clusters can be obtained, where $s < M$. For example, objects in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduced data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the objects. Other clustering criteria conveying rich semantics can also be explored. For example, in a spatial database, we may choose to define clusters geographically based on how closely different areas are located.

An advantage of sampling for data reduction is that the cost of obtaining a sample is proportional to the size of the sample, s, as opposed to N, the data set size. Hence, sampling complexity is potentially sublinear to the size of the data. Other data reduction techniques can require at least one complete pass through $\mathcal{D}$. For a fixed sample size, sampling complexity increases only linearly as the number of data dimensions, δ, increases.

Data Summarization

Descriptive data summarization techniques can be used to identify the typical properties of data and highlight which data values should be treated as noise or outliers. The basic concepts of descriptive data summarization before getting into the concrete workings of data preprocessing techniques. For many data preprocessing tasks, users would like to learn about data characteristics regarding both central tendency and dispersion of the data. Measures of central tendency include mean, median, mode, and midrange, while measures of data dispersion include quartiles, interquartile range (IQR), and variance. These descriptive statistics are of great help in understanding the distribution of the data. Such measures have been studied extensively in the statistical literature. From the outlier detection point of view, we need to examine how they can be computed efficiently in large databases. In particular, it is necessary to introduce the notions of distributive measure, algebraic measure, and holistic measure. Knowing what kind of measure we are dealing with can help us to choose an efficient implementation for it. The standard deviation is one of such measures for efficient summarization. The standard deviation is the measure of spread in data. It is defined in the Equation (2.11).

$$\sigma = \sqrt{\frac{1}{N}\left[\sum x_i^2 - \frac{1}{N}(\sum x_i)^2\right]} \tag{2.11}$$

The basic properties of the standard deviation, σ, are; (1) it measures spread about the mean. (2) $\sigma = 0$ only when there is no spread, that is, when all observations have the same value.

One way to handle data pruning of a data set is to use data summarization or data compression technique. The main objective of this approach is to compress or summarize the data into a small representative set called summarized set. Then, data pruning techniques can be applied to this set.

Zhang *et. al.* [59] have proposed a method Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is an incremental clustering method. The main idea of the BIRCH clustering method are *Clustering Feature (CF)* and *CF-tree*. The summarized information of a sub-cluster is defined as a CF in triplet. More formally, a CF for a sub-cluster $C_1 = \{\overrightarrow{X_1}, \overrightarrow{X_2}, \ldots \overrightarrow{X_q}\}$ is defined as $CF = (q, \overrightarrow{LS}, ss)$, where $\overrightarrow{LS}$ is linear sum of objects in C_1, i.e., $\overrightarrow{LS} = \sum_i \overrightarrow{X_i}$, ss is square sum of data points, i.e., $ss = \sum_i \overrightarrow{X_i}^2$. Centriod, radius and diameter of sub-cluster C_1 can be computed from the CF values. CF-values satisfy additivity property, i.e. let $CF_1 = (q_1, \overrightarrow{LS_1}, ss_1)$ and

$CF_2 = (q_2, \overrightarrow{LS_2}, ss_2)$ are two CF values of two sub-clusters C_1 and C_2, respectively. Then, CF-value of $C = C_1 \cup C_2$ is $CF(C) = (q_1 + q_2, \overrightarrow{LS_1} + \overrightarrow{LS_2}, ss_1 + ss_2)$. A CF tree, which is a height-balanced tree, is constructed incrementally from CF values of sub-clusters.

DuMouchel *et. al.* [60] have proposed a data summarization technique, which can be used to scale up a set of machine learning and statistical modeling methods. The method is called *data squashing*. Data squashing summarizes a large data set into smaller size data set called *squashed data* with same number of dimensions as the original data set. It has three sequential steps as follow. In the first step, data set is partitioned into a number of compact sub regions or bins. Bins can be constructed by creating hyper-rectangles in the original space or by creating different *layers* in the transformed data space. Let $p = (x_1, \ldots, x_N)$ be a point in the original space. The point p can be transformed into $p' = (y_1, \ldots, y_N)$, where $y_i = \dfrac{C_i - x_i}{\text{standard deviation of } i^{th} \text{ feature}}$, $C_i = i^{th}$ component of center $C = \dfrac{\sum_{\mathcal{D}} p}{|\mathcal{D}|}$ of the whole data set. Layers are created based on the distance from p' to origin. Having created the bins, next step computes moments for the data points falling in a bin. Method computes different order moments like means, minima, maxima, second order moments, third order moments, fourth order moments, etc. in each bin. These moments are sufficient statistics of the points of a bin. Last step is to create a pseudo data point corresponding to a bin from the above calculated statistics. A pseudo point of a bin is the representative of all original data points falling in the bin. They showed that data squashing outperform random sampling approach. These data summarization schemes cannot compute effective distance between a pair of summarized units. Therefore, these schemes cannot be directly applied to hierarchical clustering methods.

First successful data compression technique for hierarchical clustering method is found in [61]. The compressed representation of a set of data points are called *Data Bubble (DB)*, which contains statistical information of the set of points. Data Bubble initially selects a subset of objects randomly which requires one database scan. In next step, it classifies each object to its nearest selected object and updates statistics of the selected object by means of CF. Data Bubble utilizes Clustering Features (CF) introduced in BIRCH clustering method as follows. Let $CF = (k, \overrightarrow{LS}, ss)$ be the Clustering Features of a set of points $X = \{X_i\}$ in N dimensions feature space. A Data Bubble B, which describes X can be expressed as $B = (k, \overrightarrow{M}, e)$, where $\overrightarrow{M} = \dfrac{\overrightarrow{LS}}{k}$ is the center of X and $e = \dfrac{\sqrt{2 * n * ss - 2 * \overrightarrow{LS}^2}}{k * (k-1)}$ is called extent (spread) of X. More specifically, e is the radius of a hyper-sphere centered at M, which covers most of the data points of X. Markus *et.*

al. [61] proposed to apply Ordering Points to Identify the Clustering Structure (OPTICS) method to the data bubbles of $\mathcal{D}$. OPTICS clustering method always finds an unprocessed closest data points from already processed item. However, distance between centers of a pair of Data Bubbles does not reflect the distance between points of the Data Bubbles. A suitable measure is introduced to calculate distance between a pair of Data Bubbles. Finally, OPTICS clustering method is modified to work with Data Bubbles. Modified OPTICS is found to be up to three order faster than classical OPTICS method applied directly to data set with maintaining clustering quality.

Data Bubble [62] A Data Bubble for a set of points $X = \{X_i\} \subseteq \mathcal{D}$ is $B = (rep, k, extent, nnDist)$, where rep is representative object of X, k is the number of objects in X, $extent$ is the radius of hyper-sphere centered at rep which encloses most of the objects in X. and $nnDist(K, B)$ is a function that calculate average κ-nearest neighbor distances in B.

Markus *et. al.* [62] have pointed out problems of applying OPTICS method directly to Clustering Features (CF). Clustering quality deteriorates abruptly with high compression ratio due to three key problems namely *size distortion*, *lost objects* and *structural distortion*. Sizes of clusters are distorted with high compression ratio (number of representative objects to the size of the data set), *i.e.*, some clusters become larger and other become smaller. This is called size distortion problem. OPTICS outputs a *reachability-plot,* which is a two dimensional plot represents hierarchical structures of clusters present in the data set. None of the original objects appears in the reachability plot if CF is directly used in speeding up OPTICS method. This is called lost objects problem. With high compression rate, clustering structures are distorted in the reachability plot, this is called structural distortions. To alleviate all above problems, definition of Data Bubble is redefined as follows.

Bidyut *et. al.* [63] have proposed a scheme to summarize the data set called data sphere (ds). The data sphere (ds) collects sufficient statistics by applying the leaders clustering method twice on the data set. Using $ds = (m, l, \mu + \alpha\sigma, P)$ statistics and selected leaders as the representative points. A Single Link (SL)-clustering method is applied on these points to get better clusters.

Data Sphere(ds), Let $X = x_1, x_2, ..., x_m \subseteq \mathcal{D}$ be the followers of a leader l. A data sphere(ds) for X is defined as a 4-object $ds = (m, l, \mu + \alpha\sigma, P)$, where $m =$ number of followers including the leader l; $\mu = ld/m =$ the average distance from leader l to its followers; $\sigma = \sqrt{\frac{sd}{m} - \mu^2}$ standard deviation of distances from leader to its followers; $ld = \sum_{i=1}^{m} d_i$; $sd = \sum_{i=1}^{m} d_i^2$; $\alpha \in \Re, 0 < \alpha \leq (\tau - \mu)/\sigma$; $P =$ a subset of

followers of l, which lie outside the hyper-sphere of radius; $\mu + \alpha\sigma$ centered at l, i.e., P = $\{p_i \in C \mid\mid\mid l - p_i \mid\mid > \mu + \alpha\sigma\}$

2.4.2 Attribute subset selection

Data sets for analysis may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant. Although it may be possible for a domain expert to pick out some of the useful attributes, this can be a difficult and time-consuming task, especially when behavior of the data is not well known. Leaving out relevant attributes or keeping irrelevant attributes may be detrimental, causing confusion for the algorithm employed. This can result in discovered objects of poor quality. In addition, the added volume of irrelevant or redundant attributes can slow down the detection process. Attribute subset selection reduces the data set size by removing irrelevant or redundant attributes (or dimensions). The goal of attribute subset selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Mining on a reduced set of attributes has an additional benefit. It reduces the number of attributes appearing in the discovered objects, helping to make the objects easier to understand. For δ attributes, there are 2^δ possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as δ and the number of data classes increase. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically greedy, while searching through attribute space, they always make what looks to be the best choice at the time. The strategy is to make a locally optimal choice in such a way that leads to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution. The best (and worst) attributes are typically determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used. Basic heuristic methods of attribute subset selection include the following techniques.

1. Stepwise forward selection: The procedure starts with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set.

2. Stepwise backward elimination: The procedure starts with the full set of attributes. At each step, it removes the worst attribute remaining in the set.

3. Combination of forward selection and backward elimination: The stepwise forward selection and backward elimination methods can be combined so that, at each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.

The stopping criteria for these methods may vary. The procedure may employ a threshold on the measure to determine when to stop the attribute selection process. In the next subsection a correlation based attribute selection is described.

Correlation Coefficient

A correlation based subset selection is a step forward attribute subset selection heuristic method. Correlation describes the relationship between two different factors (variables, features). In statistics community it is called as correlation coefficient. Correlation summarizes the relationship between two variables in a single number called the correlation coefficient. The correlation coefficient is usually given the symbol r that describes direction (positive or negative) and degree (strength) of relationship between two variables. A correlation coefficient can vary from (-1 , +1). The higher the correlation coefficient, the stronger the relationship. A correlation coefficient quite close to 0, but either positive or negative, implies little or no relationship between the two variables. If the correlation value is close to -1 then the variables are called negatively correlated that means one variable increases other decreases. A correlation coefficient can be produced for different types of variables like ordinal, interval or ratio variables. For ordinal attributes, the correlation coefficient which is usually calculated is Spearman's rho. For interval or ratio attributes, the most commonly used correlation coefficient is Pearson's correlation coefficient [64].

The Pearson correlation coefficient r can be defined as follows. Suppose that there are two variables x and y , each having N values x_1 , x_2 , . . . , x_N and y_1 , y_2 , . . . ,y_N respectively. Let the mean of x be $\bar{x}$ and y be $\bar{y}$. Then the Pearson Correlation Coefficient $r(x, y)$ is given by:

$$r(x,y) := \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{2.12}$$

From Equation (2.12), the possible values of r range from -1 to +1. A correlation coefficient r close to -1 indicates that the values of x and y are inversely associated. That is, as x increases y decreases. When r is close to 0, either on the positive or the negative side, then there is little or no association between x and y . So if r is close to 1, then x and y are closely related.

Till now we have introduced, discussed some of the important ways to measure the dissimilarity for different types of data, types of clustering techniques, and to select the data. In the next section we present few well popular definitions for outlier detection techniques.

2.5 Definitions

There are very general and well accepted definitions for outliers in the outlier detection research community.

Definition 2.1. *[11] An outlier is an observation (or a set of observations) that differs so much from other observations (or sets) as to arouse suspicion that it was generated by a different mechanism.*

Definition 2.2. *[12] An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.*

Definition 2.3. *[65] An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.*

Where as some more definitions that are applicable for outlier detection depending on the application domain.

Definition 2.4. *[26] An outlier is defined as a data point which has highest outlier score. The score is the measure of a distance to its κ^{th} nearest neighbor.*

Definition 2.5. *[27] The data points having highest outlier scores are termed as outliers. Outlier score of a data point is measured as the sum of its distances to all its κ nearest neighbors in a given data set.*

Definition 2.6. *[1] An object p in a data set is outlier if it has less then q objects lying within distance r from p.*

From these definitions, it is observed that there are many ways in literature to define an outlier. The fact is that, the problem of defining a unified notion of outliers is nontrivial.

2.6 Outlier Detection Techniques

In the literature outlier detection techniques [3, 6, 66, 67] are classified into four major categories namely classification based, nearest neighbor based, clustering based and

statistical based techniques. However, this thesis is focused on mainly two categories, i.e. nearest neighbor based outlier detection techniques and clustering based outlier detection techniques. We discuss these two techniques in two subsections 2.6.1 and 2.6.2.

2.6.1 Nearest neighbor based outlier detection techniques

A basic nearest neighbor outlier detection technique is based on the following definition: The outlier score of a data object is defined as its distance to its κ^{th} nearest neighbor in a given data set. This basic technique has been applied to detect land mines from satellite ground images [26] and to detect shorted turns (outliers) in the DC field windings of large synchronous turbine-generators [68]. Normally, a threshold is enforced on the outlier score to determine a point is outlier or not. On other hand, select top-m with highest outlier scores as outliers. These outlier detection techniques are based on the key assumption [3].

Assumption: Normal data points occur in dense neighborhoods, while outliers occur far from their closest neighbors.

A distance measure for data containing a mix of categorical and continuous attributes has been proposed for outlier detection by Otey *et. al.* [69]. The authors define links between two objects by adding distance for categorical and continuous attributes separately. For categorical attributes, the number of attributes for which the two objects have the same values defines the distance between them. For continuous attributes, a co-variance matrix is maintained to capture the dependencies between the continuous values.

Several variants of distance based techniques have been proposed to improve the efficiency. Some techniques prune the search space either by ignoring objects that cannot be outlier or by focusing on objects that are most likely to be outlier.

Angiulli and Pizzuti [70–72] with the aim of taking into account the whole neighborhood of the objects, have proposed to rank them on the basis of the sum of the distances from the κ-nearest neighbors, rather than considering solely the distance to the κ^{th} nearest neighbor. This sum is called the weight of the point, and it is used to rank the points having the largest values of weight. The points which are having largest weights are called outliers.

Edwin M. Knorr *et. al.* [1] deal with finding outliers (exceptions) in large, multidimensional data sets (*e.g.* $\delta \geq 5$). They presented two simple algorithms, both having a complexity of $O(\delta N^2)$, δ being the dimensionality and N being the number of objects in the data set. They also presented an cell-based algorithm that has a complexity that is linear with respect to N, but exponential with respect to δ. Finally, for disk-resident

data sets, they presented another version of the cell-based algorithm that guarantees at most three passes over a data set. Using experimental results it is showe that algorithms significantly outperform the two simple algorithms for $\delta \leq 4$.

- *Index-Based Algorithms:* Let N be the number of objects in data set $\mathcal{D}$, and let d be the underlying distance function that gives the distance between any pair of objects in $\mathcal{D}$. For an object o, the d-neighborhood of o contains the set of objects $q \in \mathcal{D}$ that are within distance d from o (i.e., $q \in \mathcal{D} \mid d(o,q) \leq d$). The fraction p is the minimum fraction of objects in $\mathcal{D}$ that must be outside the d-neighborhood of an outlier. For simplicity of discussion, let M be the maximum number of objects within the d-neighborhood of an outlier, i.e.,$M = N(1-p)$. From this, it is obvious that given p and d, the problem of finding all Distance-Based, $DB(p,d)$-outliers can be solved by answering a nearest neighbor or range query centered at each object o. More specifically, based on a standard multidimensional indexing structure, they executed a range search with radius d for each object o. As soon as $(M+1)$ neighbors are found in the d-neighborhood, the search stops, and o is declared a non-outlier; otherwise, o is an outlier. The above analysis only considers search time. When it comes to using an index-based algorithm, the index building cost alone, even without counting the search cost, almost always renders the index-based algorithms uncompetitive. The algorithm has worst case complexity of $O(\delta N^2)$

- *A Nested-Loop (NL) Algorithm:* To avoid the cost of building an index for finding all $DB(p,d)$-outliers, Algorithm NL uses a block-oriented, nested-loop design. Assuming a total buffer size of B% of the data set size, the algorithm divides the entire buffer space into two halves, called the first and second arrays. It reads the data set in the arrays, and directly computes the distance between each pair of objects. For each object o in the first array, a count of its d-neighbors is maintained. Counting stops for a particular object whenever the number of d-neighbors exceeds M. Algorithm NL avoids the explicit construction of any indexing structure, and its complexity is $O(\delta N^2)$. Compared to a object-by-object brute force algorithm that pays no attention to I/Os, Algorithm NL is superior because it tries to minimize I/Os.

- *A Cell-Based Approach:* Suppose our data objects are 2-D and quantize each of the data objects into a 2-D space that has been partitioned into cells or squares of length $l = \frac{d}{2\sqrt{2}}$. Let $C_{x,y}$ denote the cell that is at the intersection of row x and column y.

The Layer 1 (L_1) neighbors of $C_{x,y}$ are the immediate neighbor cells of $C_{x,y}$ defined in the usual sense, that is,

$$L_1(C_{x,y}) = \{C_{u,v} \mid u = x \pm 1, v = y \pm 1, C_{u,v} \neq C_{x,y}\}$$

$$L_2(C_{x,y}) = \{C_{u,v} \mid u = x \pm 3, v = y \pm 3, C_{u,v} \notin L_1(C_{x,y}), C_{u,v} \neq C_{x,y}\}$$

The following properties are useful in ruling out many objects as outlier candidates, the same is depicted in the Figure 2.1.

- ***Property 1*** Any pair of objects within the same cell is at most distance $\frac{d}{2}$ apart.
- ***Property 2*** If $C_{u,v}$ is an L_1 neighbor of $C_{x,y}$ then any object $P \in C_{x,y}$ and any object $Q \in C_{u,v}$ are at-most distance d apart.
- ***Property 3*** If $C_{u,v} \neq C_{x,y}$ is neither an L_1 nor an L_2 neighbor of $C_{x,y}$, then any object $P \in C_{u,v}$ and any object $Q \in C_{x,y}$ must be greater distance d apart.
- ***Property 4***
  1. If there are $>$ M objects in $C_{x,y}$, none of the objects in $C_{x,y}$ is an outlier.
  2. If there are $>$ M objects in $C_{x,y} \cup L_1(C_{x,y})$, none of the objects in $C_{x,y}$ is an outlier.
  3. If there are $\leq$ M objects in $C_{x,y} \cup L_1(C_{x,y}) \cup L_2(C_{x,y})$, every objects in $C_{x,y}$ is an outlier

  The complexity of algorithm is $O(c^{\delta}N)$

Ghoting *et. al.* [19] have proposed a two-phase algorithm for fast mining of distance-based outliers, which capture only a points approximate nearest neighbors, and not its nearest neighbors. They have divided the algorithm into two phases, in the first phase, the data set is preprocessed into bins. The points which are close to each other in space are likely to be placed in the same bin. In the second phase, an extension of the Nested Loop [1] algorithm is used over bins for determination of outliers. So the pre-processing performed in the first phase facilitates fast convergence to a points approximate nearest neighbors.

Wu *et. al.* [73] use sampling to improve efficiency of the nearest neighbor-based technique. The authors compute nearest neighbor of every object within a smaller sample from the data set. Therefore the complexity of the proposed technique is reduced to $O(SN)$, where S is the sample size chosen.

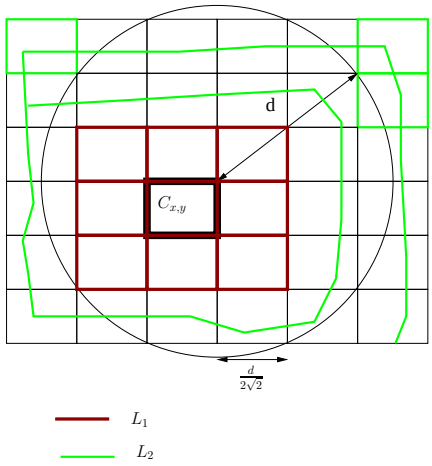

Figure 2.1: A Cell Based Approach [1]

Density-based outlier detection techniques estimate density of neighborhood of each data object. An object that lies in neighborhood with low density of objects is proposed to be anomalous. Where as an object that lies in a dense neighborhood is proposed to be normal.

Breunig *et. al.* [23] have proposed a local outlier factor (LOF), which is the degree of outlier-ness. This is measured as ratio of average local density of the κ nearest neighbors of the point and the local density of the data point itself. This is the first concept of an outlier which quantifies outlier-ness of an object on density of a point. The outlier factor is local in the sense that only a restricted neighborhood of each object is taken into account. The degree depends on how isolated the object is with respect to the surrounding neighborhood. For a normal object lying in a dense region, its local density is similar to that of its neighbors, while for an outlier object, its local density is lower than that of its nearest neighbors. Hence the outlier object get a higher LOF score. Since the LOF value of an object is obtained by comparing its density with those in its neighborhood, it has stronger modeling capability than a distance based scheme, which is based only on the density of the object itself. Note that the density based scheme does not explicitly categorize the objects into either outliers or non-outliers(If desired, a user can do so by choosing a threshold value to separate the LOF values of the two classes).

The following definitions give us the clear picture how authors have calculated outlier-ness of an object.

- κ-distance of an object p: For any positive integer κ, the κ-distance of object p, denoted as κ-distance(p), is defined as the distance $d(p, o)$ between p and an object

$o \in \mathcal{D}$ such that: (1) for at least κ objects $o' \in \mathcal{D} \setminus \{p\}$ it holds that $d(p, o') \leq d(p, o)$, and (2) for at least κ-1 objects $o' \in \mathcal{D} \setminus \{p\}$ it holds that $d(p, o') < d(p, o)$.

- κ-distance neighborhood of an object p ($\mathcal{N}_{\kappa\text{-distance}(p)}(p)$): Given κ-distance(p), the κ-distance neighborhood of p contains every object whose distance from p is not greater than the κ-distance, i.e. $\mathcal{N}_{\kappa\text{-distance}(p)}(p) = \{q \in \mathcal{D} \setminus \{p\} \mid d(p, q) \leq \kappa\text{-distance}(p)\}$.
- reachability distance of an object p w.r.t object o:

$$\textit{reach-dist}_\kappa(p, o) = max\{\kappa\text{-distance}(o), d(p, o)\}$$

- local reachability density of an object p:

$$lrd_k(p) = 1/\left[\frac{\sum_{o \in \mathcal{N}_{\kappa\text{-distance}(p)}(p)} \textit{reach-dist}_k(p,o)}{|\mathcal{N}_{\kappa\text{-distance}(p)}(p)|}\right]$$

- local outlier factor (LOF) of an object p:

$$LOF_k(p) = \frac{\sum_{o \in \mathcal{N}_{\kappa\text{-distance}(p)}(p)} \frac{lrd_k(o)}{lrd_k(p)}}{\mid \mathcal{N}_{\kappa\text{-distance}(p)}(p) \mid}$$

Several researchers have proposed variants of the LOF technique. Some variants estimate the local density of an object in a different way. The other variants have adapted original technique to more complex data types. Since the original LOF technique is $O(N^2)$ (N is the data size), several techniques have been proposed that improve the efficiency of LOF. Tang *et. al.* [74] have developed a measure which is the different from LOF, which is called Connectivity based outlier Factor(COF). LOF and COF differ, in the way k neighborhood for an object is calculated. In COF, the neighborhood for an object is computed in an incremental mode. To start, the closest object to the given object is added to the neighborhood set. The next object added to the neighborhood set is such that its distance to the existing neighborhood set is minimum among all remaining data objects. The distance between an object and a set of objects is defined as the minimum distance between the given object and any object belonging to the given set. The neighborhood is grown in this manner until it reaches size κ. Once the neighborhood is computed, the Outlier score (COF) is computed in the same manner as LOF. COF is able to capture regions such as straight lines.

Papadimitriou *et. al.* [9] have proposed a new measure called Multi-Granularity Deviation Factor (MDEF), which is also a variation of LOF. MDEF is measured based on the standard deviation of the local densities of the nearest neighbors. The outlier score

for the data object is the inverse of the standard deviation. The outlier detection technique is called Local Correlation Integral (LOCI), which not only finds outlier objects but also outlier micro-clusters.

Hautamaki *et. al.* [28] developed a measure, a simpler version of LOF, calculates a quantity called Outlier Detection using In-degree Number (ODIN) for each data object. For a given data object, ODIN is equal to the number of κ nearest neighbors of the data object which have the given data object in their κ nearest neighbor list. The inverse of ODIN is the outlier score for the data object.

2.6.2 Clustering Based Outlier Detection

Clustering is the unsupervised classification of objects (observations, data items, or feature vectors) into groups (clusters) [45]. A semi-supervised clustering is also been explored by Basu *et. al.* [75]. Even though clustering and outlier detection appear to be fundamentally different from each other, several clustering-based Outlier detection techniques have been developed. Clustering based outlier detection techniques are grouped into three classes by Varun *et. al.* [3]. That is Class 1, Class 2 and Class 3 are based on the following assumptions respectively:

Class 1: Normal data objects belong to a cluster in the data, while Outliers do not belong to any cluster.

Class 2: Normal data objects lie close to their closest cluster centriod, while Outliers are far away from their closest cluster centriod.

Class 3: Normal data objects belong to large and dense clusters, while Outliers either belong to small or sparse clusters.

Techniques based on Class 1 assumption apply a known clustering algorithm to the data set and declare any data object that does not belong to any cluster as outlier. Several clustering algorithms that do not force every data object to belong to a cluster, such as DBSCAN [76], ROCK [77], and SNN clustering [78] can be used. A disadvantage of such techniques is that they are not optimized to find outliers, since the main aim of the underlying clustering algorithm is to find clusters. Techniques based on Class 2 assumption consist of two steps. In the first step, the data is clustered using a clustering algorithm. In the second step, for each data object, its distance to its closest cluster centriod is calculated as its outlier score. A number of outlier detection techniques that follow this two step approach have been proposed using different clustering algorithms. Smith *et. al.* [79]

studied Self-Organizing Maps (SOM), k-means Clustering, and Expectation Maximization (EM) to cluster training data and then use the clusters to classify test data. In particular, SOM [80] has been widely used to detect outliers in a semi-supervised mode in several applications such as intrusion detection [79,81,82] and fault detection [83,84]. Barbara *et. al.* [85] propose a technique that is robust to outliers in the training data. The authors first separate normal objects from outliers in the training data, using frequent item-set mining, and then use the clustering-based technique to detect Outliers. Techniques based on the second assumption can also operate in semi-supervised mode, in which the training data is clustered and objects belonging to the test data are compared against the clusters to obtain an outlier score for the test data object. If the training data has objects belonging to multiple classes, semi-supervised clustering can be applied to improve the clusters.

He *et. al.* [86] incorporated the knowledge of labels to improve on their unsupervised clustering-based outlier detection technique [87] by calculating a measure called semantic outlier factor, which is high if the class label of an object in a cluster is different from the majority of the class labels in that cluster. A disadvantage of such techniques is that if the outliers in the data form clusters by themselves, these techniques are not able to detect such outliers. Techniques based on Class 3 assumption declare objects belonging to clusters whose size and/or density is below a threshold, as outlier. Several variations of the third category of techniques have been proposed [27, 32, 87–89].

The technique proposed by He *et. al.* [87], called $FindCBLOF$, assigns an outlier score known as Cluster-Based Local Outlier Factor (CBLOF) for each data point. The CBLOF score captures size of the cluster to which the data object belongs, as well as distance of the data object to its cluster centriod. Initially they divided the data set to set of clusters by squeezer algorithm and these clusters are into two groups, namely large clusters and small clusters, based on number of points in each cluster. For each data point they calculated the CBLOF and declare the outliers. The idea is clearly describe as follows: Let $A_1, ..., A_m$ be a set of attributes with domains $D_1, ..., D_m$ respectively. Let the data set $\mathcal{D}$ be a set of objects where each object $t : t \in D_1 \times .... \times D_m$. The results of a clustering algorithm executed on $\mathcal{D}$ is denoted as: $C = \{C_1, C_2, ..., C_k\}$ where $C_i \cap C_j = \emptyset$; and $C_1 \cup C_2.....C_k = D$. The number of clusters is k.

The clusters are divided into large and small clusters. Suppose $C = \{C_1, C_2, ..., C_k\}$ is the set of clusters in the sequence that $|C_1| \geqslant |C_2|.... \geqslant |C_k|$. Given two numeric parameters α and β, we define b as the boundary of large and small cluster if one of the following properties holds.

$$(|C_1| + |C_2| + ... + |C_b|) \geqslant |D| * \alpha$$

$$|C_b|/|C_{b+1}| \geqslant \beta$$

Then, the set of large cluster is defined as: $LC = \{C_i | i \leqslant b\}$ and the set of small cluster is defined as: $SC = \{C_j | j > b\}$.

Cluster-based local outlier factor: Suppose $C = \{C_1, C_2, ..., C_k\}$ is the set of clusters in the sequence that $|C_1| \geqslant |C_2|.... \geqslant |C_k|$ and the meanings of α, β, b, LC ans SC are same as they are formalized. For any object t, the cluster-based local outlier factor of t is defined as:

$$CBLOF(t) = \begin{cases} |C_i| * min(distance(t, C_j)) & \text{where } t \in C_i, C_i \in SC \text{ and} \\ & C_j \in LC \text{ for } j = 1 \text{ to } b \\ |C_i| * (distance(t, C_i)) & \text{where } t \in C_i \text{ and } C_i \in LC \end{cases}$$

H. Huang *et. al.* [90] have proposed a rank based detection algorithm (RBDA). In this approach outliers are identified based on mutual closeness of a data point and its neighbors. The mutual closeness of any two points p and q is defined as $p \in D$ and $q \in N_\kappa(p)$. That means, q is close to p because it belongs to κ-neighborhood of p.

Hubballi *et. al.* [22] have proposed a nearest neighbor based outlier detection algorithm (NDoT) works by a voting mechanism. They calculated nearest neighborhood factor(NNF) for all the points in the data set, this factor is sorted and declared the highest 1/3 points as outliers.

Lian *et. al.* [91] have proposed a cluster-based outlier factor(CBOF). Their main assumption is that not only a single point can be outlier but also a small cluster can probably be an outlier cluster. They considered the local behavior of data. Initially used LDBSCAN [92] to cluster the entire data set and based on cardinality of clusters they divided the clusters into two classes. One class is outlier class of clusters which contain only 10% of data points in each cluster, another is non outlier class of clusters. Subsequently for each point in the outlier clusters, they calculated local reachability density(LRD) [92] for each point in a cluster. Finally using the CBOF parameter of a cluster they declare whether the cluster is outlier cluster or not.

Papadimitriou *et. al.* [9] have proposed a method for evaluating outlierness, which is called the local correlation integral (LOCI). Which is effective for detecting outliers and groups of outliers (micro-clusters). It also provides an automatic, data-dictated cutoff to

determine whether a point is an outlier. It can also provide a LOCI plot for each point; this plot summarizes a wealth of information about the data in the vicinity of the point, determining clusters, micro-clusters, their diameters and their inter-cluster distances.

Bay and Schwabacher [93] showed that for sufficiently randomized data, a simple pruning step could result in the average complexity of the nearest neighbor search will come down to nearly linear. After calculating the nearest neighbors for a data point, the algorithm sets the outlier threshold for any data point to the score of the weakest outlier found so far. Using this pruning procedure, the technique discards objects that are close, and hence not interesting. The performance of this algorithm mainly depends on the three assumptions, violations of which can lead to poor performance. First, assumes that the data is in random order. If the data is not in random order and is sorted then the performance can be poor. Second, depends on the independence of data objects. Third, perform poorly when the data does not contain outliers. In the worst case, the performance of the algorithm is very poor. Because of the nested loops, it could require $O(N^2)$ distance computations.

Ramaswamy *et. al.* [18] have proposed partition based pruning method for outlier detection. They proposed a method that partitions the input data set into disjoint subsets based on BIRCH clustering algorithm [94], it stores a compact summarization for each cluster in Clustering Featuring tree (CF-tree) which is a balanced tree structure similar to an R-tree [95]. For each partition they calculated two boundaries lower and upper. If the partition does not satisfy the boundary conditions, the partitions are pruned. This results in substantial savings in computations. The steps are described below.

1. Generate Partition
   - A clustering algorithm (BIRCH) is used to cluster the data, each cluster is a separate partition.
2. Compute bounds on κth distance of a point (D^{κ}), for points in each partition.
3. Identify candidate partitions containing outliers.
   - Compute $minDkDist$ (the lower bound on D^{κ} for the n outliers)
   - If $P.upper$ for a partition P is less than $minDkDist$, none of the points in P can possibly be outliers.
   - Only partitions P for which $P.upper >= minDkDist$ are candidate partitions.
4. Compute outliers from points in candidate partitions

Table 2.2: Comparisons of different algorithms

Algorithm	Data set size(N)	Dimensions(δ)	complexity
Index-based [1]	2,000,000	$\leqslant 5$	$O(\delta N^2)$
Nested-based [1]	2,000,000	$\leqslant 5$	$O(\delta N^2)$
Cell-based [1]	2,000,000	$\leqslant 4$	$O(c^\delta + N)$
RSS [18]	101000	10	$O(\delta N^2)$
LDOF [21]	14500	32	$O(\delta N^2)$
LOF [23]	375	64	$O(\delta N^2)$
COF [74]	1700	40	$O(\delta N^2)$
FindCBLOF [87]	798	38	$O(\delta N^2)$

In [96], a feature bagging approach for detecting outliers has been proposed. It combines results from multiple outlier detection algorithms that are applied using different set of features. Every outlier detection algorithm uses a small subset of features that are randomly selected from the original feature set. As a result, each outlier detector identifies different outliers, and assigns to all data objects outlier scores that correspond to their probability of being outliers. The outlier scores computed by the individual outlier detection algorithms are then combined in order to find the better quality outliers.

In Table 2.2, the computational complexity of different outlier detection techniques are presented from the literature.

2.7 Outlier Score

In this thesis outlier score of a point is used to measure the degree of outlierness. The score gives the level of deviation of a point from its neighbors. Based on score of the data points the selection of top-m points as outliers.

Zhang *et. al.* [21] proposed a local distance-based outlier detection($ldof$) for scattered real-world data sets. This is measured as a ratio of average distance of the κ nearest neighbors of the points and the average distance among the neighbors of the point. The outlier factor of an object determine the degree to which the object deviates from its neighborhood. Calculating $ldof$ for all points in the data set, makes overall complexity $O(N^2)$. They have analyzed theoretically the properties of $ldof$ like lower bound, false detection probability and parameter settings.

The local distance-based outlier factor of p is defined as:

$$ldof(p) := \frac{\bar{d}_p}{\bar{D}_p} \tag{2.13}$$

$\bar{d}_p$ (KNN distance of p): Let $\mathcal{N}_p$ be the set of κ-nearest neighbors of object p (excluding p). The κ-nearest neighbors distance of p equals the average distance from p to all objects in $\mathcal{N}_p$. More formally, let $d(p,q) \geq 0$ be a distance measure between objects p and q. The κ-nearest neighbors distance of object p is defined as:

$$\bar{d}_p := \frac{1}{\kappa} \sum_{q \in \mathcal{N}_p} d(p,q) \tag{2.14}$$

$\bar{D}_p$ (KNN inner distance of p): Given $\mathcal{N}_p$ of object p, the κ-nearest neighbors inner distance of p is defined as the average distance among objects in $\mathcal{N}_p$.

$$\bar{D}_p := \frac{1}{\kappa(\kappa-1)} \sum_{q,q' \in \mathcal{N}_p, q \neq q'} d(q,q') \tag{2.15}$$

The definition of $ldof$ parameter indicates that, if the average distance of the point from the nearest neighborhood set is high then the point lies away from the neighbors, other wise point lies close to neighbors.

Lower bound of $ldof$($ldof_{lb}$): In any data set, an object is outlier if $ldof > 1$ has been proved by the authors. However, the threshold is problem dependent due to the complex structure of real-world data sets. Under some continuity assumption, we can calculate an asymptotic lower bound on $ldof$, denoted as $ldof_{lb}$. $ldof_{lb}$ indicates that an object is an inlier (or normal) if its $ldof$ is smaller than $ldof_{lb}$. Even with the theoretical analysis, the authors claimed that it is still hard to determine a threshold for $ldof$ to identify outliers in an arbitrary data set. Hence forth they employed top-m style outlier detection, which ranks the m objects with the highest $ldof$s, and for choosing κ, it is beneficial to use a large neighborhood size κ. The complexity of this $ldof$ method is $O(N^2)$.

From Table 2.2, it is evident that the computational complexity of outlier detection methods is directly related to the the number of data points N and the number of attributes δ of the data set. Therefore, if these two quantities could be reduced by removing some irrelevant data for outlier detection then the computation time can be reduced considerably.

Next chapter onwards, we present details of the research work done for this thesis. We start with the clustering and cluster pruning; and three methods based on cluster pruning are presented in Chapter 3.

Chapter 3

Cluster based pruning for Outlier Detection

3.1 Introduction

Outlier detection is an important task in many fields. A data set may contain data objects that do not comply with the general behavior or model of the data. These data objects are called outliers. Most data mining methods discard outliers as noise or exception. However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones. Outlier detection is very interesting, because outlier may get introduced into the data for various reasons. Detecting outliers in various applications using different techniques is a non-trivial task. Identifying outliers based on distance computation is computationally intensive because the distance computations are to be computed for all the data points present in the data set. In literature, clustering techniques have been used to reduce the computations. Initially, a clustering algorithm is used to partition the data set into clusters and data points with high similarity generally fall in one cluster. Subsequently exploiting the properties or behaviors of clusters to identify the clusters which probably do not contain any outliers and prune such clusters. Zhang, *et. al.* [21] have proposed a parameter namely local distance based outlier factor ($ldof$) and depending on the value of the $ldof$ of a point, one can take a decision about the outlier-ness of that point. Finally $ldof$ values are calculated for all the unpruned data points.

In Chapter 2, we have discussed many classifications, categories of outliers detection techniques and dissimilarity measures for different data types. We have also discussed

some of the important clustering techniques that are used in our work. In Subsection 2.6 of Chapter 2 we have presented some of well accepted outlier detection techniques that are related to our work. In this chapter we present three methods to prune some of the clusters depending on the properties of the clusters. Further $ldof$ values are calculated for unpruned points to identify the outliers.

First Method: We calculate the $ldof$ of the centriods of all the clusters. High $ldof$ value of a centriod basically indicates that this cluster is away from other clusters and probably outliers are present in this cluster. Depending on the $ldof$ values of centriods, we prune some of the clusters whose $ldof$ value of the centriod is low.

Second Method: We calculate the radius of each cluster. If the radius of a cluster is less it indicates that the points of this cluster are uniformly distributed and it is a dense cluster. Probably this cluster may not contain outliers. So, we can prune such clusters.

Third Method: We calculate the distance of the centriods of all the clusters from the overall centriod of the given data set. If this distance is more, it indicates that the points of this cluster is deviating from the normal points and that cluster may contain outliers. So, depending on the distance of the centriod, we may prune some of the clusters.

The experimental results show that these methods able to identify the outliers though some of the points are pruned from the data set before calculating the $ldof$ value. The results are comparable to the results reported by Zhang *et. al.* [21].

3.2 Proposed Methods

The main emphasis of our proposed methods is to reduce the number of computations while detecting outliers of a given data set. To declare a point as an outlier we need to use some parameters to classify an object as outlier. Some of the outlier detection techniques are based on two steps. In the first step, the data is clustered using a clustering algorithm. In the second step, for each data point an outlier score is calculated, to classify the data point as outlier or inlier.

Clustering a data set, enables to visualize the data in terms of groups. The data which exist in terms of a group helps to distinguish between inlier and outlier points. So the clustering algorithm which partition the data into different groups is used. We use one

of the most popular distance based clustering method, k-means [48] and which gives the clusters along with its centriods. Centriod is the mean position of all the points in all of the coordinate directions. The radius of a cluster is the average distance of all the points from the centriod. If the radius is small the points are close to the centriod otherwise they are well spread around the centriod.

For a given cluster C, of m δ-dimensional data points $\overrightarrow{p_i}$ where $i = 1, 2, ...m$, the centriod $\overrightarrow{g_c}$, radius R_C [59, 97] are defined as:

$$\overrightarrow{g_c} = \frac{\sum_{i=1}^{m} \overrightarrow{p_i}}{m} \tag{3.1}$$

$$R_C = \sqrt{(\frac{\sum_{i=1}^{m} (\overrightarrow{p_i} - \overrightarrow{g_c})^2}{m})} \tag{3.2}$$

The outlier score of a point is calculated using the parameter proposed in [21]. The definition of $ldof$ parameter indicates that, if the average distance of the point from the nearest neighborhood set is high then the point lies away from the neighbors, other wise point lies close to neighbors. The parameters centriod of a cluster, radius of a cluster and $ldof$ of a point are used in our proposed methods.

The method one (M-1), which is based on $ldof$ value of the centriod of a cluster. The $ldof$ values of the centriod of a cluster indicates how much a cluster is deviated from its neighbors. High $ldof$ value of the centriod of a cluster means the cluster is away from its neighbors. So we calculated $ldof$ values of centriods of the clusters. Few clusters are pruned depending on the $ldof$ values of the centriods and the number of clusters to be pruned is decided by the user. For simplicity, half of the clusters are pruned.

To find the number of clusters to be pruned depending on the distribution of data points, we propose method two (M-2). This method is based on the radius of clusters. Small radius of a cluster indicates that it is a dense cluster and distribution of points are uniform. In such clusters chances of having outliers are less. We calculate the average radius of all the clusters and clusters having radius less than average are pruned.

As the number of clusters increases the number of points inside a cluster is less and the radius of the cluster is also small. In this case percentage of pruning of points may be less. So we proposed method three (M-3) based on the centriod of the data points. The clusters whose centriods are close to the centriod of the data points are pruned.

3.2.1 Method based on $ldof$ of centriod of cluster

The use of the $ldof$ parameter efficiently to capture the outliers is the main idea behind the proposed method. Calculating $ldof$ values for all the points in a data set is computationally intensive. So by clustering we identify a group of points that are close to each other. Identifying the group that contains outliers is one of the objectives. The group center (i.e. centriod of the group) acts as a representative point of the group/cluster. The $ldof$ score of the centriod of the cluster tells the degree of outlierness of the cluster. So identifying the outlierness of the centriod projects the degree of outlierness of the cluster with respect to all other clusters. For example, as presented in Figure 3.1 the data set is divided into k clusters (namely C1, C2, ..., C10). Based on clustering algorithm, the clusters have different number of points in each cluster. It is observed from Figure 3.1, some of the clusters are shaded which are lying inside and are surrounded by other clusters. Since these clusters are surrounded by the other clusters these clusters have low $ldof$ values compared with the other clusters. Hence the clusters C3, C5, C6, C7, C8 are pruned. The degree of outlierness of the cluster is calculated based on $ldof$, assuming the points present in these clusters are inliers as they are also surrounded by the other cluster of points. Clusters having high $ldof$ values have high probability to contain the outliers, this is because the neighborhood of these clusters are less when compared to the neighborhood of the clusters whose $ldof$ is low. Therefore the clusters having low $ldof$ values may not contain outliers in these clusters. Hence half of the clusters (C3, C5, C6, C7, C8) with low $ldof$ values are removed as shown in Figure 3.1. The proposed method is presented in Algorithm 3.1. We briefly describe the steps.

- **Generating clusters:** Initially, we cluster the entire data set into k clusters using k-means clustering algorithm.
- **Pruning some clusters:** The $ldof$ value for all the centriods of the clusters are calculated. If the total number of points inside any cluster is less than the number of outliers, the cluster pruning is not performed on those clusters even though the $ldof$ value of the cluster's centriod is low. Based on the $ldof$ values of the centriods, we prune half of the clusters whose $ldof$ values are low.
- **Computing outlier points:** From the remaining unpruned clusters, calculate $ldof$ values for all the remaining points. The m points having high $ldof$ values are reported as top-m outliers.

Algorithm 3.1 Cluster Pruning Based Outlier Detection Algorithm (Method-1)

Input : $\mathcal{D}$-Data Set, k-number of clusters, i-number of iterations, n-number of outliers

Output: m-points having high $ldof$ value.

Begin

$U \leftarrow \emptyset$; // U set of unpruned data points

$U_G \leftarrow \emptyset$; // U_G set of unpruned centriods

Set $\langle C, G\rangle \leftarrow kmeans(k, i, \mathcal{D})$

// C set of k clusters and G set of k centriods

// $C = \{c_1, c_2, ..., c_k\}; G = \{g_1, g_2, ..., g_k\}$

For $\forall g_j \in G$ **do**

  $ldof(g_j)$; // calculating $ldof$ value of all cluster centriods

end

For $j = 1$ *to* k **do**

  If $|c_j| < n$ **then**

    add points of c_j to U add g_j to U_G

  end

end

Set $C \leftarrow C \setminus U$

Set $G \leftarrow G \setminus U_G$

For $\forall g_j \in G$ **do**

  Sort the $ldof(g_j)$ values ; // increasing order of $ldof(g_j)$ values

end

For $j = 1$ *to* $\lceil \frac{k}{2} \rceil$ **do**

  Set $C \leftarrow C \setminus c_j$) ; // prune c_j whose $ldof(g_j)$ is low

end

Add points of C to U

For $\forall p_i \in U$ **do**

  calculate $ldof(p_i)$

end

Sort the points in descending order according to their $ldof(p_i)$ values.

First m points with high $ldof(p_i)$ values are the desired outliers.

End

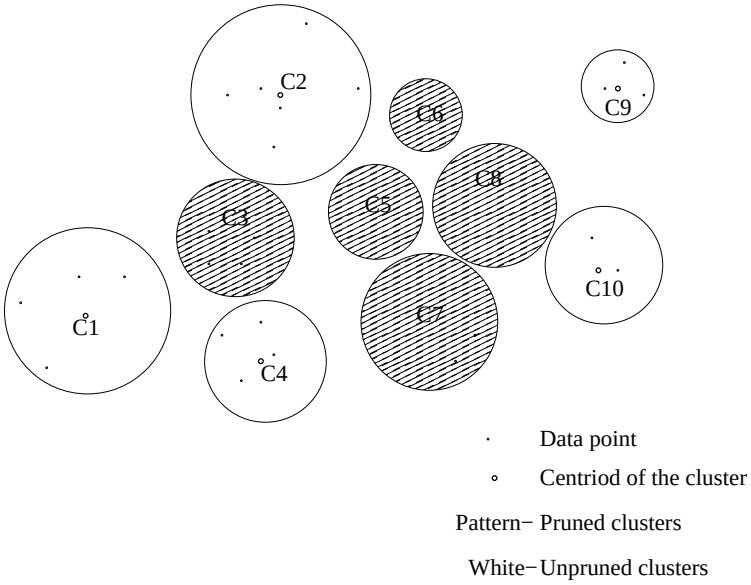

Figure 3.1: Half of the clusters pruned based on $ldof$ values of the centriods

3.2.2 Method based on radius of clusters

In the method presented in Section 3.2.1, half of the clusters are pruned irrespective of the number of clusters generated. Pruning half of clusters is not very obvious solution for a problem. Learning from the clusters itself is the better solution for identifying the clusters that are to be pruned. So in this method average radius parameter is used to identify the clusters for pruning. The radius of a cluster is the average distance of all points to the centriod. If the radius of a cluster is small it indicates that the points are uniformly distributed around the centriod and also lie close to the centriod. From Figure 3.2, there are two clusters (C3, C7) of equal size, but distribution of the points in that clusters are different so the radius are also different. The clusters with small radius have less possibility to containing outliers. Pruning these clusters reduces the computations. So the clusters whose radius is less than average radius are pruned in this method. For example as shown in Figure 3.2 we divide entire data set into seven clusters (C1, C2,....., C7) using a clustering algorithm. For each cluster a radius is calculated (the dotted concentric circle for each cluster represents the range of radius of the clusters). If the radius value is high, the spread or distribution of the points in the cluster is wide (i.e. majority of the points lie well apart from the centriod of the cluster). Other wise the most of the points lie close to the centriod. The average value of all radius of the clusters give us approximate value that can be used as a threshold to take the decision for pruning some of the clusters. Hence

based on the average value of the radius, some of the clusters are pruned which are having smaller radius than average value. The method is depicted in Algorithm 3.2 and briefly describe the steps as follows.

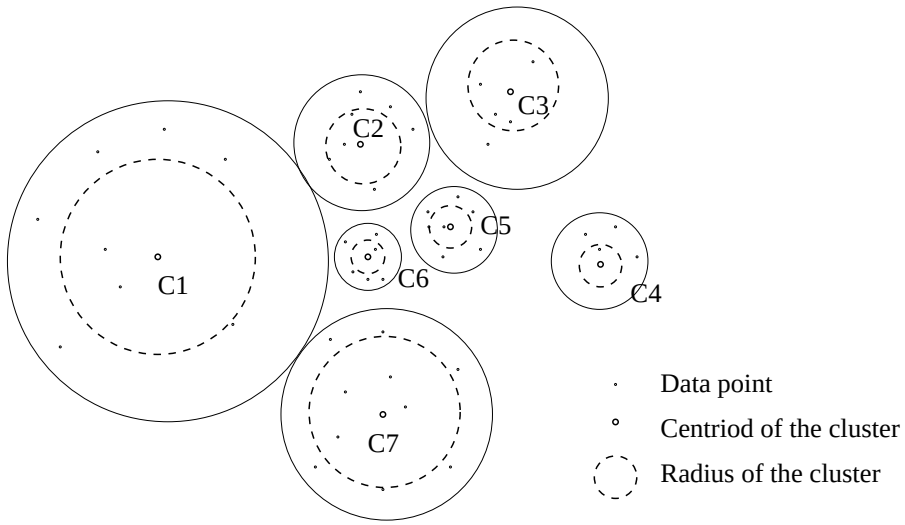

Figure 3.2: Average Radius Based Pruning

- **Generating clusters:** Initially, we cluster the entire data set into k clusters using k-means clustering algorithm.
- **Radius calculation:** We calculate radius of each cluster. Next, we calculate average radius.
- **Pruning clusters:** If the number of points inside any cluster is less than number of outliers, the cluster pruning is not performed on those clusters even radius of the cluster is less than average radius. For rest of the clusters, if radius of the cluster is less than average radius, we prune the cluster.
- **Computing outlier points:** Calculate $ldof$ values for all the points that are left in the unpruned clusters. The m points with highest $ldof$ values are reported as top-m outliers.

3.2.3 Method based on centriod of the clusters

The method presented in Section 3.2.2 uses radius of clusters to identify clusters for pruning. The method does not prune more number of clusters as the number of clusters

Algorithm 3.2 Cluster Pruning Based Outlier Detection Algorithm (Method-2)

Input : $\mathcal{D}$-Data set, k-required number of clusters, i-number of iterations, n-number of outliers.

Output: m-points having high $ldof$ value.

Begin

$U \leftarrow \emptyset$; // U set of unpruned data points

Set $\langle C, G\rangle \leftarrow kmeans(k, i, \mathcal{D})$

// C set of k clusters and G set of k centriods

// $C = \{c_1, c_2, ..., c_k\}; G = \{g_1, g_2, ..., g_k\}$

For $\forall c_j \in C$ **do**

  Set $R_{c_j} \leftarrow (\sqrt{\frac{\sum_{i=1}^{m}(p_i - g_j)^2}{|c_j|}})$; // R_{c_j} radius of clusterc_j

end

Set $AvgRad \leftarrow \frac{\sum_{j=1}^{k} R_{c_j}}{k}$; // $AvgRad$ average radius

For $\forall c_j \in C$ **do**

  If $|c_j| < n$ **then**

    add points of c_j to U

  end

  Else

    If $R_{c_j} < AvgRad$ **then**

      prune(c_j)

    end

    Else

      add points of c_j to U

    end

  end

end

For $\forall p_i \in U$ **do**

  calculate $ldof(p_i)$

end

Sort the points in descending order according to their $ldof(p_i)$ values.

First m points with high $ldof(p_i)$ values are the desired outliers.

End

increases, this is because average radius becomes very small. In order to obtain better pruning, an idea is proposed based on the assumption that inliers are the points which lie close to the centriod and outliers are the points which lie away from the centriod. From Figure 3.3 it is observed that the possibility of the clusters containing outliers is high if the clusters are away from the center of the data set. The same is observed from Figure 3.3 the chances of having outliers in clusters C1, C8 and C10 is very high. In the same way the clusters which are close to center of the data set have very less probability to contain outliers. So pruning these clusters is good for reducing computations. The distance and average distance determined by the centriods of the clusters with respect to the center of the data set is used in identifying possible clusters containing outliers. From Figure 3.3 there are three clusters C1, C8 and C10 which are having distance from the overall centriod is greater than average distance from the overall centriod. The points present in these clusters which are well separated from the center of the data set are the candidate outliers. The other clusters can be pruned as the distance is small when compared with average distance from the overall centriod (i.e. the clusters centriods which fall below the dotted circle are pruned). The proposed algorithm is depicted in Algorithm 3.3. The proposed method is briefly describe in the following steps.

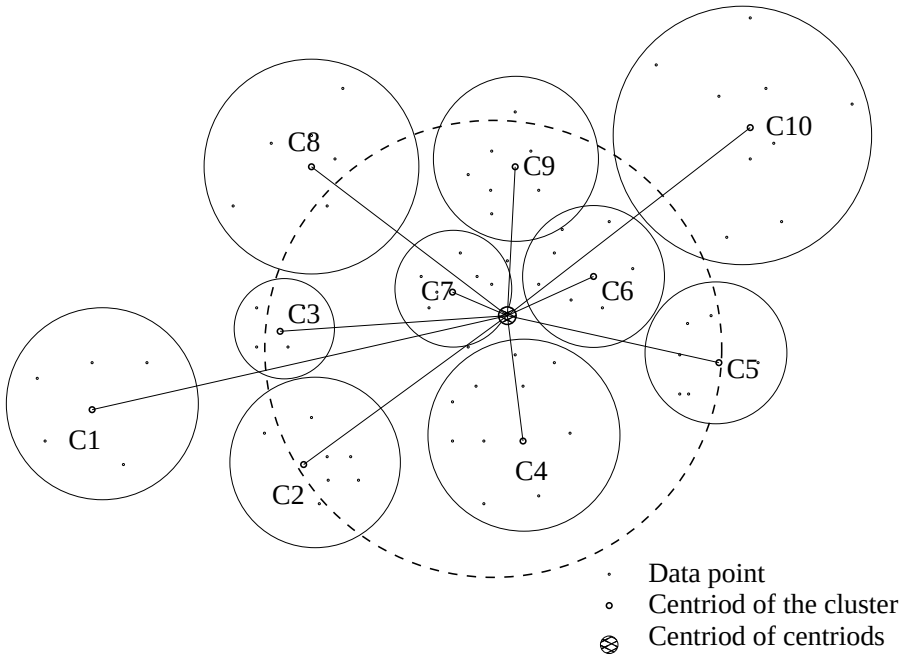

Figure 3.3: Data set center based Pruning

- **Generating clusters:** Initially, we cluster the entire data set into k clusters using k-means clustering algorithm.

- **Calculating overall centriod:** Using only the centriod points of all the clusters we calculate overall centriod.

- **Calculate average distance from the overall centriod**. Using centriods of all the clusters we calculate average distance from overall centriod. It is average sum of distances between the overall centriod and the centriods of the clusters.

- **Pruning some clusters:** If the number of points in any cluster is less than number of outliers, the cluster pruning is not applied. For rest of the clusters if the distance between centriod of cluster to overall centriod is less than average distance from overall centriod, we prune the cluster.

- **Computing outlier points:** From the remaining clusters, calculate $ldof$ value for all the unpruned points. The m points having high $ldof$ values are reported as top-m outliers.

3.3 Computational Analysis

The computational complexity of the $ldof$ method proposed by [21] is N^2, because $ldof$ values are calculated for all the data points. As we are reducing size of the data set by pruning, computational complexity of the proposed methods is reduced considerably than N^2. Initially for clustering the computational complexity of k-means algorithm is $k*i*N$, where k is the number of clusters to be formed, i is the number of iterations and N is the number of data points. After pruning we left out with few points (set of unpruned points U). Calculating $ldof$ values for all these points, the computation complexity is $(U)^2$. So the total computation of our method is $k*i*N+k*k+(w*N)^2$, where w indicates the fraction of data points that we have after pruning, which is around 0.5. Since k and i are small, so the total computations of our methods are reduced considerably. The computational complexity for the second method is little bit more than the other two methods because of the radius calculation of all the clusters.

Algorithm 3.3 Cluster Pruning Based Outlier Detection Algorithm (Method-3)

Input : $\mathcal{D}$-Data set, k-number of clusters, i-number of iterations,n-number of outliers

Output: m-points having high $ldof$ value.

Begin

$U \leftarrow \emptyset$

Set $\langle C, G\rangle \leftarrow kmeans(k, i, \mathcal{D})$

// C set of k clusters and G set of k centriods

// $C = \{c_1, c_2, ..., c_k\}; G = \{g_1, g_2, ..., g_k\}$

Set $Avg_G \leftarrow (\sqrt{\frac{\sum_{j=1}^{k}(g_j - \bar{G})^2}{k}})$

// Avg_G average distance of all g_j from overall centriod $\bar{G}$

For $\forall c_j \in C$ **do**

  If $|c_j| < n$ **then**

    add points of c_j to U

  end

  Else

    If $d(g_j, \bar{G}) < Avg_G$ **then**

      prune(c_j)

    end

    Else

      add points of c_j to U

    end

  end

end

For $\forall p_i \in U$ **do**

  calculate $ldof(p_i)$

end

Sort the points in descending order according to their $ldof(p_i)$ values.

First m points with high $ldof(p_i)$ values are the desired outliers.

End

3.4 Experimental Results

Performance of an algorithm is evaluated based on the number of outliers detected by the algorithm. We perform experiment for all the three methods that have been proposed in this Chapter and compare our findings with the result reported in [21] by Zhang, et. al.

In an outlier detection algorithm, if there are n true outliers in a data set and if an algorithm can detect n_t outliers among top-m points then these are termed as true positive (TP). The true outliers that cannot be detected by an algorithm are false negative (FN); so $n = TP + FN$. If the algorithm reports some inliers in top-m points then these are false positive (FP); so top-$m = TP + FP$ [90, 98–100]. Then, Precision, which measures the proportion of true outliers in the top-m suspicious instances, is defined as:

$$Precision = \frac{n_t}{\text{top-}m} = \frac{TP}{TP + FP}$$

Recall, which measures the accuracy of an algorithm, is defined as:

$$Recall = \frac{n_t}{n} = \frac{TP}{TP + FN}$$

Precision Recall (PR) curves are increasingly used in the machine learning community, particularly for imbalanced data sets where one class is observed more frequently than the other class. On these imbalanced or skewed data sets, PR curves are a useful alternative to Receiver Operating Characteristic (ROC) curves that can highlight performance differences that are lost in ROC curves [101]. Besides visual inspection of a PR curve, algorithm assessment often uses the area under curve-PR (AUCPR) as a general measure of performance irrespective of any particular threshold or operating point. To evaluate the performance of each method, we used the area under the curve-precision-recall (AUCPR), which is a typical criterion to measure the effectiveness of outlier detection methods [102, 103]. It takes values from 0 to 1. The value 1 is the best score, and quantifies whether the algorithm is able to retrieve outliers correctly.

In real-world data repositories, it is hard to find a data set to measure the performance of an outlier detection algorithm, because most of the data sets are having data of similar behaviors or collection of different classes of data items. So, in our experiment we use four real data sets.

IRIS data set: It is a data set of type of Iris plant. The data set contains 3 classes with 4 attributes of 50 instances each: Iris setosa, Iris versicolour and Iris virginica. The Iris setosa is iris plant that is linearly separable from each other class. We took

Iris versicolour and Iris virginica as inliers and added five Iris setosa (outliers) to the inliers.

Ionosphere data set: It is called Johns Hopkins University Ionosphere data set. Its contains 351 data points with 35 attributes (34 attributes are continuous, one attribute which classify the data point as good or bad). For our experiment we have taken all the 225 good data points as normal points. Out of 116 bad data points we took 10 points and added into good data points as outliers. So for our experiment we took total of 235 data points (225 (inliers)+10 (outliers)).

WDBC data set: It is a medical data set, WDBC (Diagnosis), which has been used for nuclear feature extraction for breast tumour diagnosis. The data set contains 569 medical diagnosis records (objects), each with 32 attributes (ID, diagnosis, 30 real-valued input features). The diagnosis is binary: Benign and Malignant. We considered the objects labeled Benign as normal data and added 10 numbers of Malignant diagnosis records into normal points as outliers.

Shuttle data set: It is originally used for classification, named Shuttle Data (STATLOG VERSION). The data set contains 58000 objects (Training set contains 43500 objects and test set contain 14500 objects). Each object has 9 real-valued features and an integer label (1-7). We consider the objects (only 13) with label 2 as outliers, and consider the objects with labels 1 as normal data, in total the number of points used for the experiment is 11491.

In all the data sets, one class is used as normal points and other class as abnormal points. We took all the normal points and injected few abnormal points into the normal class as outliers. All the data sets used in this thesis are taken from UCI repository [104].

We carried out experiments for three methods that we have proposed in this chapter:

- M-1: Based on $ldof$ of centriod of cluster.
- M-2: Based on radius of the clusters.
- M-3: Based on centriod of the clusters.

3.4.1 Iris Data Set

By varying the number of clusters, we measured the percentage of pruning of the data points that are presented in Table 3.1. It is observed from Table 3.1, the percentage of

Table 3.1: Pruning ratio for IRIS data set

k number of clusters	% of data points pruned for M-1	% of data points pruned for M-2	% of data points pruned for M-3
8	67.61	47.61	83.80
12	60.95	25.71	74.28
16	50.47	8.57	68.57
20	46.66	0	54.28

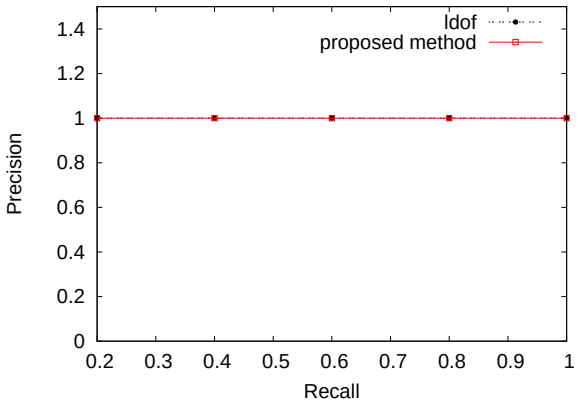

Figure 3.4: Precision Recall Curve for IRIS data set for M-1

pruning for the method M-1 varies from 67% to 46%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Similarly for the method M-2, we could able to find all the outliers but the pruning percentage is very low when the number of clusters are more. As mentioned in Section 3.2.3, there is a increase percentage of pruning in the method M-3 in comparison to the method M-2. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 3.4 shows that AUCPR for the method M-1 is same as that of the $ldof$ method, even though we have pruned 67% to 46% of points from the data set. The performance in terms of detecting outliers of the method M-1 is same as that of existing method. The AUCPR for the method M-2 and method M-3 are presented in Figure 3.5 and Figure 3.6 respectively. It is observed that the performance of the method M-2 and M-3 are also same as that of existing method though there is a considerable pruning of data points.

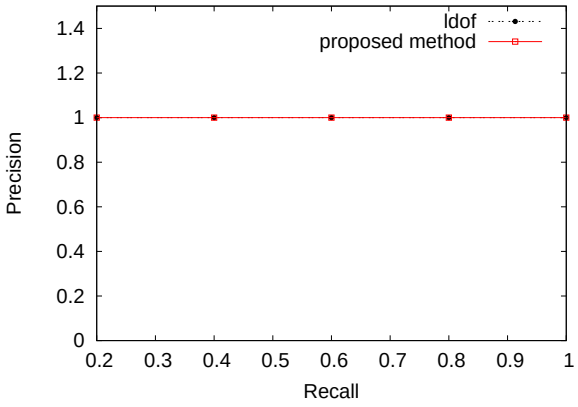

Figure 3.5: Precision Recall Curve for IRIS data set for M-2

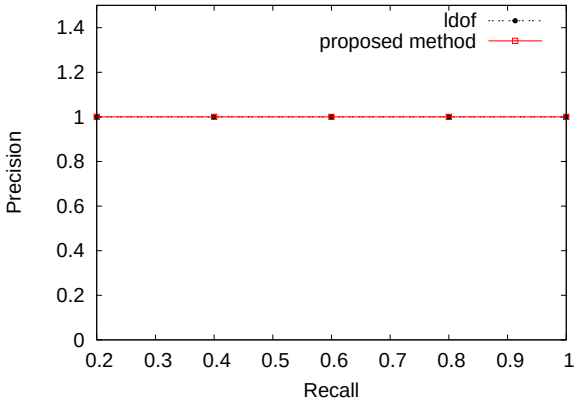

Figure 3.6: Precision Recall Curve for IRIS data set for M-3

3.4.2 Ionosphere Data Set

The percentage of pruning for this data set is presented in Table 3.2 for all the three methods. The data pruning behavior for this data set is also similar to the data pruning pattern for Iris data set. The pruning percentage is less in the method M-2 while there is an increase in the number of clusters. The comparisons of our proposed methods with reference to the existing method are given in Figure 3.7, Figure 3.8 and Figure 3.9 for the methods M-1, M-2 and M-3 respectively. The AUCPR for all the three methods are more than the existing method, so the proposed methods performed in a better way than the existing method though there is a considerable reduction in data points after pruning. The number of attributes of a point and the number of data points in Ionosphere data set is more than that of Iris data set. Therefore, for bigger data set, data pruning helps to

Table 3.2: Pruning ratio for Ionosphere data set

k number of clusters	% of data points pruned for M-1	% of data points pruned for M-2	% of data points pruned for M-3
8	51.91	61.89	65.10
10	47.97	59.57	62.00
15	42.97	41.70	53.61
20	39.14	10.63	49.78
25	20.85	9.36	46.80

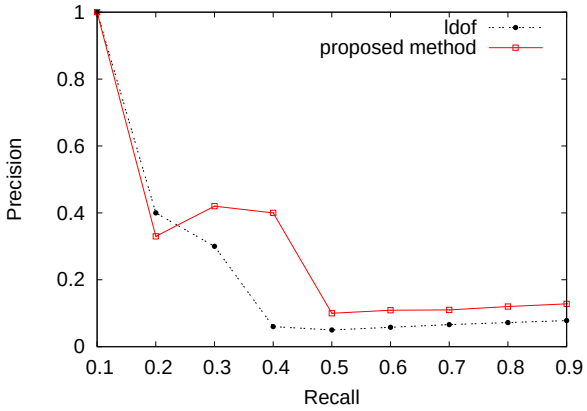

Figure 3.7: Precision Recall Curve for Inosphere data set for M-1

get a better performance for outlier detection.

3.4.3 Medical Diagnosis Data Set

Table 3.3 presents the percentage of pruning for all the three methods by varying the size of the clusters from 8 to 25. We see that in the method M-1 the percentage of pruning is varying from 44% to 63%, where as in the method M-2 the percentage of pruning varies from 12% to 53% and in the method M-3 the percentage of pruning is from 62% to 79%. We observed that when we create 8 clusters the pruning is almost maximum and while considering 25 clusters the pruning percentage is minimum. We also observed the similar pattern that was observed in the two earlier data sets. The PR curve for the methods M-1, M-2 and M-3 are presented in Figure 3.10, Figure 3.11 and Figure 3.12 respectively. The AUCPR is more for all the three proposed methods than the existing method, so for this data set also we observe a better performance for all the three methods than the existing

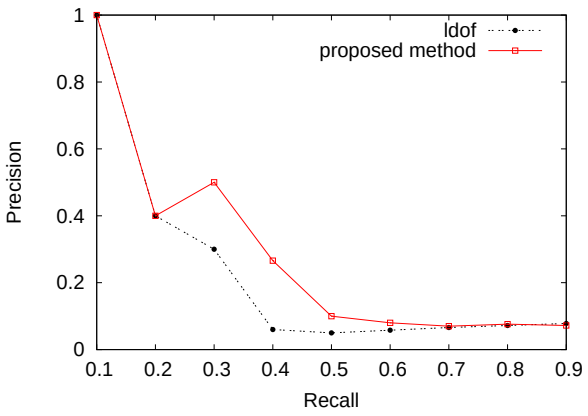

Figure 3.8: Precision Recall Curve for Inosphere data set for M-2

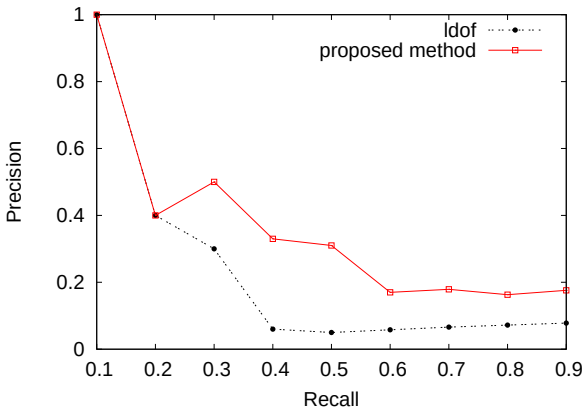

Figure 3.9: Precision Recall Curve for Ionosphere data set for M-3

one.

3.4.4 Shuttle Data Set

The percentage of data objects pruned for the proposed methods are given in Table 3.4. The percentage of pruning is very high for the M-3, and is very low for M-2 where as percentage of pruning for M-1 is medium. The AUCPR for the method M-1 is given in Figure 3.13, which shows that even after pruning 55% to 59% of data points the performance of the proposed methods is better than the existing method. Figure 3.14 shows the performance of the method M-2 and observed that it also performs better than the existing method. In the method M-3 the percentage of pruning is very high in comparison to the other two methods but from Figure 3.15 it is clear that the area covered

Table 3.3: Pruning ratio for WDBC data set

k number of clusters	% of data points pruned for M-1	% of data points pruned for M-2	% of data points pruned for M-3
8	63.76	53.40	79.29
10	59.67	52.04	78.74
15	58.78	47.95	72.47
20	52.76	38.69	71.11
25	44.41	12.26	62.12

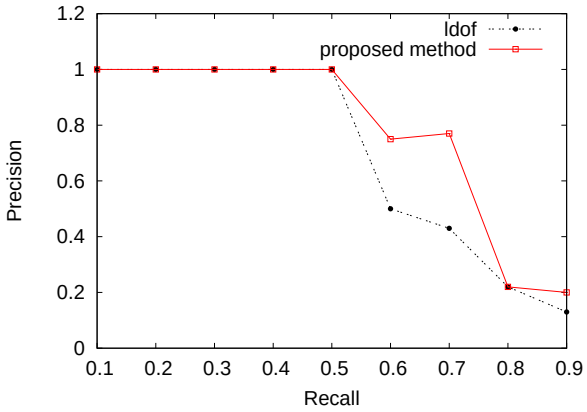

Figure 3.10: Precision Recall Curve for WDBC data set for M-1

under the PR curve is substantially more and so its performance is also good. Also it is clear that for this data set method M-3 performs far better than existing method.

IRIS data set has three classes and out of which two classes are similar and one class is dissimilar, dissimilar class is taken as abnormal point and other class of points as normal points whereas, Ionosphere data set has two classes (good, bad) which is also well separated data set. WDBC data set has two classes (Benign, Malignant) which is also well separated data set. Whereas Shuttle data set has 7 classes out of these classes we randomly select two classes for our experiment. Since these two classes are not well separated, so the precision of both the methods are not good with respect to the shuttle data set.

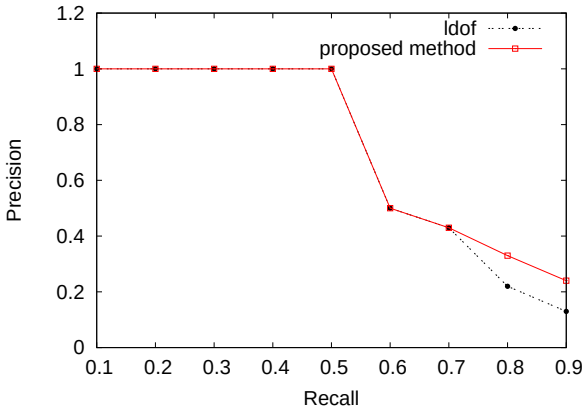

Figure 3.11: Precision Recall Curve for WDBC data set for M-2

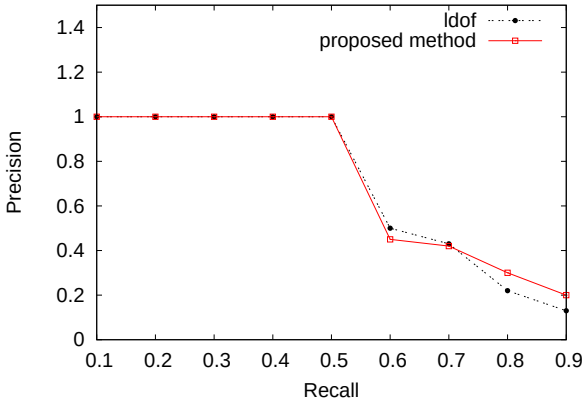

Figure 3.12: Precision Recall Curve for WDBC data set for M-3

3.5 Conclusion

In this chapter we proposed three different heuristics to identify the clusters which may not contain any outliers. Without loss of the precision of the outlier detection methods, we pruned those clusters and use the remaining unpruned points to detect the outliers. We used the parameter *ldof* [21] to identify the outliers in the unpruned data set. We carried out experiments using four different benchmark data sets. We used k-means clustering algorithm in our experiments and perform experiments with different values of k: the number of clusters needs to be generated by the clustering method. It is observed that more number of clusters leads to less data pruning. The results of our experiment indicate that performance of our methods are comparable with the performance of the method

Table 3.4: Pruning ratio for Shuttle data set

k number of clusters	% of data points pruned for M-1	% of data points pruned for M-2	% of data points pruned for M-3
40	59.16	56.79	99.73
80	61.73	50.62	99.58
120	58.38	25.34	99.51
160	56.79	7.98	99.43
200	55.57	0	99.40

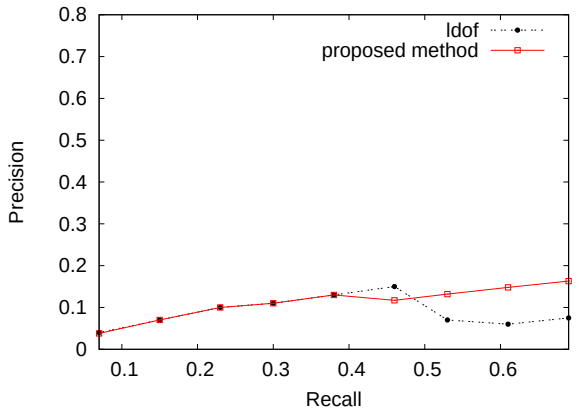

Figure 3.13: Precision Recall Curve for Shuttle data set for M-1

reported by Zhang *et. al.* [21]. The *ldof* parameter is used by Zhang *et. al.* [21] to identify outliers, the same parameter has been used in the proposed methods. It is also observed that for a data set of less number of points, data pruning may not be effective. Our proposed methods performed better than the existing method [21] for data set having more number of data points with more attributes. In the proposed methods, pruning entire cluster may have a chance of pruning few outlier points. So as to avoid any outliers get pruned while pruning the clusters, we propose a new method in the next chapter. Instead of pruning entire cluster we prune some points from each clusters based on the properties of the data points present in the respective clusters.

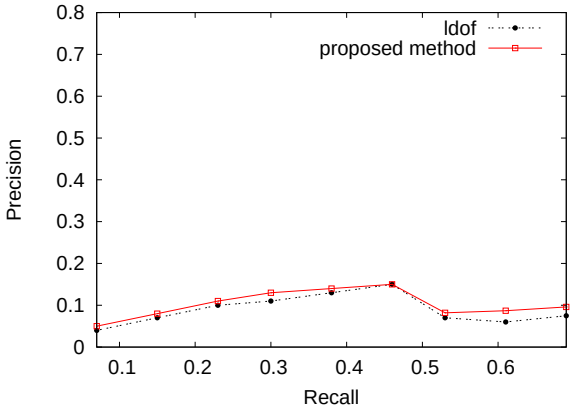

Figure 3.14: Precision Recall Curve for Shuttle data set for M-2

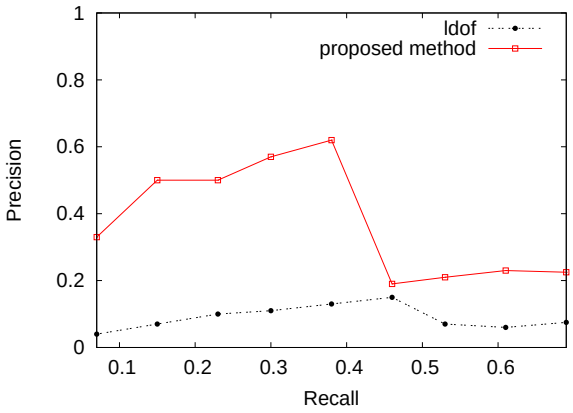

Figure 3.15: Precision Recall Curve for Shuttle data set for M-3

Chapter 4

Point Pruning based Outlier Detection Method

4.1 Introduction

In Chapter 3, the points are pruned as a group of points called clusters. Pruning entire clusters may have a chance of pruning outlier points from the data set. So in order to obtain efficient pruning from every cluster we propose a new method called point pruning based outlier detection. In this method instead of pruning entire cluster we prune only the inliers points from every cluster. Identifying the inlier points is based on the radius of the cluster. Clustering is a grouping of the similar data points, where as separating data points into two classes with the use of radius is an efficient idea for picking up the candidate outliers from each cluster. For each cluster a radius is calculated. The data points which are close to the centriod with respect to the radius are the inlier points. From Figure 4.1 it is observed that there are two concentric circles, one is bold circle and other is the dotted circle. The inner dotted circle is the radius of the cluster, that is the average distance of all the points from the centriod. We also observe from Figure 4.1, few points lie outside the radius and some lie inside the radius. The points lie close to the centriod are possibly the inlier points that are to be removed from all the clusters and the points which are not removed from different clusters are considered as the candidate outliers.

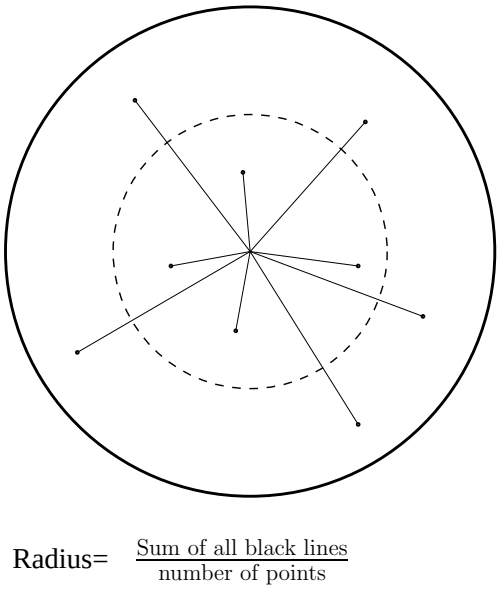

Figure 4.1: Point pruning from each cluster

4.2 Point Pruning Based Outlier Detection

In our propose method, we use k-means algorithm to cluster the data set. k-means algorithm produces a set of clusters. From each of these clusters we identified inlier points and remove the points. Given a cluster of points, centriod is the point which is the center of the cluster. The points which are close to the centriod, have very less probability of being outliers. The radius is calculated for each cluster. The radius of the cluster is the average distance of all points from the centriod. Based on the radius of each cluster, prune the points whose distance from the centriod is less than radius. For a given cluster c, of q δ-dimensional data points $\overrightarrow{p_i}$ where $i = 1, 2, ...q$, the centriod $\overrightarrow{g_c}$, radius R_c are defined as: [59]

$$\overrightarrow{g_c} = \frac{\sum_{i=1}^{q} \overrightarrow{p_i}}{q} \tag{4.1}$$

$$R_c = \sqrt{(\frac{\sum_{i=1}^{q} (\overrightarrow{p_i} - \overrightarrow{g_c})^2}{q})} \tag{4.2}$$

Therefore, for all unpruned points in every cluster we calculate the $ldof$ values and report the top-m points with high $ldof$ values as outliers.

As presented in Section 2.7 the outlier score of point is calculated using the parameter

proposed by Zhang *et. al.* [21]. The proposed method is presented in Algorithm 4.1. We briefly describe the steps.

- **Generating clusters:** Initially, we cluster the entire data set into k clusters using k-means clustering algorithm and calculate radius of each cluster.
- **Clusters having less number of points:** If the clusters contain less number of points than the number of outliers. Point pruning is avoided for all such clusters. As there are very few points in the clusters, which may be the possible outliers.
- **Pruning points inside each cluster:** Calculate distance between point (p_i) and centriod (g_c). If the distance of a point (p_i) is less than the radius of a cluster (R_c), the point is pruned.
- **Computing outlier points:** Calculate $ldof$ values for all the points that are left unpruned in all the clusters. The m points having high $ldof$ values are reported as top-m outliers.

4.3 Computational Analysis

The computational complexity of the $ldof$ method proposed in [21] is N^2, because $ldof$ values are calculated for all the data points. As we are reducing the size of the data set by pruning, the computational complexity of the proposed method is reduced considerably than N^2. Initially for clustering the computational complexity of k-means algorithm is $k*i*N$, where k is the number of clusters to be formed, i is the number of iterations and N is the number of data points. After pruning we left out with few points(set of unpruned points U). Calculating $ldof$ values for all these points, the computation complexity is $(U)^2$. So the total computation of our method is $k*i*N + k*n_p + (w*N)^2$, where w indicates the fraction of data points that we have after pruning, which is around 0.5 and n_p represents average number of points in each cluster. The component $k*i*N$ is for k-means algorithm, $k*n_p$ is for radius calculatin and $(w*N)^2$ is for $ldof$ calculation. Since k and i are small, so the total computations of our method is reduced considerably.

4.4 Experimental Results

The Precision and Recall as discussed in Chapter 3 is also used to evaluate the performance of this method. The experiments are conducted on the same data sets (Iris, Ionosphere,

Algorithm 4.1 Point Pruning Based Outlier Detection Algorithm

Input : $\mathcal{D}$-Data Set, k-number of clusters, i-number of iterations, n-number of outliers

Output: m-points having high $ldof$ value.

Begin

$U \leftarrow \emptyset$; // U set of unpruned data points

Set $\langle C, G \rangle \leftarrow kmeans(k, i, \mathcal{D})$

// C set of k clusters and G set of k centriods

// $C = \{c_1, c_2, ..., c_k\}; G = \{g_1, g_2, ..., g_k\}$

For $\forall c_j \in C$ **do**

  Set $R_{c_j} \leftarrow (\sqrt{\frac{\sum_{i=1}^{m}(p_i - g_j)^2}{|c_j|}})$; // R_{c_j} radius of cluster c_j

end

For $j = 1$ *to* k **do**

  If $|c_j| > n$ **then**

    For $\forall p_i \in c_j$ **do**

      If $d(p_i, g_j) < R_{c_j}$ **then**

        prune(p_i)

      end

      Else

        add p_i to U

      end

    end

  end

  Else

    For $\forall p_i \in c_j$ **do**

      add p_i to U

    end

  end

end

For each point $p_i \in U$ **do**

  calculate $ldof(p_i)$

end

Sort the points in descending order according to their $ldof(p_i)$ values.

First m points with high $ldof(p_i)$ values are the desired outliers.

End

Table 4.1: Pruning ratio for IRIS data set

k number of clusters	% of data points pruned for proposed method
8	46.66
12	32.38
16	11.42
20	5.71

WDBC and Shuttle) that are presented in Chapter 3. These data sets are taken from UCI repository. The number of data points and number of attributes are different for each of the data sets. The performance of the method is measured based on the AUCPR.

4.4.1 Iris Data Set

By varying the number of clusters, we measured the percentage of pruning of the data points that are presented in Table 4.1. It is observed from Table 4.1, the percentage of pruning for the method varies from 46% to 6%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 4.2 shows that AUCPR for the proposed method is same as that of the $ldof$ method, even though we have pruned 46% to 6% of points from the data set. The performance in terms of detecting outliers of the proposed method is same as that of existing method.

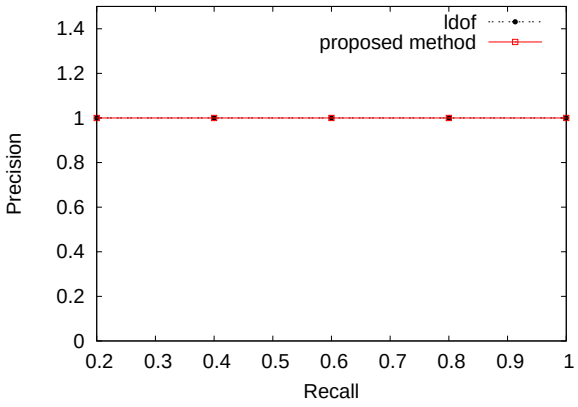

Figure 4.2: Precision Recall Curve for IRIS data set

Table 4.2: Pruning ratio for Ionosphere data set

k number of clusters	% of data points pruned for proposed method
8	61.70
10	60.42
15	50.63
20	45.53
25	40.85

4.4.2 Ionosphere Data Set

By varying the number of clusters, we measured the percentage of pruning of the data points that are presented in Table 4.2. It is observed from Table 4.2, the percentage of pruning for the method varies from 61% to 40%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. The data pruning behavior for this data set is also similar to the data pruning pattern for Iris data set. The pruning percentage is less for Iris data set with the increase in the number of clusters. The comparisons of our proposed method with reference to the existing method are given in Figure 4.3. The AUCPR for the proposed method is at par with the existing method, so the proposed method's performance is similar to the existing method though there is a considerable reduction in data points after pruning. The number of attributes of a point and the number of data points in Ionosphere data set in more than that of Iris data set. Therefore, for bigger data set, data pruning helps to get a better performance for outlier detection.

4.4.3 Medical Diagnosis Data Set

Table 4.3 presents the percentage of pruning for the proposed method by varying the size of the clusters from 8 to 25. It is observed from Table 4.3, the percentage of pruning for the method varies from 55% to 47%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 4.4 shows that AUCPR for the proposed method is same as that of the $ldof$ method, even though we have pruned 55% to 47% of points from the data set.

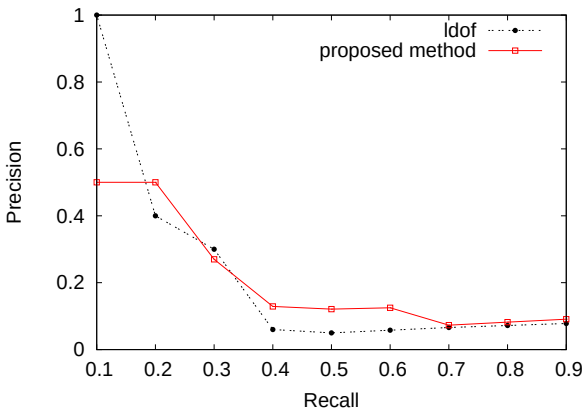

Figure 4.3: Precision Recall Curve for Ionosphere data set

Table 4.3: Pruning ratio for WDBC data set

k number of clusters	% of data points pruned for proposed method
8	55.31
10	53.67
15	50.68
20	49.59
25	47.95

4.4.4 Shuttle Data Set

By varying the size of the clusters from 40 to 200, we measured the percentage of pruning of the data points that are presented in Table 4.4. It is observed from Table 4.4, the percentage of pruning for the method varies from 56% to 53%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 4.5 shows that AUCPR for the proposed method is relatively higher than that of *ldof* method, even though we have pruned 56% to 53% of points from the data set. The low precision of Shuttle data set is explained in Section 3.4.4.

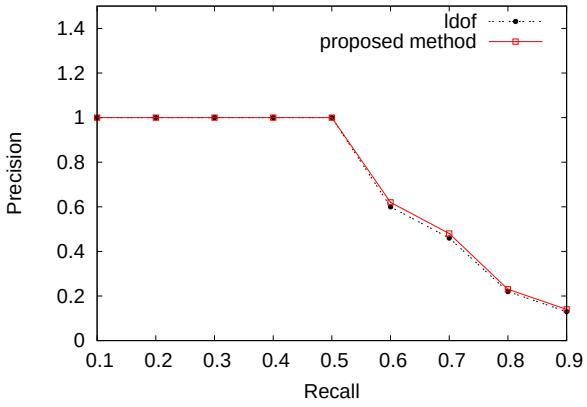

Figure 4.4: Precision Recall Curve for WDBC data set

Table 4.4: Pruning ratio for Shuttle data set

k number of clusters	% of data points pruned for proposed method
40	56.79
80	54.38
120	53.36
160	53.46
200	53.42

4.5 Conclusion

In this chapter, we have proposed an efficient outlier detection method. We used a local distance-based outlier factor to measure the degree to which an object deviates from its neighbor. It is observed that we can prune out on an average 50% of the data points, which are probably not outliers. The precision of detecting outliers is at par with the existing method using $ldof$ proposed by Zhang *et.al.* [21]. In this method, removing inlier data points from the respective cluster is the new idea when compared with the methods proposed in Chapter 3. Pruning data points from data set improves the computation time of detecting outliers, but may effect the precision of the method if more number of points get pruned. Some information about the distribution of data points may lose due to the pruning of entire cluster from data set. It is possible to retain some information about the pruned clusters by including a representative data point for each pruned cluster. To incorporate this idea, we propose a new method in the next chapter that uses both point

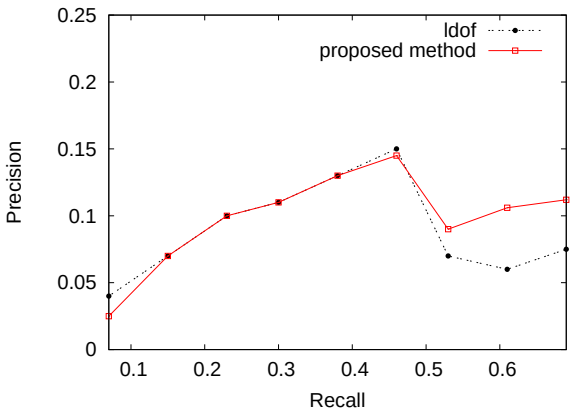

Figure 4.5: Precision Recall Curve for Shuttle data set

pruning and cluster pruning.

Chapter 5

Effective data summarization based pruning for Outlier Detection

5.1 Introduction

In the method proposed in Chapter 4, point pruning from different clusters has improved the percentage of pruning, resulting in the sparse data set as compared with the methods proposed in Chapter 3. Combining cluster pruning and point pruning with data summarization helps in pruning more relevant inliers data points by retaining some information of pruned clusters. Data summarization is a key data mining concept which involves techniques for finding a compact description of a data set. Data summarization of a data set is useful to identify the characteristics of the data set. Simple summarization methods such as tabulating the mean and standard deviations are often applied for data analysis, data visualization and automated report generation [105, 106]. Data summarization techniques can reduce the size and complexity of large multidimensional data sets to more manageable proportions. They can also highlight the relevant aspects of the data more clearly, leading to more coherent visualizations, and also facilitating more accurate and efficient visual analysis [107]. Summarization is performed using various techniques. These techniques are designed for the automated and unsupervised analysis and exploration of raw data, followed by the generation of effective summaries based on the analysis [106]. A summary of a data is essentially a concise version of the original and it is very useful for the data analysis.

The main aim of data summarization is, given an input data set, to provide a more concise view. Summarization has been widely explored in many domains, including transactional databases [108], network data streams [108,109], Intrusion Detection Systems (IDS) [110–112], Point of Sales data (POS) [113] and natural text [114]. The data summarization techniques can give an idea of how the techniques can be applied in many domains. They have proven to be effective in obtaining knowledge out of large data sets, that is easier to interpret. A summary of the input data could be easier and faster to analyze and still obtain similar knowledge.

5.2 Data summarization Based Pruning

This section presents a new outlier detection method for effective pruning of data points. To improve the pruning of data points we propose a method that summarizes the group of data points. The summarization is done on the cluster using the statistics of a cluster. The statistics are presented in the form a single unit called Data Entity (DE). In this proposed scheme, a leaders clustering method is used to obtain set of clusters of the whole data set. The summary of each cluster is made as a single unit called data entity (DE).

A DE is defined as follow.

Definition 5.1. *Data Entity(DE). A data entity corresponding to a leader l is defined as a 6-tuple $DE = \langle s, l, d_\mu, d_\sigma, g, d(l,g)\rangle$, where,*
s= number of followers including the leader l.
d_μ= the average distance from the leader to its followers.
d_σ= standard deviation of the distances from leader to its followers.
g= centriod of the data entity.
$d(l,g)$= the distance between the leader l and the centriod g.

The leaders clustering method is used to cluster the data set, each cluster represents a data entity. The parameters obtained from each data entity are used for further analysis. Figure 5.1 shows the affinity of a DE, the inner concentric circle is the scope of d_σ. If the centriod and the leader lie within the distance of d_σ, then we can understand that the followers of the leader are uniformly distributed around the leader. So we can infer that the chances of having outliers in this DE is very low. Similarly from Figure 5.2, if the centriod and leader are separated more than d_σ distance away from each other, then the points inside the DE are not uniformly distributed and the probability of having outliers

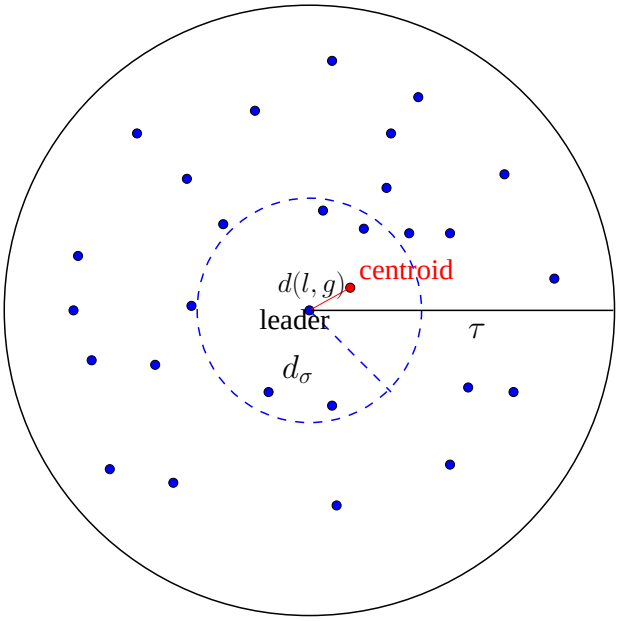

Figure 5.1: Distance between centriod and leader is less than σ; (i.e. $d(l,g) < \sigma$)

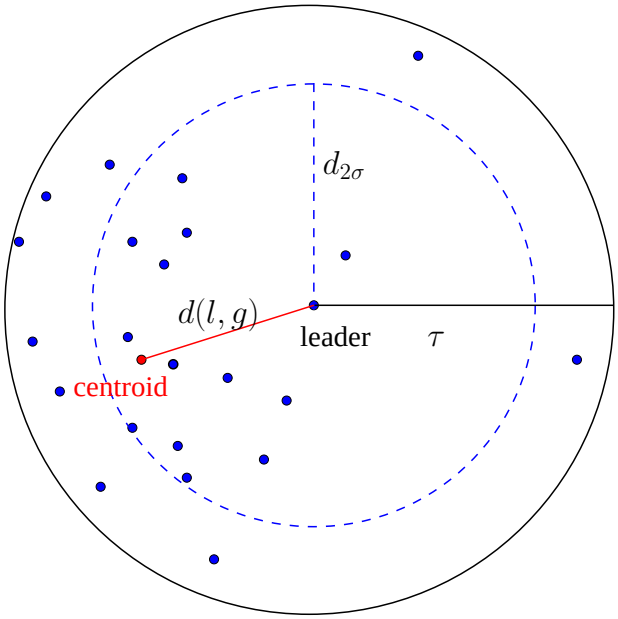

Figure 5.2: Distance between centriod and leader is greater than σ; (i.e. $d(l,g) > \sigma$)

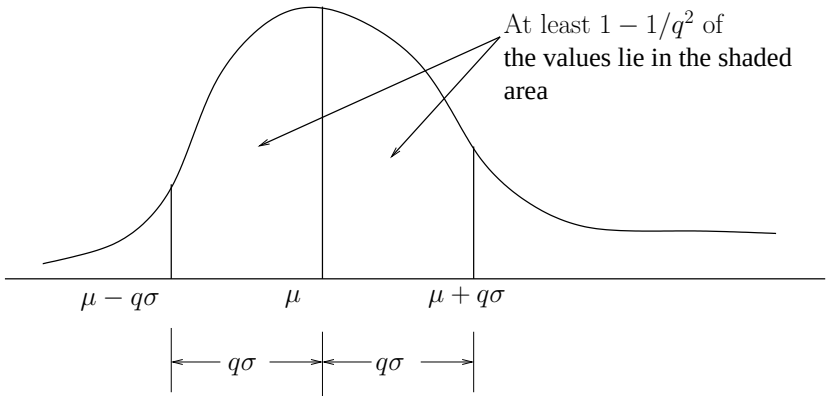

Figure 5.3: Chebyshev Inequality illustration

in these clusters are very high. For pruning inliers points from these clusters/data entities, Chebyshev inequality is used.

Chebyshev's Inequality: For any number q greater than 1, at-least $(1 - 1/q^2)$ of the data values lie within q standard deviations (σ) of the mean(μ) [115, 116].

The chebyshev inequality states that the probability of data points distributed around the mean are q standard deviations away from the mean. The same is explain in Figure 5.3. In other words if the data distribution is unknown, Chebyshev's inequality can be stated as: If x is a random variable with finite mean μ and standard deviation σ, then, for any value $q > 0$.

$$P\{|x - \mu| > q\sigma\} \leq \frac{1}{q^2} \tag{5.1}$$

or

$$P\{|x - \mu| < q\sigma\} \geq (1 - \frac{1}{q^2}) \tag{5.2}$$

In Equation (5.1), x represents a data point, μ is the data mean, σ is the standard deviation and P is the probability distribution of data points. Equation (5.1) states that the probability of data points lie away from μ within $q\sigma$ is $\frac{1}{q^2}$. For example if $(q = 2)$, the probability distribution of the points whose distance is greater than $q\sigma$ from μ is less than $\frac{1}{q^2}$ (i.e. 25%). and from Equation (5.2), the probability distribution of the points whose distance is less than $q\sigma$ from μ is greater than $1 - \frac{1}{q^2}$ (i.e. 75%). So for any distribution, using chebyshev inequality [115, 116] we can shown that at least 75%(i.e. $\frac{3}{4}$ fraction of points) of the data would fall below two standard deviations $(q = 2)$ from the mean and 25% (i.e. $\frac{1}{4}$ fraction of points) of the data fall above the two standard deviations from the

mean. Using chebyshev inequality, we obtain 25% of points from each data entity (i.e. the data points $2 * d_\sigma$ away from the leader).

In this data summarization based pruning method, the data set is clustered into a set of clusters. Few clusters are pruned based on the parameters/statistics as defined in DE. For the remaining unpruned clusters chebyshev inequality is applied to prune some points from these clusters. Finally based on outlier score of the unpruned points declare the top-m outliers.

For pruning clusters and points from the cluster, the distance between leader and centriod is the main important parameter considered. To prune clusters, distance between leader and centriod $d(l, g)$ is compared with the standard deviation of the distances of the points present in the cluster. If the distance $d(l, g) < d_\sigma$ prune the entire set of followers/points of the leader l, because if distance between leader and centriod is less than the standard deviation of the cluster, then the probability of points distributed around leader and centriod are uniformly distributed. Even the centriod and leader are relatively very close and all the points are equiv distance away from leader and centriod i.e. τ distance. So we prune the entire cluster which is shown in Figure 5.1.

Similarly if the distance between leader and centriod is greater than one standard deviation, then from chebyshev inequality Equation (5.2), we can say that leader and centriod are not close to each other. So pruning the points which are below two standard deviation from the leader. Henceforth capturing the 25% of points lying away from the leader. From Figure 5.2 we see that centriod is more than one standard deviation distance away from the leader. So we pruned all the points which lie below two standard deviations.

So using chebyshev's inequality, centriod, mean, standard deviation and $ldof$ values a new method of detection of outlier is presented here. This method is divided into 3 steps; 1) clustering 2) data summarization with pruning and 3) outlier score. In clustering step we use well known leaders-followers clustering algorithm to find out leaders-followers. In pruning step few clusters are pruned based on distance between leader and centriod, and pruning few points/followers from remaining unpruned clusters based on the chebyshev's inequality. For the remaining unpruned points we calculate $ldof$ and declare the top-m points as outliers in the outlier score step. Algorithm 5.1 describes how we have detected the outliers using data summarization method.

We briefly describe the steps.

- **Generating clusters:** Initially, we cluster the entire data set into different clusters using leaders clustering algorithm. Each cluster is summarized as a data entity.

- **Cardinality:** If the cardinality of a cluster is less than the number of outliers, we use all the points of the cluster as candidate points.
- **Pruning some clusters/points:** Pruning (removing possible inliers), is based on two important parameters
  - If the distance between leader and centriod of the cluster $d(l, g)$ is smaller than one standard deviation of the cluster, add only leader of the cluster to the candidate points.
  - If the distance between leader and centriod of the cluster $d(l, g)$ is greater than one standard deviation of the cluster, add all the points whose distance from the leader is greater than two standard deviations.
- **Computing outlier points:** Calculate $ldof$ values for all the candidate points. The points having top-m $ldof$ values are outliers.

5.3 Computational Analysis

The computational complexity of the $ldof$ method proposed in [21] is N^2, because $ldof$ values are calculated for all the data points. As we are reducing the size of the data set by pruning, the computational complexity of the proposed method is reduced considerably than N^2. Initially for clustering the computational complexity of leaders clustering is $O(pN)$, where $p = \mid \mathcal{L} \mid$, and N is the number of data points. After pruning we left out with few points (set of unpruned points U). Calculating $ldof$ values for all these points, the computation complexity is $(U)^2$. So the total computation of our method is $p * N + (w * N)^2$, where w indicates the fraction of data points that we have after pruning, which is around 0.2. Since p is small, so the total computations of our methods are reduced considerably.

5.4 Experimental Results

The experiments are conducted on the same data sets as described in Chapter 3 and the same performance metrics are used to show the performance of our method. The number of cluster pruned, number of points pruned and total percentage of pruning are presented. AUCPR that described in the Chapter 3 is used to compare the performance of our proposed method with reference to the method proposed in [21].

Algorithm 5.1 Data Summarization Based Pruning for Outlier Detection

Input : $\mathcal{D}$-DataSet, τ-leaders threshold distance, n-number of outliers.

Output: m-points having high $ldof$ value.

Begin

$U \leftarrow \emptyset$; // U set of unpruned data points

Set $\langle C, \mathcal{L}, G \rangle \leftarrow leaders(\mathcal{D}, \tau)$;

// C set of clusters, $\mathcal{L}$ set of leaders and G set of centriods

// $C = \{c_1, c_2, ..., c_k\}; \mathcal{L} = \{l_1, l_2, ..., l_k\}; G = \{g_1, g_2, ..., g_k\}$

For $\forall c_i \in C$ **do**

  If $|c_i| < n$ **then**

    add points of c_i to U

  end

  // d_{σ_i} = standard deviation of the distances of a cluster c_i

  Else

    If $d(l_i, g_i) < d_{\sigma_i}$ **then**

      add l_i into U

    end

    Else

      For $\forall p_i \in c_i$ **do**

        If $d(p_i, l_i) > 2d_{\sigma_i}$ **then**

          add p_i to U

        end

      end

    end

  end

end

For $\forall p_i \in U$ **do**

  calculate $ldof(p_i)$

end

Sort the points in descending order according to their $ldof(p_i)$ values.

First m points with high $ldof(p_i)$ values are the desired outliers.

End

5.4.1 Iris Data Set

By varying the τ value from 0.20 to 0.45 we measured the percentage of pruning of the data points that are presented in Table 5.1. It is observed from Table 5.1, the percentage of pruning for the method varies from 85% to 38%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 5.4 shows that AUCPR for the proposed method is same as that of the $ldof$ method, even though we have pruned 85% to 38% of points from the data set.

Table 5.1: Precision and Pruning ratio for IRIS data set

τ	k-number of clusters	clusters pruned	number pruned points		candidate points	% of pruning
			$d(l,g) < \sigma$	$d(p,l) < 2\sigma$		
0.20	90	0	0	90	15	85.71
0.25	78	0	0	78	27	74.28
0.30	67	1	3	64	38	63.80
0.35	56	1	5	51	49	53.33
0.40	49	0	0	49	56	46.66
0.45	40	1	5	35	65	38.09

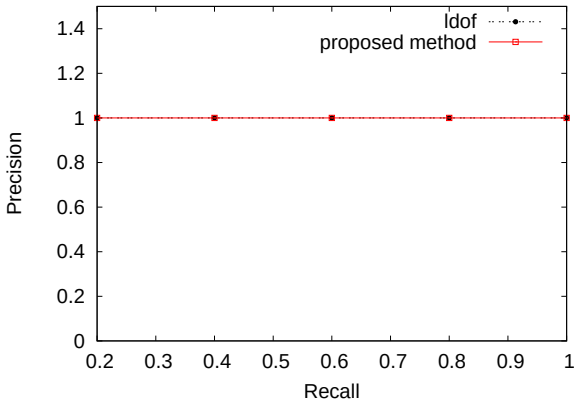

Figure 5.4: Precision Recall Curve for IRIS data set

5.4.2 Ionosphere Data Set

By varying the τ value from 1.0 to 3.0 we measured the percentage of pruning of the data points that are presented in Table 5.2. It is observed from Table 5.2, the percentage of pruning for the method varies from 89% to 61%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 5.5 shows that AUCPR for the proposed method is same as that of the *ldof* method, even though we have pruned 89% to 61% of points from the data set.

Table 5.2: Precision and Pruning ratio for Ionosphere data set

τ	k-number of clusters	clusters pruned	number pruned points		candidate points	% of pruning
			$d(l,g) < \sigma$	$d(p,l) < 2\sigma$		
1.0	90	0	0	145	90	61.70
1.5	49	3	24	162	49	79.14
2.0	30	3	41	160	34	85.53
2.5	17	2	45	165	25	89.36
3.0	13	1	18	185	32	86.38

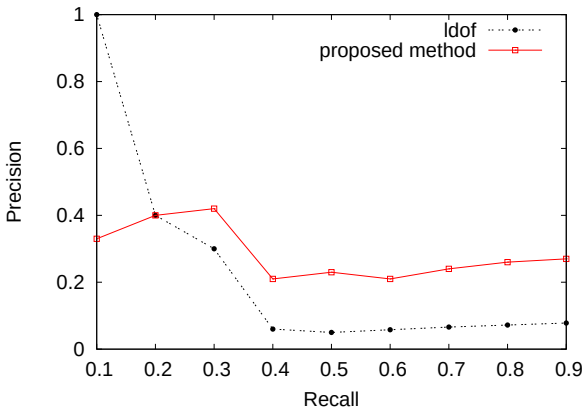

Figure 5.5: Precision Recall Curve for Ionosphere data set

5.4.3 Medical Diagnosis Data Set

By varying the τ value from 25 to 45 we measured the percentage of pruning of the data points that are presented in Table 5.3. It is observed from Table 5.3, the percentage of pruning for the method varies from 79% to 66%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 5.6 shows that AUCPR for the proposed method is same as that of the *ldof* method, even though we have pruned 79% to 66% of points from the data set.

Table 5.3: Precision and Pruning ratio for WDBC data set

τ	k-number of clusters	clusters pruned	number pruned points		candidate points	% of pruning
			$d(l,g) < \sigma$	$d(p,l) < 2\sigma$		
25	114	8	43	202	122	66.75
30	87	6	42	225	100	72.75
35	71	3	21	248	98	73.29
40	56	5	64	217	86	76.56
45	48	2	32	260	75	79.56

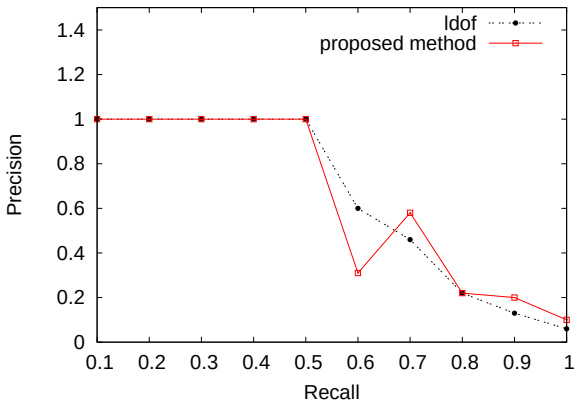

Figure 5.6: Precision Recall Curve for WDBC data set

Table 5.4: Precision and Pruning ratio for Shuttle data set

τ	k-number of clusters	clusters pruned	number pruned points		candidate points	% of pruning
			$d(l,g) < \sigma$	$d(p,l) < 2\sigma$		
10	390	27	1346	9194	951	91.72
20	111	8	1315	9325	851	92.59
30	76	4	657	9563	1271	88.93
40	58	5	2003	8673	815	92.90
50	46	3	1001	9623	867	92.45

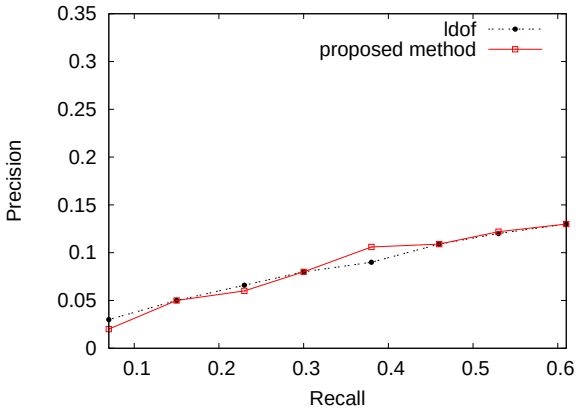

Figure 5.7: Precision Recall Curve for Shuttle data set

5.4.4 Shuttle Data Set

By varying the τ value from 10 to 50 we measured the percentage of pruning of the data points that are presented in Table 5.4. It is observed from Table 5.4, the percentage of pruning for the method varies from 92% to 88%. Even though we prune these many percentage of points, we could able to find out outliers from the data set. Finally the Precision-Recall Curve is used to compare the performance of the proposed method with the existing method proposed by Zhang, *et. al.* [21]. Figure 5.7 shows that AUCPR for the proposed method is same as that of the *ldof* method, even though we have pruned 92% to 88% of points from the data set. The low precision of Shuttle data set is explained in Section 3.4.4.

5.5 Conclusion

In this chapter we proposed a new data summarization method to identify the clusters which may or may not contain any outliers. Initially we grouped the data using leaders clustering method, and then we estimated the spread of data in a cluster based on the distance between the leader and the centriod of the cluster. This method is a combination of cluster pruning proposed in Chapter 3 and point pruning proposed in Chapter 4. Without loss of the precision of the outlier detection methods, we pruned those clusters which may not contain outliers, but to retain some information of the pruned cluster we include a representative data points for such clusters. The clusters which may contain outliers are processed to separate possible inlier points from outlier points and the probable inliers are removed from the clusters. The remaining unpruned points are used to detect the outliers. We used the parameter *ldof* [21] to identify the outliers in the unpruned data set. We carried out experiments using four different benchmark data sets. The results of our experiments indicate that performances of our methods are comparable with the performance of the method reported by Zhang *et. al.* [21]. Till now we have proposed methods to reduce the computation time while detecting outliers by removing some of the data points from the data set. The distance computations also take more time if the dimensions/attributes of data points are more. It is also an interesting issue to investigate whether computation time can be reduced or not while detecting outliers by reducing the number of attributes in the data set. In the next chapter we introduce a new method for outlier detection that involves attribute pruning.

Chapter 6

Correlation based Pruning Method for Outlier Detection

6.1 Introduction

In the previous chapters the proposed methods are based on the objects pruning (pruning in the form of set of objects (cluster) or data object itself) from the data set. That is most of the outlier detection algorithms attempt to detect outliers by computing the distances in full dimensional space [96]. Even the methods proposed in the previous chapters attempt to find out outliers in the full dimensional space. However, in very high dimensional spaces, the data are very redundant and the concept of similarity may not be meaningful. Examining the behavior of the data in sub-spaces, it is possible to develop more effective algorithms and similarity search in high dimensional spaces [117, 118]. It can be shown that this is also true for the problem of outlier detection, since in many applications only the subset of attributes is very useful for detecting anomalous behavior. In addition, when significant number of features in a data set is considered noisy [119], finding outliers in all dimensions typically do not result in effective detection of outliers, while at the same time it is difficult to identify a few relevant dimensions where the outliers may be observed.

To select the features that are relevant for any detection technique is an important problem. In general, a feature is good if it is relevant, but is not redundant to any of the other relevant features [120]. So the correlation between two variables/features is used as a goodness measure. Therefore a feature is good if it is highly correlated to the set of attributes/features but not highly correlated to any of the other features.

For example the euclidean distance calculation between pair of data point x and y

in δ dimension is given in Equation (6.1). In order calculate the distance between two points we have to perform addition, subtraction, square and square root. Selecting few attributes and calculating distance between two points on these attributes reduces the number of additions, subtractions and square operations.

$$d(x,y) = \sqrt{\sum_{i=1}^{\delta} (x_i - y_i)^2} \tag{6.1}$$

Therefore for distance computation in δ dimensional has high computation than calculating same distance computation in smaller dimension.

So, we propose a method to select small set of attributes using correlation coefficient. Correlation coefficient is used to identify the correlation factor between the pair of attributes.

6.2 Correlation Based Method

In this section, we discuss how to select the features. The attribute is selected from a set of attributes having similar correlation value. Correlation summarizes the relationship between two variables in a single number called the correlation coefficient. There is one classical well known measure called linear correlation coefficient (also called as Pearson correlation coefficient [64]). It is a symmetrical measure for two variables [120]. Other measures in this category are basically variations of the correlation formula, such as least square regression error and maximal information compression index [121]. Correlation describes the relationship between two different variables/features [120]. In statistics community it is called as correlation coefficient. The correlation coefficient is usually given the symbol r that describes direction (positive or negative) and degree (strength) of relationship between two variables. A correlation coefficient can vary from (-1 , +1). The relationship between the two variables X and Y is explained as shown in Figure 6.1, in sub figure (a) the two variables X and Y are directly proportional to each other as X increases Y also increases, in (b) X and Y are inversely proportional as X increases Y decreases, in (c) X and Y are highly correlated, they are proportional to each other, in (d) the two variables are not proportional to each other, (e) and (f) are not at all correlated to each other there is no relationship between the two variables.

The correlation between all pair of features/attributes is calculated based on Equation (6.2), where X and Y are two attributes, $\bar{x}$ is the mean of X, and $\bar{y}$ is the mean of Y.

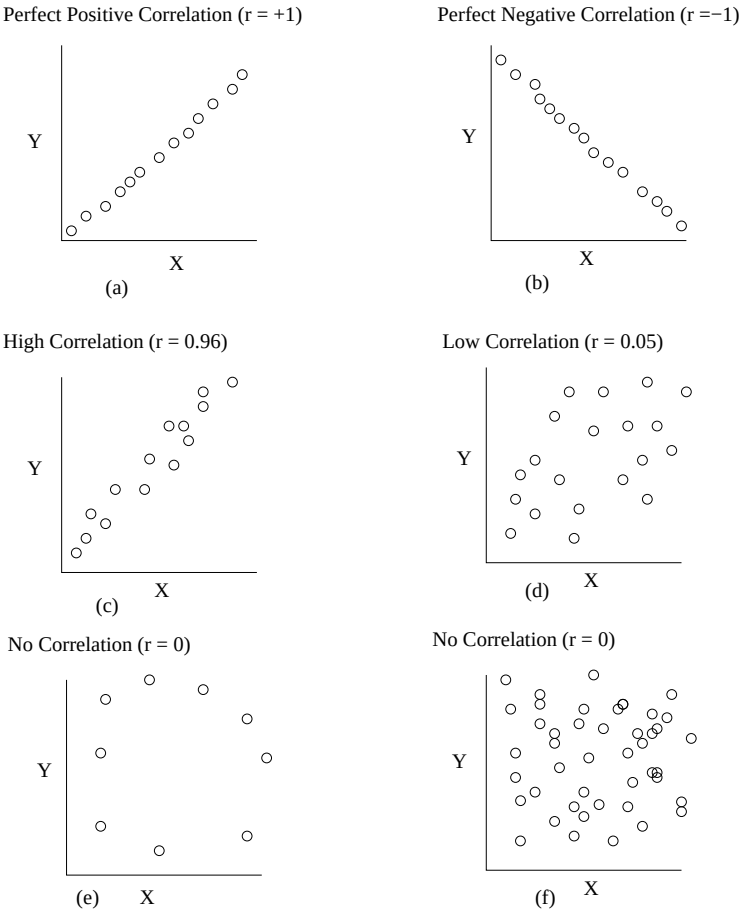

Figure 6.1: Pearson correlated coefficient. (a) perfect positive correlation (r = +1), (b) perfect negative correlation (r = -1), (c) high correlation (r = 0.96), (d) low correlation (r = 0.05), (e) no correlation (r = 0), and (f) no correlation (r = 0).

$$r := \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{6.2}$$

There are several benefits of choosing linear correlation as a feature selection measure. It reduces redundancy among selected features [122]. However, linear correlation measures may not be able to capture correlations that are not linear in nature. Another limitation is that the calculation requires all features must be numerical values.

The proposed method is divided into two steps. First step, correlation step, finding the subset of attributes having similar correlation value. In the second step, outlier score is calculated with only the selected attributes. From the correlated set of features, one feature is selected as a representative feature and is used to determine the outliers from the

data set. In outlier score step, outlier score is calculated for all the data points with only the representative features that are selected from the correlation coefficient. Removing some of these features help to reduce the computational time when compared with the existing $ldof$ method in full attributes. Algorithm 6.1 describes our method.

We briefly describe the steps needed to detect outliers.

- **Correlation coefficient calculation:** Initially, select few number of objects/instances (user defined), to calculate pair wise correlation between all the pair of attribute.
- **Group the set of attributes based on the correlation value:** Based on the required correlation value (threshold value) group the set of attributes.
- **Select the representative attribute:** Select one attribute from each correlated set of attributes. The one selected is used as a representative attribute of the correlated set.
- **Computing outlier points:** Calculate $ldof$ values for all the points using only the selected representative attributes. The top-m points with high $ldof$ values are reported as outliers.

6.3 Computational Analysis

The computational complexity of the $ldof$ method proposed by Zhang *et. al.* [21] in full δ dimensions is $O(\delta N^2)$, because $ldof$ values are calculated for all the data points in δ dimensions. As we are reducing the number of attributes by selecting few attributes based on correlation, the computational time of the proposed method is reduced considerably. Initially calculating correlation coefficient for all pair of attributes based on very few randomly selected data points has very less computational time.

After removing some attributes, the few selected features (set of unpruned features U_A) are used for calculating $ldof$ values. All the points with only U_A attributes have less computational time than calculating $ldof$ in full dimensions/attributes.

6.4 Experimental Results

The performance of an algorithm is evaluated based on the number of outliers detected by the algorithm. We use the same performance metrics described in Chapter 3 to

Algorithm 6.1 Attribute Pruning Based Outlier Detection Algorithm

Input : $\mathcal{D}$-data set, t_c : threshold correlation value, n-number of outliers

Output: m-points having high $ldof$ value.

Begin

$U_A \leftarrow \emptyset$; // U_A : Set of selected attributes

select $\mathcal{T} \subset \mathcal{D}$; // $\mathcal{T}$: contains n random points

// f_p Pearson Correlation Coefficient

// $A = \{a_1, a_2, ..., a_\delta\}$; δ-number of attributes

For $\forall(a_i, a_j)$, $a_i, a_j \in A$, *with elements of* $\mathcal{T}$ **do**

  $M[i, j] = f_p(a_i, a_j)$; // M Correlation Matrix

end

Construct subsets $S_1, S_2, ..., S_l$ of A,

such that for all pair of attributes $(a_i, a_j) \in S_i$ is $f_p(a_i, a_j) > t_c$

For $\forall S_l \subset A$ **do**

  add one attribute $a_1 \in S_l$ to U_A

end

For $\forall p_i \in \mathcal{D}$ *using attributes of* U_A **do**

  calculate $ldof(p_i)$

end

Sort the points in descending order according to their $ldof(p_i)$ values.

First m points with high $ldof(p_i)$ values are the desired outliers.

End

Table 6.1: Computation time for Libras Movement data set

Threshold (t_c) correlation coefficient	Percentage of attributes pruned	Computation time (secs) for $ldof$	
		in full δ	in reduced attributes
0.9	75.55	0.257717	0.2063 + 0.013644
0.8	83.33	0.257717	0.2098 + 0.012765
0.7	86.66	0.257717	0.2114 + 0.012026

evaluate the performance. However the data sets use in this chapter are different from previous chapters. The different data sets are used in this method because, the data set must contain more number of attributes. The three real data sets are taken from UCI repository [104]. The correlation coefficient is only calculated for all pair of attributes with a sample of 10 objects from each data set. A subset of similar correlation values are formed based on the threshold correlation coefficient (t_c). In all our experiments we consider different values for t_c and use threshold as 0.9, 0.8 and 0.7. We present the performance results in terms of percentage of pruning, computation time and AUCPR. The computational time is presented in seconds, i.e. how much time the method has taken for calculating $ldof$ score for all the points (without attribute pruning and with attribute pruning). The computational time for calculating correlation coefficient is also considered in our experiment. The total time requirement for our method is time for calculating the correlation coefficient and the time for calculating $ldof$ score. Computational time for calculating correlation coefficient for all pair of attributes is comparatively less because we use a randomly selected small sample of data points from the data set. In our experiments the sample of data points selected for calculating correlation coefficient is 10 (i.e. equal to number of outliers).

6.4.1 Libras Movement Data Set

In this experiment, we use the data set LIBRAS, acronym of the Portuguese name LIngua BRAsileira de Sinais, is the official Brazilian sign language. The data set (movement libras) contains 15 classes of 24 instances each, where each class references to a hand movement type in LIBRAS. The hand movement is represented as a bi-dimensional curve performed by the hand in a period of time. The curves were obtained from videos of hand movements, with the Libras performance from 4 different people, during 2 sessions. Each video corresponds to only one hand movement and has about 7 seconds. Each video corresponds to a function F in a functions space which is the continual version of the

input data set. In the video pre-processing, a time normalization is carried out selecting 45 frames from each video, in according to an uniform distribution. In each frame, the centriod pixels of the segmented objects (the hand) are found, which compose the discrete version of the curve F with 45 points. All curves are normalized in the unitary space. In order to prepare these movements to be analyzed by algorithms, they have carried out a mapping operation, that is, each curve F is mapped in a representation with 90 features, with representing the coordinates of movement. The data set contains 360 (24 in each of fifteen classes) instances, each with 91 attributes (90 numeric (double) and 1 for the class (integer)). We regard the class (integer/class = 15) as outlier, added only 10 points, and remaining classes as inliers.

The attribute pruning is presented in terms of percentage of pruning for different correlation values is shown in Table 6.1. The performance in terms of Precision and Recall is shown in Figure 6.2. The proposed method performs better than the *ldof* method. Even though 75% to 86 % of attributes are pruned, the AUCPR is better than the method without attribute pruning.

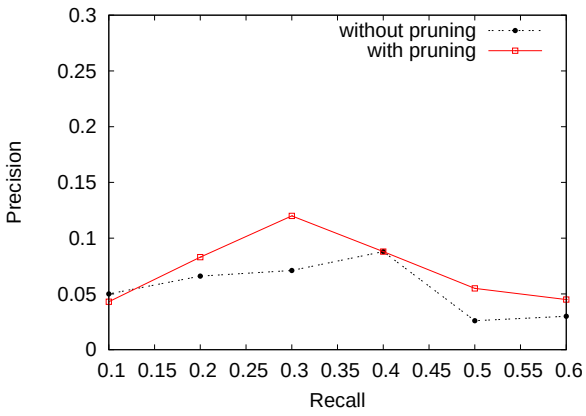

Figure 6.2: Precision Recall Curve for Libras Movement data set

6.4.2 Musk Data Set

This data set describes a set of 102 molecules of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks. The goal is to predict whether new molecules are musks or non-musks. However, the 166 features that describe these molecules depend upon the exact shape, or conformation, of the molecule. Because bonds can rotate, a single molecule can adopt many different shapes. The data

Table 6.2: Computation time for Musk data set

Threshold (t_c) correlation coefficient	Percentage of attributes pruned	Computation time (secs) for $ldof$	
		in full δ	in reduced attributes
0.9	72.89	6.604828	0.6714 + 3.628507
0.8	79.51	6.604828	0.6798 + 3.405509
0.7	82.53	6.604828	0.6859 + 3.308946

set is generated by all the low-energy conformations of the molecules were generated to produce 6,598 conformations.

The data set describes whether a molecules is a musk or non-musk. It contains 6598 (with two classes, musk and nonmusk) instances, each with 167 attributes (166 numeric (double) and 1 for the class (1,0)). We consider the class-0 points as outliers and the class-1 data points as inliers. Out of 6598 objects we separated class-0 and class-1 instances. We added 10 outliers that is class-0 instances into 5581 class-1 instances.

The performance of the method is shown using AUCPR. It is observed from Figure 6.3 after pruning 72% to 82% of attributes, the AUCPR for the method is also better than the method without attribute pruning. The percentage if pruning is also good for different values of correlation as presented in Table 6.2. The computation time as presented in Table 6.2 is less when compared to computation time in full dimension.

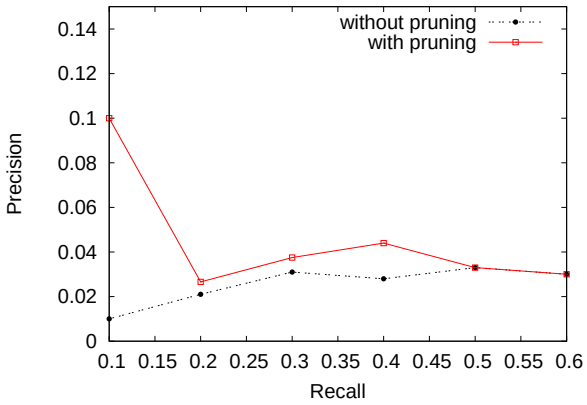

Figure 6.3: Precision Recall Curve for Musk data set

Table 6.3: Computation time for Isolet data set

Threshold (t_c) correlation coefficient	Percentage of attributes pruned	Computation time (secs) for $ldof$	
		in full δ	in reduced attributes
0.9	53.16	31.618082	8.7038 + 17.348010
0.8	82.01	31.618082	8.7350 + 9.957266
0.7	95.62	31.618082	8.7499 + 6.791235

6.4.3 Isolet Data Set

The data set ISOLET predicts which letter-name that was spoken. The data set contains 7797 data objects with 617 attributes. There 26 classes of data, each class has 300 data points. As the data is very similar, we separated one class (class-1) and multiplied the class of points with a value (1.6) to make that class of points as outliers. We selected 10 points from this class and added into other class of points as outliers. We performed the experiments to find out the outlier points from the data set with attribute pruning and without pruning.

In Table 6.3 we presented the percentage pruning of attributes for different values of correlation coefficient. There is a wide range of pruning percentage for different values of correlation values. Even the computation time without pruning and with pruning also have wide difference in seconds as shown in Table 6.3. Finally the performance of the method in terms of AUCPR is very good as shown in Figure 6.4. This is because of the small factor (constant value) multiplied to a class of data points to make them as outliers.

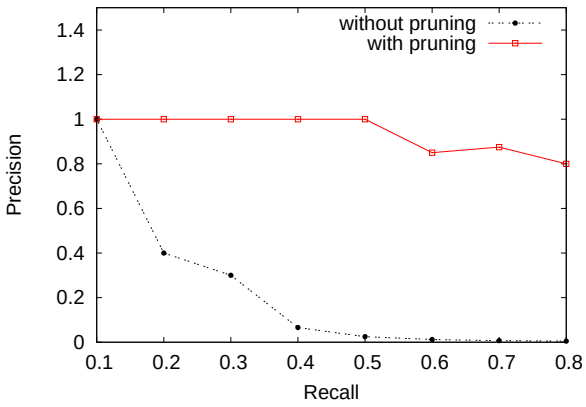

Figure 6.4: Precision Recall Curve for Isolet data set

The number of attributes present in data sets Libras, Musk and Isolet are 90, 166 and

617 respectively. From the experiments it is observed that for more number of attributes, there is a considerable reduction in time for detecting outliers with reference to the method of outlier detection without attribute pruning by Zhang *et. al.* [21]. So, attribute pruning method is useful for big data set with large number of attributes.

6.5 Conclusion

In this chapter, we proposed an outlier detection method based on correlation. Initially correlation coefficient between all pair of attributes is calculated and based on correlation value, subsets of attributes are formed. All the attributes in the subset are similar based on correlation value. Selecting one attribute from the set of correlated attributes provides a representative attribute for the set and it reduces the number of attributes for calculating $ldof$ value. Using $ldof$ parameter the outlier score of each point is calculated considering only the selected/representative attributes. The computational time is reduced as we are reducing the size of the attributes. The precision of detecting outliers of our method is far better than the method without pruning [21] though we pruned out some redundant features.

Chapter 7

Conclusion and Future Works

The work presented in this thesis puts forward algorithms to detect outliers from a data set. The main direction of this work is for reducing the computations while detecting outliers from a data set. The reduction in computations can be achieved for outlier detection from a big data set if the data set is reduced by pruning some of the data points for which chances of being outlier is less. Also for data set which consists of large number of attributes, the computation can be reduced by selecting some attributes that are closely related to other attributes. The main contributions of the thesis are as follows:

7.1 Contributions

Cluster Based Pruning In Chapter 3, we proposed three cluster based pruning methods. In the first method, we used the behaviors of the centriods of the clusters. The high $ldof$ value of a centriod indicates that it is away from other clusters and chances of having outliers are more. We kept such type of clusters for outlier detection and pruned the other clusters. The second and third methods are an improvement of the first method. In the first method we pruned half of the clusters, but in other two methods the number of clusters that are to be pruned were decided depending on the behaviors of the clusters. In the second method the radius of clusters were used as a deciding factor to pruned clusters; but if we divided the data set into a large number of clusters it had a affect on the pruning percentage. To overcome this difficulty, in third method we used the distance of cluster's centriod to the centriod of the complete data set as a deciding factor whether to prune a cluster or not. The cluster pruning helped us to remove around 60% of points from the data set, even then we achieved similar precision for the proposed methods with

respect to the existing method [21].

Point Based Pruning The second contribution of this work is based on point pruning. Instead of pruning the entire cluster, in this method we pruned some points from each clusters. The points that are away from the cluster's center are deviating from normal points and these are the probable outliers. We have used the radius of the cluster as deciding factor. The points that lie inside the radius are pruned. The experiment results showed that we got a considerable reduction in the data set and eventually reduced the computation for calculating the $ldof$ value.

Data Summarization Based Pruning The third contribution of this work is a new data summarization scheme. It is proposed to prune both the clusters and the points depending on the behaviors of the clusters. The clusters are represented as a *Data Entity*. The *Data Entity* summarizes the properties of the cluster. Depending on the properties of *Data Entity*, either we have pruned the entire cluster or pruned some data points from the cluster. When we pruned the entire clusters, then a representative data point is included in the data set. Experiment results showed that this method is also comparable with the existing method [21].

Attribute Based Pruning This contribution is to handle data set with large number of attributes. The Pearson correlation factor has been used to find the set of correlated attributes. From the set of correlated attributes, only one attribute has been considered for further use. This selected attribute considered as the representative attribute for the set of correlated attributes. So the number of attributes reduced considerably and eventually reduced the distance computation while calculating the $ldof$ value.

7.2 Discussions

The main objective of this work is to find the outliers from a given data set using less number of computations. The basic computations involved in outlier detection is distance computations between data points. Our main emphasis is to reduce the distance computations. So, if we can remove the data points from the data set, which are not outliers, then it leads to the reduction in distance computations. We used clustering techniques to identify the data points that can be pruned. The formation of clusters depends on the behaviors of the data set and the distributions of data points in the data

set. In our work it is observed that a particular method is not effective for all kind of data sets.

For a dense data set it has been observed that the formation of most of the clusters are also dense in nature. Possibilities of having outliers in a dense cluster is less because most of the data points are of similar nature and these are closely related. So, without effecting the precision of the outlier detection methods, these dense clusters can be pruned. In our cluster pruning method, the entire cluster is pruned if the cluster is dense in nature. Therefore, this cluster pruning method is more effective for dense data set. On the other hand if the nature of formed clusters is relatively sparse in nature. In such situations pruning the entire clusters is not an effective decision. In our work we proposed a method based on point pruning. On the basis of the properties of the data points of a clusters, we pruned some of the points.

In case of cluster pruning, it has been observed that for some data sets the pruning percentage is very high and as a result some of the outliers got pruned. In cluster pruning the entire cluster has been pruned and we do not have any information about those point during outlier detection phase. To overcome these difficulties, we proposed a new method based on data summarization. In this method we applied both cluster pruning and point pruning methods. Each cluster is represented by *Data Entity*. The properties of data entity indicate whether we should go for data pruning or cluster pruning. If the decision is for a cluster pruning, then the information of that data entity is kept in the data set as a representative point.

Finally we considered the data set with large number of attributes. Due to the presence of large number of attributes, the distance computations are very high. To reduce the number of distance computations, we proposed attribute pruning in our last method. We used Pearson Correlation Coefficient to find the correlated attributes of the data set. We considered one attributes from the set of correlated attributes. Due to the reduction in attributes in the data set, the distance computations reduced drastically. Finally we calculate the $ldof$ value of each point using reduced attributes set to identify the outliers.

We performed several experiments using bench mark data sets and found that our methods performed at per with the existing method [21]. Due to pruning (data points or attributes), the computation reduced significantly.

7.3 Future Directions

The work presented in this thesis is to reduce the number of computations while detecting outliers from a given data set. To reduce the computations, we used cluster pruning and attributes pruning. For better performance this work can be extended in the following directions.

- In our work, we used K-means and leader selection clustering algorithms to divide the data set into clusters. It will be an useful study to identify the appropriate clustering algorithm for a given data set, because the properties of clusters depend on the clustering algorithm and the distribution of data points in the data set.
- In our proposed data pruning methods, we used $ldof$ value [21] to calculate the outlier score of a data point. Study may be carried out to check the effectiveness of other methods to calculate the outlier score of a data point.
- In our proposed attribute pruning method, we used Pearson Correlation Coefficients to identify the set of correlated attributes. Study may be performed by using other correlation coefficients and measure the performance of the method. There is also a possibility to apply the pruning methods (cluster pruning or data point pruning) after selecting the relevant attributes.
- There is a scope to apply this pruning technique to find the outliers in streaming data sets. Since in streaming data set all data points are not available at the same time, so the use of incremental clustering techniques may be explored for the clustering step of our outlier detection methods.
- Another important direction is to identify the proper method for a particular data set. Because the effectiveness of outlier detection methods depends on the behaviors of the data set; particularly distribution of data points, number of attributes, properties of attributes, etc.

References

[1] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, 2000.

[2] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining, southeast asia edition: Concepts and techniques*. Morgan kaufmann, 2006.

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[4] Markos Markou and Sameer Singh. Novelty detection: a review–part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[5] Christopher M Bishop. Novelty detection and neural network validation. In *IEE Proceedings of Vision, Image and Signal Processing,*, volume 141, pages 217–222. IET, 1994.

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

[7] Surendra P Verma and Lorena Díaz-González. Application of the discordant outlier detection and separation system in the geosciences. *International Geology Review*, 54(5):593–614, 2012.

[8] Edwin M Knorr and Raymond T Ng. A unified notion of outliers: Properties and computation. In *KDD*, pages 219–222, 1997.

[9] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proceedings of 19th International Conference on Data Engineering*, pages 315–326. IEEE, 2003.

[10] JF Gentleman and MB Wilk. Detecting outliers. II. Supplementing the direct analysis of residuals. *Biometrics*, pages 387–410, 1975.

[11] Douglas M Hawkins. *Identification of outliers*. Springer, 1980.

[12] Vic Barnett and Toby Lewis. *Outliers in statistical data*. Wiley New York, 1994.

[13] Richard Arnold Johnson, Dean W Wichern, et al. *Applied multivariate statistical analysis*. Prentice hall Englewood Cliffs, NJ, 1992.

[14] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.

[15] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):631–645, 2007.

[16] FY Edgeworth. XLI. On discordant observations. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 23(143):364–375, 1887.

[17] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John Wiley & Sons, 2005.

[18] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, volume 29, pages 427–438. ACM, 2000.

[19] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16(3):349–364, 2008.

[20] Yufei Tao, Xiaokui Xiao, and Shuigeng 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, KDD '06*, pages 394–403, New York, NY, USA, 2006. ACM.

[21] Ke Zhang, Marcus Hutter, and Huidong Jin. A new local distance-based outlier detection approach for scattered real-world data. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD '09*, pages 813–822, Berlin, Heidelberg, 2009. Springer-Verlag.

[22] Neminath Hubballi, Bidyut Kr Patra, and Sukumar Nandi. NDoT: nearest neighbor distance based outlier detection technique. In *Pattern Recognition and Machine Intelligence*, pages 36–42. Springer, 2011.

[23] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. *SIGMOD Rec.*, 29:93–104, 2000.

[24] Dongmei Ren, Baoying Wang, and William Perrizo. RDF: A density-based outlier detection method using vertical data representation. In *Fourth IEEE International Conference on Data Mining, (ICDM'04)*, pages 503–506. IEEE, 2004.

[25] Kathryn Chaloner and Rollin Brant. A bayesian approach to outlier detection and residual analysis. *Biometrika*, 75(4):651–659, 1988.

[26] Simon Byers and Adrian E Raftery. Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93(442):577–584, 1998.

[27] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection. In Daniel Barbara and Sushil Jajodia, editors, *Advances in Information Security*, chapter 4. Springer, 2002.

[28] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. *Proceedings of International Conference on Pattern Recognition*, 3:430–433, 2004.

[29] MP Veyssieres and Richard E Plant. Identification of vegetation state and transition domains in californias hardwood rangelands. *University of California*, 1998.

[30] P Arabie, LJ Hubert, and G De Soete. Complexity theory: An introduction for practitioners of classification. *Clustering and Classification: P. Arabie, LJ Hubert, G. De Soete*, page 199, 1996.

[31] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. ACM, 2001.

[32] Mon-Fong Jiang, Shian-Shyong Tseng, and Chih-Ming Su. Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, 22(6):691–700, 2001.

[33] J-B Hardouin and M Mesbah. Clustering binary variables in subscales using an extended rasch model and akaike information criterion. *Communications in Statistics-Theory and Methods*, 33(6):1277–1294, 2004.

[34] Boris Mirkin. *Mathematical classification and clustering: From how to what and why*. Springer, 1998.

[35] Zengyou He, Xiaofei Xu, and Shengchun Deng. Squeezer: an efficient algorithm for clustering categorical data. *Journal of Computer Science and Technology*, 17(5):611–624, 2002.

[36] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. *The VLDB Journal*, 8(3-4):222–236, 2000.

[37] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[38] Fuyuan Cao, Jiye Liang, Deyu Li, Liang Bai, and Chuangyin Dang. A dissimilarity measure for the k-modes clustering algorithm. *Knowledge-Based Systems*, 26:120–127, 2012.

[39] Bernd Fischer and Joachim M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):513–518, 2003.

[40] Alan H Cheetham and Joseph E Hazel. Binary (presence-absence) similarity coefficients. *Journal of Paleontology*, pages 1130–1136, 1969.

[41] Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. Springer, 2009.

[42] Michael R Anderberg. *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*. Academic press, 2014.

[43] Paul MB Vitányi. Information distance in multiples. *IEEE Transactions on Information Theory*, 57(4):2451–2456, 2011.

[44] P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55, 1936.

[45] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[46] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. Towards effective and efficient distributed clustering. In *Workshop on Clustering Large Data Sets (ICDM2003)*, 2003.

[47] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[48] Stuart Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[49] Geoffrey H Ball and David J Hall. Promenade-an on-line pattern recognition system. Technical report, DTIC Document, 1967.

[50] MM Astrahan. Speech analysis by clustering, or the hyperphoneme method. Technical report, DTIC Document, 1970.

[51] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[52] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[53] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.

[54] Sergios Theodoridis and Konstantinos Koutroumbas. Clustering: basic concepts. *Pattern Recognition*, pages 483–516, 2006.

[55] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition, 1975.

[56] H. Späth. *Cluster analysis algorithms for data reduction and classification of objects*. Computers and their applications. E. Horwood, 1980.

[57] Sushmita Mitra and Tinku Acharya. *Data mining: multimedia, soft computing, and bioinformatics*. John Wiley & Sons, 2005.

[58] S Asharaf and M Narasimha Murty. A rough fuzzy approach to web usage categorization. *Fuzzy sets and systems*, 148(1):119–129, 2004.

[59] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

[60] William DuMouchel, Chris Volinsky, Theodore Johnson, Corinna Cortes, and Daryl Pregibon. Squashing flat files flatter. In *Proceedings of the fifth ACM SIGKDD*

international conference on Knowledge discovery and data mining, pages 6–15. ACM, 1999.

[61] Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Fast hierarchical clustering based on compressed data and optics. In *Principles of Data Mining and Knowledge Discovery*, pages 232–242. Springer, 2000.

[62] Markus M Breunig, Hans-Peter Kriegel, Peer Kröger, and Jörg Sander. Data bubbles: Quality preserving performance boosting for hierarchical clustering. In *ACM SIGMOD Record*, volume 30, pages 79–90. ACM, 2001.

[63] Bidyut Kr Patra and Sukumar Nandi. Effective data summarization for hierarchical clustering in large datasets. *Knowledge and Information Systems*, pages 1–20, 2013.

[64] Jiguang Wang. *Pearson Correlation Coefficient*. Springer New York, 2013.

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

[66] Irad Ben-Gal. Outlier detection. In *Data Mining and Knowledge Discovery Handbook*, pages 131–146. Springer, 2005.

[67] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, 2007.

[68] Sigurour E Guttormsson, RJ Marks, MA El-Sharkawi, I Kerszenbaum, et al. Elliptical novelty grouping for on-line short-turn detection of excited running rotors. *IEEE Transactions on Energy Conversion,*, 14(1):16–22, 1999.

[69] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining Knowledge Discovery*, 12(2-3):203–228, 2006.

[70] Fabrizio Angiulli, Stefano Basta, and Clara Pizzuti. Distance-based detection and prediction of outliers. *IEEE Transactions on Knowledge and Data Engineering*, 18:145–160, 2006.

[71] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–26, London, UK, 2002. Springer-Verlag.

[72] Fabrizio Angiulli and Clara Pizzuti. Outlier mining in large high-dimensional data sets. *IEEE Transactions on Knowledge and Data Engineering*, 17:203–215, 2005.

[73] Mingxi Wu and Christopher Jermaine. Outlier detection by sampling with accuracy guarantees. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 767–772. ACM, 2006.

[74] Tang Jian, Chen Zhixiang, Fu Ada Wai Chee, and Cheung David Wai Lok. Enhancing effectiveness of outlier detections for low density patterns. In *PAKDD '02: Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 535–548, London, UK, 2002. Springer-Verlag.

[75] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, New York, NY, USA, 2004. ACM.

[76] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[77] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*, page 512, Washington, DC, USA, 1999. IEEE Computer Society.

[78] Levent Ertöz, Michael Steinbach, and Vipin Kumar. *Finding topics in collections of documents: A shared nearest neighbor approach.* Springer, 2004.

[79] Rasheda Smith, Alan Bivens, Mark Embrechts, Chandrika Palagiri, and Boleslaw Szymanski. Clustering approaches for anomaly-based intrusion detection. In *In Proceedings of the Intelligent Engineering Systems through Artificial Neural Networks.*, page 579584. ASME Press, 2002.

[80] Teuvo Kohonen, editor. *Self-organizing maps.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[81] Khaled Labib and Rao Vemuri. Nsom: A real-time network-based intrusion detection system using self-organizing maps. *Networks and Security*, pages 1–6, 2002.

[82] Manikantan Ramadas, Shawn Ostermann, and Brett Tjaden. Detecting anomalous network traffic with self-organizing maps. In *Recent Advances in Intrusion Detection*, LNCS, pages 36–54. Springer, 2003.

[83] Alexander Ypma and Robert PW Duin. Novelty detection using self-organizing maps. *Progress in connectionist-based information systems*, 2:1322–1325, 1997.

[84] V. Emamian, M. Kaveh, and A. H. Tewfik. Robust clustering of acoustic emission signals using the kohonen network. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000, ICASSP '00*, pages 3891–3894, Washington, DC, USA, 2000. IEEE Computer Society.

[85] Daniel Barbará, Yi Li, Julia Couto, Jia-Ling Lin, and Sushil Jajodia. Bootstrapping a data mining intrusion detection system. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 421–425, New York, NY, USA, 2003. ACM.

[86] Zengyou He, Shengchun Deng, and Xiaofei Xu. Outlier detection integrating semantic knowledge. In *WAIM '02: Proceedings of the Third International Conference on Advances in Web-Age Information Management*, pages 126–131, London, UK, 2002. Springer-Verlag.

[87] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[88] A Pires and Carla Santos-Pereira. Using clustering and robust estimators to detect outliers in multivariate data. In *Proceedings of the International Conference on Robust Statistics*, 2005.

[89] Matthew V. Mahoney and Philip K. Chan. Learning rules for anomaly detection of hostile network traffic. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 601, Washington, DC, USA, 2003. IEEE Computer Society.

[90] Huaming Huang, Kishan Mehrotra, and Chilukuri K Mohan. Rank-based outlier detection. *Journal of Statistical Computation and Simulation*, 83(3):518–531, 2013.

[91] Lian Duan, Lida Xu, Ying Liu, and Jun Lee. Cluster-based outlier detection. *Annals of Operations Research*, 168(1):151–168, 2009.

[92] Lian Duan, Lida Xu, Feng Guo, Jun Lee, and Baopin Yan. A local-density based spatial clustering algorithm with noise. *Information Systems*, 32(7):978–986, 2007.

[93] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*, pages 29–38, New York, NY, USA, 2003. ACM.

[94] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25:103–114, 1996.

[95] Hanan Samet. *The design and analysis of spatial data structures*. Addison-Wesley Reading, MA, 1990.

[96] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 157–166, New York, NY, USA, 2005. ACM.

[97] Murray H Protter and Charles Bradfield Morrey. *College calculus with analytic geometry*. Addison-Wesley, 1964.

[98] Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*. Pearson Education India, 1999.

[100] Hui Cao, Gangquan Si, Yanbin Zhang, and Lixin Jia. Enhancing effectiveness of density-based outlier mining scheme with density-similarity-neighbor-based outlier factor. *Expert Systems with Applications*, 37(12):8090–8101, 2010.

[101] Mark Goadrich, Louis Oliphant, and Jude Shavlik. Learning ensembles of first-order clauses for recall-precision curves: A case study in biomedical information extraction. In *Inductive logic programming*, pages 98–115. Springer, 2004.

[102] Mahito Sugiyama and Karsten Borgwardt. Rapid distance-based outlier detection via sampling. In *Advances in Neural Information Processing Systems*, pages 467–475, 2013.

[103] Charu C Aggarwal. *Outlier analysis*. Springer, 2013.

[104] M. Lichman. UCI machine learning repository, 2013.

[105] Gregory Piateski and William Frawley. *Knowledge discovery in databases*. MIT press, 1991.

[106] Ronald R Yager. A new approach to the summarization of data. *Information Sciences*, 28(1):69–86, 1982.

[107] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008 (CVPR 2008)*, pages 1–8. IEEE, 2008.

[108] Varun Chandola and Vipin Kumar. Summarization–compressing data into an informative representation. *Knowledge and Information Systems*, 12(3):355–378, 2007.

[109] Yan Hu, Dah-Ming Chiu, and John CS Lui. Entropy based adaptive flow aggregation. *Networking, IEEE/ACM Transactions on*, 17(3):698–711, 2009.

[110] Varun Chandola, Eric Eilertson, Levent Ertoz, Gyorgy Simon, and Vipin Kumar. *MINDS: Architecture & Design*. Springer, 2007.

[111] Leonid Portnoy, Eleazar Eskin, and Salvatore J. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, pages 5–8, November 2001.

[112] Rongbo Zhu. Intelligent rate control for supporting real-time traffic in wlan mesh networks. *Journal of Network and Computer Applications*, 34(5):1449–1458, 2011.

[113] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[114] Lei Yu and Fuji Ren. A study on cross-language text summarization using supervised methods. In *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering, 2009. (NLP-KE 2009)*, pages 1–7. IEEE, 2009.

[115] Sheldon Ross. *A First Course in Probability 8th Edition*. Pearson, 2009.

[116] Brett G Amidan, Thomas A Ferryman, and Scott K Cooley. Data outlier detection using the chebyshev theorem. In *Aerospace Conference*, pages 3814–3819. IEEE, 2005.

[117] Cui Zhu, Hiroyuki Kitagawa, and Christos Faloutsos. Example-based robust outlier detection in high dimensional datasets. In *Proceedings of 5th IEEE International Conference on Data Mining*, pages 829–832. IEEE, 2005.

[118] Zengyou He, Xiaofei Xu, Zhexue Joshua Huang, and Shengchun Deng. Fp-outlier: Frequent pattern based outlier detection. *Computer Science and Information Systems*, 2(1):103–118, 2005.

[119] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256, 1992.

[120] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*, volume 3, pages 856–863, 2003.

[121] Pabitra Mitra, CA Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[122] Subrata K Das. Feature selection with a linear dependence measure. *IEEE transactions on Computers*, 20(9):1106–1109, 1971.

Publications Related to Thesis

Published:

Journals

1. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. Centroid Based Cluster Pruning Outlier Detection Method. International Journal of Systems, Algorithms & Applications, Volume-3, Pages: 20-24, March 2013, ISSN Online: 2277-2677.

Conference Proceedings

1. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. Distance based fast outlier detection method. India Conference (INDICON), Pages: 1-4, IEEE 2010.

2. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. An outlier detection method based on clustering. Second International Conference on Emerging Applications of Information Technology (EAIT), Pages: 253-256, IEEE 2011.

3. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. Pruning based method for outlier detection. Third International Conference on Emerging Applications of Information Technology (EAIT), Pages: 210-213, IEEE 2012.

4. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. An Outlier Detection Method Based on Cluster Pruning. Second International Conference on Business & Information Management (ICBIM-2014), Pages:138-141, IEEE 2014.

Under Communication:

Journals

1. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. Cluster Based Pruning for Outlier Detection. International Journal of Data Mining and Knowledge Discovery, March 2015.

2. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. Effective Data Summarization Based Pruning for Outlier Detection. International Journal of Knowledge and Information Systems, March 2015.

3. **Rajendra Pamula**, Jatindra Kumar Deka, and Sukumar Nandi. Correlation Based Pruning for Outlier Detection. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, March 2015.

Brief Biography of the Author

Rajendra Pamula graduated Bachelor of Technology (B.Tech), Computer Science and Engineering from Rajiv Gandhi College of Engineering and Technology, Nandyla, India, affiliated to JNTU Hyderabad, India; completed Master of Technology (M.Tech), in Computer Science and Engineering from GITAM Univeristy (formely Gandhi Institute of Technology and Management), Visakhapatnam, India; affiliated to Andhra Univeristy, Visakhapatnam, India. He is currently working as Assistant Professor in the department of Computer Science and Engineering at Indian School of Mines, Dhanbad. His present research work is data pruning based outlier detection. His research interests are Data Mining and Outlier Detection.

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati 781039, India