

A Study on Distance-based Outlier
Detection on Uncertain Data

March 2014

Salman Ahmed SHAIKH

A Study on Distance-based Outlier
Detection on Uncertain Data

Graduate School of
Systems and Information Engineering

University of Tsukuba

March 2014

Salman Ahmed SHAIKH

Acknowledgements

All praises and thanks are due to Allah, our Lord and Creator, the Almighty, for endowing me with strength, patience and knowledge to complete this dissertation. I thank Allah, the Generous, for having finally made my dream come to reality.

This work would not have been completed without the help and support from many people and organizations. Here I would like to express my thankfulness to every one who helped me or participated by any mean to accomplish this work.

First of all, I express my immense gratitude and sincere appreciation to my academic advisor, Professor Hiroyuki Kitagawa, for his invaluable advices and continuous encouragement throughout the whole dissertation process. Without his keen supervision, scientific rigor, intellectual insights, and fruitful discussions, this work would not have been completed in the present form. His questions during our meetings and seminars helped me a lot to develop patience, intellect and meticulous attention to detail, and positive communication. I also really appreciate his wise understating of my Islamic culture which made my daily life much easier from the first days in his laboratory.

My gratitude goes also to Associate professor Toshiyuki Amagasa and Assistant professor Hideyuki Kawashima for their friendly discussions and hospitality. I am extremely grateful for their many good advices, kind help, and indispensable guidance on both academic and personal levels. I am also thankful to Assistant professors Yasuhiro Hayase and Chiemi Watanabe for their fruitful suggestions during the seminars to improve my work.

I would like to express my appreciation to the members of my dissertation committee, Professor Mikio Yamamoto and Kazuhiro Fukui and Associate Professors Jun Sakuma and Toshiyuki Amagasa for devoting time and effort to read and evaluate the manuscript, and for their helpful comments to improve it.

Many thanks to all my colleagues and friends at KDE Laboratory for making

it a delightful place to work. I really appreciate their kindness and cooperation. Special thanks to a colleague Takahiro Komamizu for his help and support on both academic and personal levels. I also want to thank Miss Yumiko Hisamatsu and Miss Hiroko Odagiri who have been kind enough to advise and help in their respective roles.

This thesis was made possible by a Japanese Government Scholarship. I would like to acknowledge the financial, technical and administrative support of the Ministry of Education, Culture, Sports, Science and Technology (MEXT), particularly in the award of Monbukagakushou scholarship which undoubtedly contributed to the success of my Ph.D. program.

I wish to express my sincere appreciation and thanks to my friends who shared with me the joys and the hardships of living in Japan. Without their unconditional support everything would be much harder.

A lot of thanks to my brothers, my sisters, and my family members who constantly supported and encouraged me at all times of this thesis. Special thanks goes to my wife for her unconditioned emotional and moral support and great patience all along the way.

Most of all, I would like to express my deepest and endless gratitude and thanks to my beloved parents. Despite the long distance, their prayers were always with me, which encouraged me a lot to complete this dissertation. Without their strong support and continuous encouragement, the completion of this work would be hardly possible. To them, I would like to dedicate this thesis.

Abstract

Recently, many new techniques for collecting data have resulted in an increase in the availability of uncertain data. For example, many new hardware technologies such as sensors generate data which is imprecise, many scientific measurement techniques are inherently imprecise, in many applications such as privacy-preserving data mining, the data is modified by adding perturbations to it, many mobile objects tracking applications generate imprecise data, etc. The uncertainty information in the data is useful and can be used to improve the quality of the underlying results. Therefore in this dissertation, we focus on one of the challenging research problem in data mining, outlier detection on uncertain data.

To address this problem, we focus on distance-based approach because the distance-based approach is the simplest and the most commonly used. It can be used as preprocessing before applying more sophisticated application dependent outlier detection techniques. Moreover, it coincides well with other data mining techniques i.e., k -nearest neighbors, clustering, etc. In this dissertation, the uncertainty of data is modeled by the Gaussian probability density function, because in statistics it is the most important and the most commonly used. Hence in this dissertation, the following problems are being solved related to outlier detection on uncertain data. 1) Distance-based outlier detection on uncertain data (UDB outlier detection), 2) Top- k outlier detection on uncertain data (k UDB outlier detection), 3) Continuous outlier detection on uncertain data streams (CUDB outlier detection).

1) UDB Outlier Detection: In this research, we give a novel definition of distance-based outliers on uncertain data. Since the distance probability computation is expensive, a cell-based approach is proposed to index the dataset objects and to speed up the outlier detection process. The proposed approach identifies and prunes the cells containing only inliers based on its bounds on out-

lier score ($\#D$ -neighbors). Similarly it can also detect the cells containing only outliers. Although the cell-based approach is very effective, yet it may leave some cells undecided, i.e., they are neither identified as inlier cells nor as outlier cells. For the uncertain data objects in such cells the Naive computation follows, which is the use of nested loop to compute un-pruned objects' $\#D$ -neighbors. The computation of $\#D$ -neighbors is an expensive process due to the expensive distance probability computation. Therefore, an approximate approach of outlier detection using the bounded Gaussian uncertainty is also proposed. The basic idea is that the bounded Gaussian distribution is a good approximation of the Gaussian distribution and can increase the outlier detection efficiency at a small loss of accuracy.

2) k UDB Outlier Detection: Existing approaches on outlier detection on uncertain data including our proposed approach, UDB outlier detection, can only perform binary classification, i.e., they can either classify an object as an inlier or an outlier. However, end users are usually interested in strong outliers and their ranking. Hence in this research, we present a top- k distance-based outlier detection approach. In order to detect top- k distance-based outliers from uncertain data efficiently, we propose a data structure, populated-cells list (PC-list). The PC-list is a sorted list of non-empty cells of a d -dimensional grid, where the grid is used to index dataset objects. Using the PC-list, the top- k outlier detection algorithm needs to consider only a fraction of the dataset objects and hence quickly identifies candidate objects for the top- k outliers. Finally, exact outlier score ($\#D$ -neighbors) is computed for each candidate object to find the top- k outliers and their ranking. Since the computation of exact outlier score is costly, two approximate top- k outlier detection approaches are also presented to reduce this cost. The first approximate approach, approximates only the candidate objects' $\#D$ -neighbors, while the second approximate approach makes use of the bounded Gaussian uncertainty to increase the efficiency of the top- k outlier detection.

3) CUDB Outlier Detection: Many of the automated data collection devices generate time series data streams (e.g., WSNs, monitoring cameras, etc.), which contain uncertainty. Outlier detection on uncertain static data is itself a challenging research problem in data mining. Moreover, the continuous and high speed arrival of data makes it more challenging. Hence in this part of the dissertation, we propose continuous outlier detection approach on uncer-

tain time series data streams. Namely, a distance-based approach is proposed to detect outliers continuously from a set of uncertain objects' states that are originated synchronously from a group of data sources (e.g., sensors in WSN). A set of objects' states at a timestamp is called a state set. Usually, the duration between two consecutive timestamps is very short and the state of all the objects may not change much in this duration. Therefore, to avoid the unnecessary computation at every timestamp, an incremental approach of outlier detection is proposed which makes use of the outlier detection results obtained from the previous timestamp to detect outliers in the current timestamp. Moreover, an approximate continuous outlier detection approach using the bounded Gaussian uncertainty is also proposed to further reduce the cost of the incremental outlier detection.

Finally, extensive experimental evaluations on real and synthetic datasets are presented for each of the proposed outlier detection approaches, to prove their accuracy, efficiency and scalability.

Keywords Distance-based Outlier Detection, Continuous Outlier Detection, Uncertain Data, Gaussian Uncertainty, Bounded Gaussian Uncertainty, Cell-based Approach, PC-list based Approach.

Contents

Acknowledgements	i
Abstract	iii
List of Figures	xi
List of Tables	xii
List of Algorithms	xv
1 Introduction	1
1.1 Background	1
1.2 Dissertation Motivation	3
1.3 Dissertation Contributions	4
1.3.1 Outlier Detection on Uncertain Data (UDB Outlier De- tection)	4
1.3.2 Top- k Outlier Detection on Uncertain Data (k UDB Out- lier Detection)	5
1.3.3 Continuous Outlier Detection on Uncertain Data (CUDB Outlier Detection)	6
1.4 Dissertation Organization	6
2 Preliminaries	9
2.1 Outlier Detection	9
2.1.1 What is an Outlier?	10
2.1.2 Outlier Detection Applications	11
2.1.3 Outlier Detection Approaches	14
2.2 Uncertain Data	19

2.2.1	Causes of Uncertainty	20
2.2.2	Types of Uncertainty	21
3	Related work	25
3.1	Outlier Detection on Deterministic Data	25
3.1.1	Distance-based Outlier Detection on Static Data	27
3.1.2	Distance-based Outlier Detection on Data Streams	28
3.2	Outlier Detection on Uncertain Data	28
3.2.1	Outlier Detection on Uncertain Static Data	29
3.2.2	Outlier Detection on Uncertain Data Streams	30
4	Outlier Detection on Uncertain Data	33
4.1	Overview	33
4.2	Problem Formulation	35
4.3	Cell-based Outlier Detection	38
4.3.1	Cell-based Pruning	39
4.3.2	Object-wise Bounds Pruning	42
4.3.3	Un-pruned Objects Processing and Grid File Index	44
4.3.4	Complexity Analysis	45
4.3.5	Discussion: Determination of Values for Parameters D , p and l	46
4.4	Cell-based Outlier Detection using the Bounded Gaussian Un- certainty	47
4.4.1	Cell-based Pruning for the Bounded Gaussian	48
4.4.2	Simple Object-wise Distance Pruning	52
4.4.3	Object-wise Bounds Pruning	52
4.4.4	Un-pruned Objects Processing for the Bounded Gaussian	53
4.4.5	Complexity Analysis	54
4.5	Discussion: Outlier Detection in High-dimensional Data and Sub-space Outlier Detection	54
4.6	Experiments	56
4.6.1	Datasets	56
4.6.2	Accuracy	59
4.6.3	Efficiency	62
4.7	Summary	69

5 Tok-k Outlier Detection on Uncertain Data	71
5.1 Overview	71
5.2 Problem Formulation	72
5.3 PC-list-based Outlier Detection	74
5.3.1 Grid (\mathcal{G}) Structure	75
5.3.2 PC-list Structure	76
5.3.3 Cell Bounds	77
5.3.4 Candidate Outlier Cells Detection	79
5.3.5 The k UDB(CG) and the k UDB(Approx) Algorithms . .	80
5.3.6 Complexity Analysis	82
5.4 PC-list-based Outlier Detection using the Bounded Gaussian Uncertainty	83
5.4.1 Grid (\mathcal{G}) and PC-list Structures for the Bounded Gaussian	84
5.4.2 Cell Bounds for the Bounded Gaussian	86
5.4.3 Candidate Outlier Cells Detection for the Bounded Gaus- sian	86
5.4.4 The k UDB(BG) Algorithm	87
5.4.5 Complexity Analysis	87
5.5 Experiments	88
5.5.1 Accuracy	89
5.5.2 Efficiency	93
5.6 Summary	97
6 Continuous Outlier Detection on Uncertain Data Streams	99
6.1 Overview	99
6.2 Problem Formulation	100
6.3 Cell-based Outlier Detection	102
6.3.1 Grid (\mathcal{G}) Structure	102
6.3.2 Cell Bounds	104
6.3.3 Cell Pruning	104
6.4 Incremental Outlier Detection	105
6.5 Incremental Outlier Detection using the Bounded Gaussian Uncertainty	109
6.5.1 Bounded Gaussian Cell-based Outlier Detection	110

6.5.2	Bounded Gaussian Incremental Outlier Detection	111
6.6	Complexity Analysis	111
6.7	Experiments	113
6.7.1	Datasets	114
6.7.2	Results	115
6.8	Summary	121
7	Conclusions and Future Works	123
7.1	Conclusions	123
7.1.1	UDB Outlier Detection	123
7.1.2	k UDB Outlier Detection	124
7.1.3	CUDB Outlier Detection	125
7.2	Future Works	125
7.2.1	Continuous Top- k Outlier Detection	126
7.2.2	UDB Outlier Detection for Very High Dimensional Data	126
7.2.3	UDB Outlier Detection for General Uncertainty Models	127
	Appendix	129
	A: A PDF for the Difference between Two Random Variables following d -dimensional Gaussian Distributions	129
	B: Transformation of a Correlated d -dimensional Gaussian Dis- tribution into an Uncorrelated Gaussian Distribution	130
	C: Approximate Values of the PDF for the Difference between Two Random Variables following the d -dimensional Gaussian PDF	132
	Bibliography	133
	List of Publications	147

List of Figures

1.1	Sensors arrangement in a research lab [4], for the monitoring of weather parameters	2
2.1	Tuple-level uncertainty in radar dataset [102]	22
2.2	Attribute-level uncertainty in weather sensor dataset	23
3.1	Distance-based outlier by Knorr et al. [65]	26
4.1	Cell layers	40
4.2	Cell and layers bounds	42
4.3	Bounded gaussian cell grid	50
4.4	Simple object-wise distance pruning	53
4.5	Datasets used in the experiments.	57
4.6	Precision-recall trade-off curves	58
4.7	Precision with increasing σ_p	59
4.8	Recall with increasing σ_p	60
4.9	Naive vs. proposed Approaches	61
4.10	Varying parameter l	63
4.11	Varying object's uncertainty σ	64
4.12	Varying parameter D	65
4.13	Varying parameter p	65
4.14	Cell-based pruning	66
4.15	Object-wise pruning	66
4.16	Un-pruned objects' processing	67
4.17	Vary dimensions d	68
4.18	Vary dimensions d and parameter D	68
4.19	Pre-computation time	69

5.1	Cell layers and bounds	75
5.2	PC-list building	77
5.3	Precision-recall trade-off curves	88
5.4	Precision with increasing σ_p	89
5.5	Recall with increasing σ_p	90
5.6	Effectiveness of the PC-list based approach	91
5.7	Varying parameter l	92
5.8	Varying object's uncertainty σ	92
5.9	Varying parameter D	93
5.10	Varying parameter k	94
5.11	Vary dimensions d	95
5.12	Vary dimensions d and parameter D	96
6.1	Cell Grid (\mathcal{G})	103
6.2	State sets	105
6.3	o_p moved among cells	106
6.4	o_p moved within cell	107
6.5	Met Office Weather (MOW) dataset [5]	114
6.6	Varying SC-object's percentage for UG dataset	115
6.7	Varying SC magnitude for UG dataset	116
6.8	SC-objects within and among cells	116
6.9	Varying parameter D for UG dataset	117
6.10	Varying parameter D for MOW dataset	117
6.11	Varying parameter p for UG dataset	118
6.12	Varying parameter p for MOW dataset	118
6.13	Varying object's uncertainty σ for UG dataset	119
6.14	Varying object's uncertainty σ for MOW dataset	119
6.15	Varying dimensions d for UG dataset	120
6.16	Varying dimensions d and parameter D for UG dataset	120

List of Tables

2.1	Uncertainty in commercial sensor measurements	20
3.1	Existing and proposed techniques of outlier detection on uncertain data	29
4.1	Pruning techniques for outlier detection on uncertain data	35
4.2	Standard error percentage in execution times	62

List of Algorithms

4.1	UDB Outlier Detection: Naive Approach	38
4.2	UDB Outlier Detection: Cell-based Approach	43
4.3	UDB Outlier Detection: ObjectWisePruning	44
4.4	UDB Outlier Detection: Cell-based Approach (Bounded Gaussian)	51
5.1	k UDB Outlier Detection: Naive Approach	74
5.2	k UDB Outlier Detection: PC-list Approach	80
6.1	CUDB Outlier Detection: Incremental Approach	108

Chapter 1

Introduction

In this chapter, the background, the motivation and the contributions of this dissertation are discussed. Moreover, the dissertation organization is given at the end of this chapter.

1.1 Background

In recent years, uncertain data has become ubiquitous because of new technologies for data collection data which can only measure and collect data in an imprecise way, e.g., wireless sensor networks, RFID, GPS, etc. Uncertainty in data is often caused by the limitations in underlying data collection equipments, inconsistent supply voltage and delay or loss of data in transfer [98]. Furthermore, many technologies such as privacy-preserving data mining create data which is inherently uncertain in nature. The uncertainty information in the data is useful which can be leveraged in order to improve the quality of underlying results. Therefore, there is a need for tools and techniques for mining and managing uncertain data. Hence in this dissertation, the problem of outlier detection from uncertain data is addressed.

Outlier detection is a fundamental problem in data mining. It has applications in many domains including credit card fraud detection [41], network intrusion detection [71], industrial damage detection [74], environment monitoring [49], medical sciences [14] etc. Several definitions of outlier have been given in past, but there exists no universally agreed definition. Hawkins [52] defined an outlier as an observation that deviates so much from other obser-

variations as to arouse suspicion that it was generated by a different mechanism. Barnett and Lewis [20] mentioned that an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.

In statistics, one can find over 100 outlier detection techniques. These have been developed for different data distributions, parameters, desired number of outliers and type of expected outliers [20, 73]. However, most statistical techniques are not useful in computer science due to several reasons. For example, most statistical techniques are univariate, in some techniques parameters are difficult to determine, and in other techniques outliers cannot be obtained until the underlying data distribution is known. In order to overcome these problems, several outlier detection techniques have been proposed in data mining [11, 63, 65, 68, 82, 86, 112, 120].

Most of the outlier detection techniques proposed in data mining are suitable only for deterministic data. However, due to the increasing usage of automated data collection technologies, data contains certain degree of inherent uncertainty [39, 54, 81, 98]. In order to get reliable results from such data, uncertainty needs to be considered in calculation. Hence this dissertation focuses on outlier detection from uncertain data, where the uncertainty of data is modelled by the most commonly used probability density function, i.e., the Gaussian distribution.

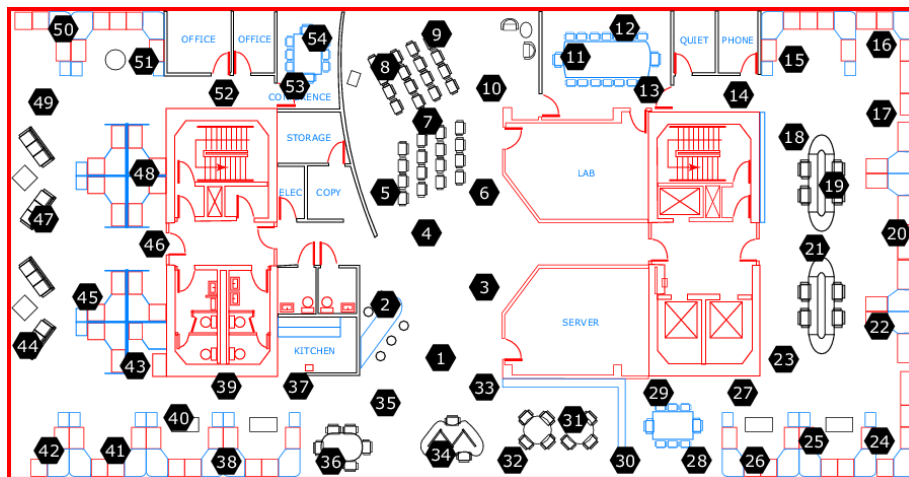


Figure 1.1: Sensors arrangement in a research lab [4], for the monitoring of weather parameters

1.2 Dissertation Motivation

Outlier detection is a fundamental problem in data mining with numerous applications in variety of domains. It is a well-studied problem both in statistics and data mining on the deterministic data. However, due to the ubiquitous presence of uncertain data, there is a need for outlier detection techniques on uncertain data. Outlier detection on uncertain data is in its infancy and quite a few researchers have explored this area. Table 3.1 lists some of the works proposed on outlier detection from uncertain data. It is very clear from the table that there are very few existing works on outlier detection from uncertain data. Specially there is no work on outlier detection from uncertain data streams considering the attribute-level uncertainty. Hence in this dissertation, our focus is outlier detection from attribute-level uncertain static data and data streams.

Outlier detection on uncertain data has applications in many domains including medical sciences where the accuracy has prime importance. At some places speed is more important e.g., nuclear reactors. Therefore in this dissertation, we present outlier-detection approaches on uncertain data. Namely, we present distance-based outlier detection approaches on uncertain data of the Gaussian distribution. The main objectives of this research are as follows.

1. To improve the accuracy of outlier detection in the presence of uncertainty.
2. To speed up the outlier detection process from uncertain data.

As stated earlier, outlier detection on uncertain data has applications in many domains, here we give two real world applications of outlier detection from uncertain data.

Motivating example 1 (Endangered species monitoring): Several wildlife conservation organizations monitor endangered species to prevent their loss. This is normally achieved by attaching sensors directly on the endangered species to monitor their activities and/or movement. Data obtained from these sensors are time-series data streams and contain uncertainty. Outlier detection on such data helps identify abnormal activities and/or movement of such species and let their caretakers take appropriate actions in case of danger.■

Motivating example 2 (Identification of malfunctioning sensors): In WSNs (e.g., WSN for the monitoring of weather parameters as shown in Fig. 1.1), observations are obtained continuously and contain certain degree of inherent uncertainty. Assuming that the observations are being generated synchronously at a timestamp by all the sensors in the WSN, a set of observations is obtained at every timestamp. An observation is outlier if it deviates markedly from other observations in the set and the sensors generating outlying observations in the majority of timestamps (say $> 50\%$) are identified as malfunctioning. ■

1.3 Dissertation Contributions

The ultimate goal of our research is to present accurate, scalable and efficient outlier detection algorithms for uncertain data. This dissertation compiles a study on distance-based outlier detection on uncertain data, where the uncertainty of a data object is given by the well-known Gaussian distribution. The Gaussian distribution is chosen to model an object's uncertainty because in statistics, the Gaussian distribution is the most famous and the most commonly used. As discussed in chapter 3 and summarized in table 3.1, there are very few existing works on outlier detection on uncertain data. Specially there is no work on outlier detection from uncertain data streams considering the attribute-level uncertainty. Hence, in this dissertation we focus on the attribute-level uncertainty of data. Moreover in this dissertation, our focus is low-dimensional data and we have used cell-based approach [65] to index the dataset objects and as a main pruning technique. This dissertation consists of three main contributions. Two of the contributions deal with static data, while one focuses on data streams. The main contributions are briefly described as follows.

1.3.1 Outlier Detection on Uncertain Data (UDB Outlier Detection)

In this research, we give a definition of distance-based outliers on uncertain data. In order to compute distance-based outliers in uncertain data, their outlier score ($\#D$ -neighbors) needs to be computed which is computationally very expensive. To efficiently obtain outliers from uncertain datasets, strong pruning techniques are required. Hence a cell-based approach is proposed to index the

dataset objects and to speed up the outlier detection process. The proposed approach identifies and prunes the cells containing only inliers based on its bounds on outlier score ($\#D$ -neighbors). Similarly it can also detect the cells containing only outliers. Although the cell-based technique is very effective, yet it may leave some cells undecided, i.e., they are neither identified as inlier cells nor as outlier cells. For the uncertain data objects in such cells the Naive computation follows, which is the use of nested loop to compute un-pruned objects' $\#D$ -neighbors.

To further reduce the computation cost of outlier detection, an approximate approach using the bounded Gaussian uncertainty is also proposed. The basic idea is that the bounded Gaussian distribution is a good approximation of the Gaussian distribution and can increase the outlier detection efficiency at a small loss of accuracy. Finally, detailed experiments on real and synthetic datasets are presented to prove the accuracy, efficiency and scalability of the proposed approaches.

1.3.2 Top- k Outlier Detection on Uncertain Data (k UDB Outlier Detection)

In most of the existing outlier detection approaches on uncertain data including our work *UDB-outlier detection* (Sec. 1.3.1), an object can be either classified as outlier or inlier. Since there is no universally agreed definition of outliers, different algorithms return different outliers depending upon the combination of parameter values. Some combinations return a very few while others return a lot of outliers. Moreover, no outlier ranking is available and users are unable to differentiate between strong and weak outliers. Therefore in this research we present a top- k approach of distance-based outlier detection, which returns k objects with lowest outlier scores ($\#D$ -neighbors) or in other words, k strongest outliers along with their ranking.

In this research, a populated-cells list (PC-list) is used to find the top- k outliers from uncertain datasets. The PC-list is a sorted list of non-empty cells of a d -dimensional grid, where the grid is used to index data objects. Using PC-list, the top- k outlier detection algorithm needs to consider only a fraction of the dataset objects and hence quickly identifies candidate objects for the top- k outliers. Finally, an exact outlier score ($\#D$ -neighbors) is computed for each

candidate object to find the top- k outliers and their ranking. Furthermore, two approximate top- k outlier detection approaches are also presented in this work to increase the efficiency of outlier detection. The first approximate approach, approximates only the candidate objects' $\#D$ -neighbors, while the second approximate approach makes use of the bounded Gaussian uncertainty to increase the efficiency of the top- k outlier detection algorithm. Lastly, we present detailed experiments on real and synthetic datasets to prove the accuracy, efficiency and scalability of the proposed approaches.

1.3.3 Continuous Outlier Detection on Uncertain Data (CUDB Outlier Detection)

In WSNs (e.g., moving objects monitoring, weather monitoring system, etc.), uncertain data arrive continuously and at high speed. Detection of outliers from uncertain static data is a challenging research problem in data mining and on top of that, the continuous arrival of data makes it more challenging. Hence, in this research, we present a continuous outlier detection approach on uncertain time series data streams.

In particular, we propose a continuous distance-based outlier detection approach on a set of uncertain objects' states that are originated synchronously from a group of data sources (e.g., sensors in WSN) at every timestamp. A set of objects' states at a timestamp is called a state set. Generally, the duration between two consecutive timestamps is very short and the state of all the objects may not change much in this duration. Therefore, we propose an incremental approach of outlier detection, which makes use of the results obtained from the previous state set to efficiently detect outliers in the current state set. In addition, an approximate incremental outlier detection approach using the bounded Gaussian uncertainty is proposed to further reduce the cost of the incremental outlier detection. Finally, an extensive empirical study on synthetic and real datasets is presented, which shows the effectiveness of the proposed approaches.

1.4 Dissertation Organization

In the above sections we have presented the background and motivations of this work, and briefly presented the main contributions of this dissertation. In this

section we detail the structure of this dissertation.

Chapter 2: In this chapter, we outline the basics of outlier detection, its applications and approaches. We also describe the uncertain data and the causes and the types of data uncertainty.

Chapter 3: In this chapter, we review state-of-the-art research works in the field related to our work. We first discuss some outlier detection works on deterministic data and then on uncertain data.

Chapter 4: This chapter introduces the uncertain distance-based outlier detection on uncertain data. We first define the distance-based outliers on uncertain data and then present a cell-based pruning technique to speed-up the outlier detection process. Moreover, an approximate approach of outlier detection is also presented to further increase the outlier detection efficiency and scalability. Accuracy, efficiency and scalability of the proposed approaches are proved with the help of experiments.

Chapter 5: In this chapter, a top- k distance-based outlier detection approach on uncertain data is presented. We present a populated-cells list approach to quickly identify the top- k candidate outlier objects. In addition, to further increase the efficiency of outlier detection, an approximate approach of top- k outlier detection is also presented. We also present detailed experiments to prove the accuracy, efficiency and scalability of the proposed approaches.

Chapter 6: This chapter deals with continuous outlier detection approach on uncertain data streams. We propose an incremental approach of outlier detection, which makes use of the results obtained from the previous timestamp to efficiently detect outliers in the current timestamp. An approximate approach is also presented to increase the efficiency of the incremental outlier detection. Efficiency and scalability of the proposed approaches are demonstrated with the help of experiments.

Chapter 7: In the final chapter, we give out the main conclusion on the work of this dissertation and outline some directions for future work.

Chapter 2

Preliminaries

In this chapter, we outline the basics of outlier detection, its applications and approaches. We also describe the uncertain data and the causes and the types of data uncertainty.

2.1 Outlier Detection

Outlier detection refers to the problem of finding patterns in data that do not conform to expected behavior. These nonconforming patterns are often referred as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different application domains [32]. Of these, outliers and anomalies are two terms used most commonly in the context of anomaly detection.

Outlier detection has several applications in variety of domains such as fraud detection for credit cards, insurance, or health care, intrusion detection for cyber-security, fault detection in safety critical systems, anomaly detection in text data, fault detection in web applications, identification of malfunctioning sensors in sensor networks, novelty detection in robot behavior and military surveillance for enemy activities.

The importance of outlier detection is due to the fact that outliers sometimes contain more important and often critical, actionable information than the normal data in a wide variety of application domains [32]. For example, an abnormal data traffic pattern in a computer network could mean that some hacker has gained access of a machine in a network and is accessing data from

it [68]. Drastic variation in the credit card usage pattern may suggest credit card theft [15], abnormal sensor readings from one or two sensors in a sensor network indicates the presence of malfunctioning sensors in the sensor network or an abnormal MRI data may suggest the presence of some disease [103].

Outliers or anomalies detection in data is quite an old field in the statistics community. One can find outlier definitions and techniques since late 18th century [48]. During this time, several outlier detection techniques have been proposed by several researchers. However, majority of the statistical techniques are univariate, many of them are distribution dependent, some of them are developed particularly for specific applications and only few of them are more generic. Recently, this field has been explored by a lot of researches from data mining community and one can find a lot of definitions and approaches of outlier detection for different types of data, applications of data or dimensions of data. In the following section, we will first discuss about some of the famous definitions of outlier and then will explore some of the well-known outlier detection applications.

2.1.1 What is an Outlier?

Outliers are patterns in data that do not conform to a well defined notion of normal behavior. Outliers might be induced in the data for a variety of reasons, such as malicious activity, for example, credit card fraud, cyber-intrusion, terrorist activity or breakdown of a system, but all of the reasons have the common characteristic that they are interesting to the analyst. The interestingness or real life relevance of anomalies is a key feature of anomaly detection [32].

Outlier detection is closely related to, but different from noise removal [56, 92, 109]. Noise removal deals with identifying and discarding of unwanted noise from data, with the aim of cleaning it. Noise is unwanted data, which is of no interest to the data miners and if not removed affects the results of analysis. Noise removal is therefore important and is needed for data cleaning before the data analysis.

Novelty detection is another topic related to outlier detection and is studied by wide variety of researchers [75, 76, 94]. It aims at detecting previously unobserved patterns in the data, for example, a new topic of discussion in a news group. The distinction between novel patterns and outliers is that the novel pat-

terns are typically incorporated into the normal model after being detected [32].

In contrast to noise and novelty in data, outlier is significant and useful information in the data, which is required by data miners to find the interesting patterns in data. It is therefore, outliers are sometimes more important than the normal data. In the following subsection, we will discuss some of the well-known applications of outlier detection.

2.1.2 Outlier Detection Applications

Outlier detection is a fundamental problem in data mining and has several applications in variety of domains. In the following, we discuss some of the well-known outlier detection applications.

Intrusion Detection

According to the Internet Security Dictionary [111], intrusion detection refers to the detection of malicious activity (break-ins, penetrations, and other forms of computer abuse) in a computer related system. These intrusions or attacks or malicious activities are very important and useful from a computer security perspective.

Intrusion detection is a security system for computers and networks. It monitors various areas of computers and networks (ports, data sent, data received) to prevent the computers and/or networks from possible internal or external attacks [9]. An intrusion is different from the normal behavior of the system, and therefore several outlier detection techniques have been proposed for the intrusion detection. Intrusion detection is quite a challenging area of data mining, because it is sometimes difficult to differentiate between the normal and abnormal computer/network usage pattern. Moreover, huge data volume and its continuous arrival require computationally efficient and online analysis [32].

Fraud Detection

Unauthorized usage of resources provided by commercial organizations such as credit card companies, banks, stock market, web shops and so on is referred to as fraud and detecting users involved in such activities is commonly referred as fraud detection. The hackers involved in fraud try to use up the organization's

resources in an unauthorized way by posing as real customers. Such frauds must be detected immediately to avoid economic losses.

In 1999, Fawcett and Provost [46] used a term activity monitoring for the identification of fraud and gave a framework to detect such activities. According to them the typical approach of outlier detection techniques in such domains is to maintain a usage profile for each customer and monitor the profiles to identify abnormalities.

Outlier Detection in Medical and Health Care

With the advancement in technology, medical facilities are improving and variety of machines and sensors are used to diagnose patients and to keep track of their health, respectively. Outlier detection in the medical and health care usually deals with patient data and can have outliers due to malfunctioning instrument, unstable patient condition, environmental conditions, etc. In [116], authors focused on detecting disease outbreaks in a specific area.

Outlier detection in this domain is critical and requires a very high degree of accuracy. In this domain, we usually deal with patient records which consists of several attributes including patient age, sex, heart beat, blood pressure, temperature, etc. The records may also contain temporal aspect, e.g., blood pressure values with respect to time and spatial aspect, e.g., heart beat at home versus at gymnasium. The main objective of the existing outlier detection approaches in this area is the identification of abnormal records.

Some of the medical devices produce time-series data, such as Electroencephalograms (EEG) and Electrocardiograms (ECG). Therefore outlier detection from time-series data is another challenging research problem in this domain. [70] used collective outlier detection techniques to detect outliers from such data.

Industrial Damage Detection

With the usage and passage of time, industrial units deteriorate. Early detection of this deterioration is essential to avoid drastic damages and human loss. Sensors are usually attached to these industrial units to early detect such damage. Several outlier detection techniques have been applied to detect such damage in this domain [74].

According to Varun et al. [32], industrial damage detection can be further classified into two domains, one that deals with defects in mechanical components such as motors, engines, and so on, and the other that deals with defects in physical structures

Outlier Detection in Text Data

Outlier detection from text data is a challenging research problem in data mining, due to very high dimensions and sparsity of data. Outlier detection approaches in this area primarily identify novel topics or events or news stories from a collection of documents or news articles [104]. This domain is also used by social networks, such as twitter and facebook for the detection of outlier and/or novel issues, news and topics. The outliers are usually caused due to a hot interesting event or an anomalous topic. Since the documents are collected over time, the data also has a temporal aspect. According to Varun et al. [32], a challenge for outlier detection approaches in this domain is the handling of the large variations in documents belonging to one category or topic .

Sensor Networks Anomaly Detection

With the availability of low cost and small size sensors, they are being used in several applications. Such as, sensor networks are popular for weather forecasting, endangered species monitoring, industrial damage detection, etc. Due to this reason, sensor networks have become a significant area of research. Usually the data obtained from sensors contain certain degree of inherent uncertainty, due to the limitations of equipment, inconsistent supply voltage, delay or loss of data in transfer, etc. Therefore, the sensor networks data is being studied under the head of deterministic as well as the uncertain data management and mining.

In the wireless sensor networks, the data is collected from several sensors. One of the interesting outlier detection application from such data is the identification of malfunctioning sensors [97]. Since the sensor networks are also used for the monitoring of endangered species, outlier detection from such data are the abnormal activities and/or movements of endangered species.

The data generated by a sensor or sensor networks is generally in the form of continuous streams. Therefore another challenging issue in this domain is

the continuous and online outlier detection from such data. Moreover, due to resource constraints, the algorithms to detect outliers from such data must be efficient. Some authors have given distributed outlier detection approaches to detect outliers at the sensor node where it is being generated [34, 84], while others prefer to process sensor data at a central location [25]. In addition, the presence of noise in the sensor data pose another challenge in the efficient and accurate processing of such data.

2.1.3 Outlier Detection Approaches

Due to the importance of outlier detection in data mining, several outlier detection approaches have been proposed. Some are proposed for specific applications while others are more generic. In the following subsections, some of the well-known approaches are discussed.

Nearest-Neighbor or Distance-based Outlier Detection

The concept of nearest neighbor analysis has been used in several outlier detection techniques. Nearest neighbor techniques assume that normal data have dense neighborhoods, while outliers exist far from their closest neighbors.

Nearest neighbor-based outlier detection techniques require a distance or similarity measure defined between two data instances and that is why nearest-neighbor outlier detection techniques are sometimes referred as distance-based outlier detection techniques. Distance (or similarity) between two data instances can be computed using Euclidean distance for continuous attributes, however other measures can also be used [106]. A matching coefficient method is usually used for the categorical attributes, however several works including [24,33] used more complex distance measures.

For multidimensional data objects, similarity or distance is normally evaluated for each attribute and then combined. Majority of the nearest neighbor approaches do not expect that the distance measure to be metric. Nearest neighbor-based outlier detection methods can be generally classified into two categories: 1) methods that use the distance between a data instance and its k^{th} nearest neighbor as the outlier score; 2) methods that make use of the relative density of each data object to compute its outlier score.

A basic nearest neighbor outlier detection technique is based on the following definition: The outlier score of a data object in a given dataset can be defined as its distance to its k^{th} nearest neighbor. Guttormsson et al. [95] used the basic technique to detect shorted turns (outliers) in the DC field windings of large synchronous turbine-generators. Similarly, Bayers et al. [30] applied this technique to detect land mines from satellite ground images. Normally, a threshold is used to distinguish between an outlier or inlier object when using the basic technique. On the other hand, Ramaswamy et al. [91] gave a slightly different definition by selecting n instances with the largest outlier scores as the outliers.

Eskin et al. [44], Angiulli et al. [17] and Zhang et al. [119] computed the outlier score of a data object as the sum of its distances from its k nearest neighbors. Bolton and Hand [23] used a similar technique called Peer Group Analysis to detect credit card frauds. Knorr et al. [63, 65] used the count of the number of nearest neighbors that are not more than D distance apart from the given data object to find if it is an outlier or not.

Wei et al. [69] in 2003 proposed a hypergraph-based technique, called HOT. In that technique, authors modelled the categorical values using a hypergraph, and measured distance between two data instances by analyzing the connectivity of the graph. Otey et al. [83] used a distance measure for data containing a mix of categorical and continuous attributes for outlier detection. They defined links between two instances by adding distance for categorical and continuous attributes separately. For categorical attributes, the number of attributes for which the two instances have the same values defines the distance between them. For continuous attributes, a covariance matrix is maintained to capture the dependencies between the continuous values. Palshikar [85] adapted the technique proposed in Knorr et al. [63] to continuous sequences. Kou et al. [67] extended the technique proposed in Ramaswamy et al. [91] to spatial data.

Density-based Outlier Detection

In this approach of outlier detection, the density of the neighborhood of each data object is estimated. An object is an outlier if it lies in a neighborhood with low density. On the other hand a data object that lies in a dense neighborhood is identified as normal.

If the dataset has varying densities, density-based techniques perform poorly.

In order to handle the problem of varying dataset densities, several techniques have been proposed to compute the relative neighborhood densities of the dataset objects.

Breunig et al. [26, 27] proposed density-based outlier detection techniques in which an outlier score is assigned to a data object, known as Local Outlier Factor (LOF). The LOF score of an object is given by the ratio of average local density of the k nearest neighbors of the data object and the local density of the data object itself. In order to compute the local density of a data object, the radius of the smallest hyper-sphere centered at the data object is found, that contains its k nearest neighbors. The authors then compute the local density by dividing k with the volume of this hyper-sphere. A data object is normal if it lies in a dense region and its local density is similar to that of its neighbors. On the hand, a data object is outlier, if its local density is lower than that of its nearest neighbors. Hence the outlier instance gets a higher LOF score.

A variation of the LOF was discussed by Tang et al. [107] in 2002. They called their approach Connectivity-based Outlier Factor (COF). The main difference between the COF and LOF lies in the way in which an object's k neighborhood is computed. Firstly, the nearest data object to the given data object is added to the neighborhood set. The next data object added to the neighborhood set is one whose distance to the existing neighborhood set is minimum among the remaining data objects. The distance between an object and a set of objects is obtained by computing the minimum distance between the given data object and any object belonging to the given set. Objects are added in this manner until the neighborhood size reaches k . After the neighborhood computation, the outlier score (COF) is obtained in the similar manner as in the case of LOF.

Hautamaki et al. [51] in 2004 proposed a simpler version of LOF, which computes Outlier Detection using In-degree Number (ODIN) for each data object. For a target data object, ODIN is defined as the number of k nearest neighbors of the data object which have the target data object in their k nearest neighbor list. The outlier score of a data object is obtained by taking the inverse of ODIN. Brito et al. [28] proposed a similar technique. In 2003, a measure called Multi-Granularity Deviation Factor (MDEF), which is a variation of LOF, was proposed by Papadimitriou et al. [86]. According to the authors, MDEF for a target data object is equal to the standard deviation of the local densities of the nearest neighbors (including the data object itself). The outlier score of the tar-

get data object is obtained by taking the inverse of the standard deviation. The authors named their outlier detection technique as LOCI. LOCI finds outlier micro-clusters in addition to the outlier objects.

Clustering-based Outlier Detection

Clustering [59, 106] is mainly designed to group similar data objects into clusters. Primarily it is an unsupervised technique, however semisupervised clustering techniques have also been explored recently by few authors including [21]. Although the main purpose of clustering is quite different from outlier detection, several works have been proposed for clustering-based outlier detection.

Clustering-based techniques are based on one of the following assumptions: 1) Normal data objects belong to a cluster in the data, while anomalies do not belong to any cluster. 2) Normal data objects lie close to their closest cluster centroid, while anomalies are far away from their closest cluster centroid. 3) Normal data objects belong to large and dense clusters, while anomalies either belong to small or sparse clusters.

Outlier detection techniques which follow the first assumption apply clustering-based algorithm to the data set. These techniques declare a data object an outlier if it does not belong to any cluster. Several existing algorithms such as DBSCAN [45], ROCK [50], and SNN clustering [42] do not force every data instance to belong to a cluster and can be used as outlier detection technique under the first assumption. In 2002, Yu et al. [117] proposed the FindOut algorithm which is an extension of the WaveCluster algorithm given by Sheikholeslami et al. [99]. In their algorithm, the detected clusters are removed from the data and the remaining data objects are identified as outliers. Such techniques are mainly designed for clustering and are not optimized to find outliers, which is one of the disadvantage of such techniques.

Outlier detection techniques which follow the second assumption consist of two steps. The data is clustered using a clustering algorithm in the first step. In the second step, for each data object, its distance to its nearest cluster centroid is calculated as its outlier score. Several outlier detection techniques proposed in data mining follow this two step approach using different clustering algorithms. In [101] authors proposed Self-Organizing Maps (SOM), K -means Clustering, and Expectation Maximization (EM) to cluster training data. These clusters are

then used to classify test data.

According to the third assumption, objects belonging to clusters whose size and/or density is below a threshold are identified as outliers. In this category of clustering-based outlier detection, several algorithms have been proposed [44, 53, 61, 72, 87]. He et al. [53] proposed a technique, called FindCBLOF. This technique assigns an outlier score CBLOF (Cluster-Based Local Outlier Factor) to each data object. The outlier score in this technique (CBLOF score) captures the size of the cluster to which the data objects belongs and the distance of the data object to its cluster centroid.

Classification-based Outlier Detection

In machine learning and data mining, classification [40, 106] is used to learn a model (classifier) from a set of labeled data instances (training) and then, classify a test instance into one of the classes using the learned model (testing). These techniques operate in two-phase fashion, like some of the clustering-based techniques. The first phase (also known as training phase) learns a classifier using the available training data (the training data is also called labelled data). The second phase (testing phase) classifies a test data object as normal or outlier, using the classifier [32].

Statistical Outlier Detection

Statistical outlier detection techniques are very old and the earliest of such techniques date back to early eighteenth century. The underlying principle of any statistical outlier detection technique is: An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed [18]. These techniques are based on the assumption that normal data objects lie in high probability regions of a stochastic model, while outliers occur in the low probability regions of the stochastic model.

Statistical techniques first identify the data distribution using the given data and then uses statistical inference test to find if a test data instance/object is consistent with this distribution or not. Instances which either do not belong or have a low probability of being generated from the learned distribution, based on the applied test statistic, are identified as outliers. Under the statistical outlier detection approaches, there exists parametric as well as nonparametric techniques.

The parametric techniques make use of the knowledge of the underlying data distribution and find out the parameters from the given data [43]. However, the nonparametric techniques do not consider knowledge of the underlying distribution [37].

2.2 Uncertain Data

In recent years many new techniques for collecting data, e.g., sensors, RFIDs, GPS have resulted in an increase in the availability of uncertain data [39, 54, 81, 98]. The causes of uncertainty may include but are not limited to limitation of equipments, absence of data, inconsistent supply voltage and delay or loss of data in transfer [98]. The uncertainty information in the data is useful information which can be leveraged in order to improve the quality of the underlying results. Some examples of applications which generate uncertain data are as follows [10]:

- A lot of scientific measurement techniques are inherently imprecise. In such cases, the level and type of uncertainty is derived from the errors in the instrumentation used for experiments.
- Many new hardware technologies such as sensors, GPS, RFIDs generate data which contains uncertainty. Table 2.1 lists the maximum measurement errors in some of the commercially available sensors. In such cases, the error in the sensor network readings can be modeled, and the resulting data can be modeled as imprecise or uncertain data.
- In some applications like the tracking of mobile objects, the future trajectory of the objects is modeled by forecasting techniques. Small errors in current readings can get amplified over the forecast into the distant future of the trajectory. This is usually encountered in cosmological applications when one models the probability of encounters with Near-Earth-Objects (NEOs). Errors in forecasting are also encountered in non-spatial applications such as electronic commerce.
- In several applications like privacy-preserving data mining, the data is modified by adding noise to it. In such cases, the format of the output is quite similar to that of uncertain data.

Table 2.1: Uncertainty in commercial sensor measurements

Company	Sensor Type	Model	Parameter	Error (%)*
Stevens [6]	Weather	WXT520	Air temperature	1
			Barometric pressure	0.2
			Relative humidity	5
			Wind speed	5
			Wind direction	3
	Pyranometer	LI200SA	Solar radiation	5
Vaisala [7]	Weather	HMP155	Air temperature	0.1
			Barometric pressure	0.05
		WMT700	Relative humidity	1.7
			Wind speed	2
		PTB110	Wind direction	0.55
	Pyranometer	CM6B	Solar radiation	2
Xylem [8]	Weather	WE100	Air temperature	1
		WE550	Barometric pressure	0.2
		WE570	Relative humidity	5
		WE600	Wind speed	5
		WE700	Wind direction	3
		Pyranometer	WE300	Solar radiation

*Maximum measurement error percentage : For some parameters, percentages are calculated from their respective maximum error values.

Uncertain data management and mining has gained a lot of interest among researchers in recent years because of a number of emerging fields which utilize this kind of data. For example, in fields such as privacy-preserving data mining, additional noise is added to data in order to hide the identity of the records. Often-times the data may be represented using statistical methods such as forecasting. In such scenarios, the data is uncertain in nature. Such data sets may often be probabilistic in nature. In other cases, databases may show existential uncertainty in which the presence or absence of tuples in database is uncertain. Such data sets lead to a number of unique challenges in managing and mining the underlying data.

2.2.1 Causes of Uncertainty

The causes of uncertainty in data obtained from devices like sensors, RFIDs, etc., may include but are not limited to limitation of equipments, inconsistent

supply voltage and delay or loss of data in transfer [98]. In this section, we focus on the uncertainty in data caused by the hardware and software faults in the data collection equipments.

The common hardware faults that have been observed to cause uncertainty in data include low battery, short-circuited connections, calibration errors and damaged sensors. Ramanathan et al. [89] and Szewczyk et al. [105] found that one of the main cause behind abnormally large or small sensor readings is short-circuit connections. They also identified that the low battery voltage results in a combination of constant faults and noise in temperature sensors. An example dataset available online (the Intel Lab, Berkeley deployment [4]), contains several tuples affected by variation in battery voltage.

A well-known root cause for sensor data uncertainty is calibration errors [79, 89, 90]. Calibration errors can corrupt the sensor measurements in different ways: 1) the parameters associated with a sensors original calibration formulas may change during a deployment (drift fault), 2) the measured value can differ from its true value by a constant amount (offset fault), and 3) the rate of the measured data can differ from the true/expected rate (gain fault). Calibration errors may affect all the samples collected during a deployment, and the faulty data may still exhibit normal patterns. For example, ambient temperature measurements affected by an offset fault will still exhibit a periodic pattern. Without the availability of ground-truth values or a model for expected sensor behavior, detecting data faults due to calibration errors remains an open problem [98].

In 2003, authors in [29] made use of spatial correlation across sensor nodes to develop methods for online sensor calibration that can be used to recover from calibration errors during a deployment, once such an error is detected. An example of software fault is given in Ni et al. [79] where the authors identify instances of short faults due to software errors during communication and data logging.

2.2.2 Types of Uncertainty

Uncertainty in datasets can be broadly divided into two types. 1) Tuple-level uncertainty, 2) Attribute-level uncertainty. Although converting a dataset with attribute-level uncertainty to a dataset with tuple-level uncertainty is a fairly simple operation, this transformation usually leads to a loss of shared correlation

structures (besides resulting in a database that requires storing a significantly larger number of tuples).

Tuple-level Uncertainty

In tuple-level uncertainty, uncertainty lies in the existence of a tuple in a relation or dataset. Tuple-level uncertainty is also known as existential uncertainty. According to [57], in the tuple-level uncertainty, tuples may belong to the database with less than absolute confidence. A commonly used model to work out this type of uncertainty is representing tuples as probabilistic events, and model the database as a joint distribution defined on these events. Fig. 2.1 shows a relation with tuple-level uncertainty. In this relation, each tuple is associated with the confidence of appearing in the relation. Most of the recent work in probabilistic databases has been concentrated on the development of tuple-level uncertainty [36, 47, 93].

Time	Radar Loc	Car Model	Plate No	Speed	Conf
11:45	L1	Honda	X-123	130	0.4
11:50	L2	Toyota	Y-245	120	0.7
11:35	L3	Toyota	Y-245	80	0.3
12:10	L4	Mazda	W-541	90	0.4
12:25	L5	Mazda	W-541	110	0.6
12:15	L6	Chevy	L-105	105	1.0

Figure 2.1: Tuple-level uncertainty in radar dataset [102]

Attribute-level Uncertainty

In attribute-level uncertainty (also known as value-level uncertainty), each uncertain attribute in a tuple is represented by its own independent probability density function. For example, if readings are taken of wind speed and temperature, each would be described by its own probability density function or other statistical parameters such as variance, as knowing the reading for one measurement would not provide any information about the other. Fig. 2.2 shows a relation with attribute-level uncertainty. In the relation, minimum and max-

imum temperature values are uncertain and are given by a probability density function.

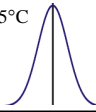
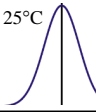
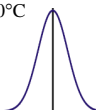
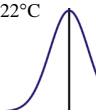
Date	City	Min. Temp	Max. Temp
11/04/2013	Tokyo	15°C 	25°C 
11/04/2013	Tsukuba	10°C 	22°C 

Figure 2.2: Attribute-level uncertainty in weather sensor dataset

Many applications produce data that is more naturally represented using attribute-level uncertainty as opposed to tuple-level uncertainty, e.g., mobile objects databases [35], and sensor network datasets [38], etc. The focus of this thesis is to detect outliers from uncertain data obtained from automated data collecting devices like sensors, RFIDs, etc., in which each tuple's attribute may have uncertain values. A commonly used model to capture this type of uncertainty is representing tuple attributes as probability density function. In this dissertation, our focus is attribute-level uncertainty and we have used Gaussian probability density function to represent the uncertainty of each tuple's attribute.

Chapter 3

Related work

In this chapter, we review state-of-the-art research works related to our work. There exists a lot of works on outlier detection from deterministic data, however very few authors have explored outlier detection on uncertain data. In the following, we will first discuss some of the outlier detection approaches/techniques given for deterministic data and then on uncertain data.

3.1 Outlier Detection on Deterministic Data

Outlier detection is one of the most important area of data mining research. Due to this reason outlier detection encompasses a broad spectrum of techniques. A lot of techniques given for outlier detection are basically identical but with different names given by the authors. Such as, authors use outlier detection, exception mining, novelty detection, noise detection, anomaly detection or deviation detection to describe their various approaches [55].

Several definitions of outlier have been given in literature, but there exists no universally agreed definition. Hawkins [52] in 1980 defined an outlier as an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism. Barnett and Lewis [20] in 1994 mentioned that an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.

According to John [62], an outlier may also be *surprising veridical data*, a point belonging to class A but actually situated inside class B so the true (veridical) classification of the point is surprising to the observer. Aggarwal and

Yu [12] defined outliers as noise points lying outside a set of defined clusters or alternatively outliers are the points that lie outside of the set of clusters but are also separated from the noise. These outliers behave differently from the norm.

A brief survey of different outlier detection approaches and applications is presented in chapter 2. In this dissertation our focus is distance-based outlier detection approach, because the distance-based approach is the simplest and the most commonly used. It can be used as preprocessing before applying more sophisticated application dependent outlier detection techniques. Moreover, it coincides well with other data mining techniques i.e., k -Nearest Neighbours, clustering, etc. Hence in this section, our focus is distance based approach of outlier detection. Although several definitions have been given for the distance-based outlier, there exists no universally agreed definition. Three main definitions of distance-based outlier on deterministic data to date are as follows.

- Outliers are the data points for which at least fraction p of the objects are outside the distance D . [63]
- Outliers are the data points whose distance to their k^{th} nearest neighbor is largest. [91]
- Outliers are the data points whose average distance to their k nearest neighbors is largest. [17]

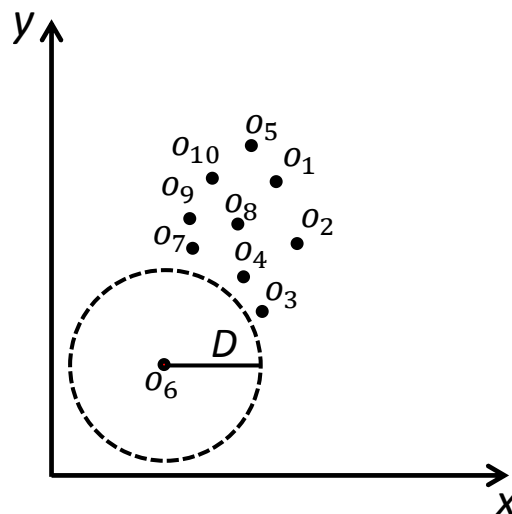


Figure 3.1: Distance-based outlier by Knorr et al. [65]

3.1.1 Distance-based Outlier Detection on Static Data

The very first definition of distance-based outlier was given by Knorr et al. in [65]. They defined a point p to be an outlier if at most M points are within D -distance of p . In Fig. 3.1, object o_6 is a distance-based outlier if the number of objects within its D distance is less than or equal to M . They also presented a cell-based approach to efficiently compute the distance-based outliers. They proved with experiments that their proposed cell-based algorithm outperforms the simple algorithms (the Naive algorithm using the nested-loop) for $k \leq 4$, where k denotes the number of dimensions. However for $k > 4$, the number of cells increase exponentially and the performance of cell-based algorithm decline significantly.

[91] formulated distance-based outliers as the data points whose distance to their k^{th} nearest neighbor is largest. They also ranked each point on the basis of its distance to its k^{th} nearest neighbor and declared the top n points in this ranking to be outliers. In addition to developing relatively straightforward solutions to finding such outliers based on the classical nested-loop join and index join algorithms, they developed an efficient partition-based algorithm for mining outliers. This algorithm first partitions the input data set into disjoint subsets, and then prunes entire partitions as soon as it is determined that they cannot contain outliers.

Angiulli et al. in [17] gave a slightly different definition of outliers than [91] by considering the average distance to their k nearest neighbours. They called the average distance, the weight. Outliers are the points with the largest values of weight. In order to compute these weights in an efficient way, authors linearized the search space through the Hilbert space filling curve. The algorithm consists of two phases, the first phase provides an approximated solution, within a small factor, after executing at most $d + 1$ scans of the data set with a low time complexity cost, where d is the number of dimensions of the data set. During each scan the number of points candidate to belong to the solution set is sensibly reduced. The second phase returns the exact solution by doing a single scan which examines further a little fraction of the data set.

3.1.2 Distance-based Outlier Detection on Data Streams

Beside the works discussed in Sec. 3.1.1 related to outlier detection from static data, there are some works on the detection of distance-based outliers over stream data including [16, 58, 66]. These works are based on the definition of distance-based outliers by Knorr et al. [65].

In [16], a method for detecting distance-based outliers in data streams is presented. They made use of the sliding window model, where outlier queries are performed in order to detect anomalies in the current window. In their work, two algorithms are presented. The first one exactly answers outlier queries, but has larger space requirements. The second algorithm is directly derived from the exact one, has limited memory requirements and returns an approximate answer. Later on [66] extended [16] work by adding the concepts of multi-query and micro-cluster based distance-based outlier detection.

In [58], authors gave a distance-based approach of outlier detection over data streams and made use of cell-based approach proposed in [65]. They presented an algorithm to detect outliers from time-series data streams, where streams in their work is a sequence of states generated synchronously by a group of objects. Their algorithm used a differential detection approach based on the change between consecutive state sets to reduce the computation cost of outlier detection in each state set.

All the works discussed in this section were given for deterministic data and cannot handle uncertain data. However, in this dissertation, the focus is outlier detection from uncertain data. Therefore in the next section, existing works related to outlier detection from uncertain data are presented. Since there are not many techniques proposed for outlier detection from uncertain data, we will explore all the major outlier detection techniques proposed so far, rather than sticking to distance-based approaches of outlier detection from uncertain data.

3.2 Outlier Detection on Uncertain Data

Recently a lot of research has focused on managing, querying and mining of uncertain datasets [13, 60, 112]. Table 3.1 summarizes the related work of outlier detection on uncertain data, both on static data and data streams. So far, very few works have been proposed to detect outliers from uncertain data. In the

Table 3.1: Existing and proposed techniques of outlier detection on uncertain data

Types of Uncertainty	Types of Data	
	Static Data	Data Streams
Attribute-level	Aggarwal et al.* [13] Jiang et al.* [60] Matsumoto et al.* [77] UDB outlier detection** k UDB outlier detection**	CUDB outlier detection**
Tuple-level	Wang et al.* [112]	Wang et al.* [113] Cao et al.* [31]

*Existing outlier detection techniques on uncertain data.

**Proposed outlier detection techniques on uncertain data.

following subsections, these works are discussed precisely.

3.2.1 Outlier Detection on Uncertain Static Data

The problem of outlier detection on uncertain datasets was first studied by Aggarwal et al. in [13]. According to them an uncertain object o is a density-based (δ, η) outlier, if the probability of existence of o in some subspace of a region with density at least η is less than δ . In order to compute (δ, η) outliers, firstly density of all subspaces needs to be computed and then the η -probability of each o in the dataset is computed to find if o is an outlier. Since this computation is very expensive, a sampling procedure is used to approximate the η -probability. In contrast to [13] work, this dissertation addresses the problem of distance-based outlier detection in full space, where the distance between two uncertain objects is computed by the Gaussian difference distribution [114]. Hence, our problem definition is quite different from [13]. Matsumoto et al. [77] extended the Aggarwal's work [13] by providing a parallel version of their algorithm using GPU (Graphics Processing Unit). In their work, parallelization is provided by the cross-platform OpenCL framework, which is used for programming GPU kernels.

Wang et al. in [112] also proposed outlier detection on uncertain data. Their work focuses on the uncertainty in the existence of a tuple, i.e., tuple-level uncertainty. Hence, each tuple in their work is associated with the confidence of

appearance in the dataset (An example of tuple-level uncertainty is shown in Table 2.1). In contrast, in this dissertation, attribute level uncertainty is considered, i.e., the uncertainty lies in the measurements obtained from sensors and this uncertainty is given by the Gaussian probability density distribution, with an assumption that sensor measurements may deviate from true values. In order to increase the efficiency of outlier detection, dynamic programming approach (DPA) and grid-based pruning approach (GPA) are used in their work.

In 2011, B. Jiang et al. [60] gave an outlier detection model considering both uncertain objects and their instances. According to their model, an uncertain object has some inherent attributes and consists of a set of instances which are modeled by a probability density distribution. Outliers are detected at both the instance level and the object level. To detect outlier instances, normal instances are first identified. By assuming that uncertain objects with similar properties tend to have similar instances, the normal instances for each uncertain object are learned using the instances of objects with similar properties. Consequently, outlier instances are detected by comparing against normal ones. Finally the objects, most of whose instances are outliers are identified as outlier. Technically, they used a Bayesian inference algorithm to solve the problem, and developed an approximation algorithm and a filtering algorithm to speed up the computation. However, in this dissertation, objects outlierness depends on its $\#D$ -neighbors (expected number of objects that lie within D distance of target object) rather than instances. Moreover, we have used the Gaussian difference distribution [114] to compute the probability that two uncertain objects lie within the D distance of each other.

3.2.2 Outlier Detection on Uncertain Data Streams

Since outlier detection from uncertain data streams is quite a new research field, only a couple of works are available related to it. Both the works focus on tuple-level uncertainty in contrast to the proposed work in this dissertation which focuses on attribute-level uncertainty.

In [113], Wang et al. proposed an outlier detection model for probabilistic data stream and presented a definition of distance-based outlier over sliding window. They showed the problem of detecting an outlier over a set of possible world instances is equivalent to the problem of finding the k^{th} element in its

neighborhood. Based on this observation, a dynamic programming algorithm (DPA) is proposed to reduce the outlier detection cost. Their work mainly focuses on tuple-level uncertainty. In contrast, in this dissertation, attribute level uncertainty is considered, i.e., the uncertainty lies in the measurements obtained from sensors and this uncertainty is given by the Gaussian probability density distribution, with an assumption that sensor measurements may deviate from true values. Moreover in their work, sliding window is used to detect continuous outliers from data streams. However, we have made use of an incremental approach of outlier detection, which makes use of results obtained from the previous state set to efficiently detect outliers in the current state set, where state set is a set of objects' states at a particular timestamp.

Recently Cao et al in [31] gave a solution of distance-based outlier detection from uncertain data streams [113]. In their work, tuple-level uncertainty is considered, where existential uncertainty lies in the tuple instances (attributes) rather than the complete tuple. Tuples' outlierness depends on its outlier instances (A tuple is an outlier if the sum of existential probability of its outlier instances is less than the user defined threshold). In contrast to their work, in this dissertation, attribute level uncertainty is considered and we have made use of an incremental approach of outlier detection, which makes use of the results obtained from the previous state set to efficiently detect outliers in the current state set rather than using the sliding window approach.

Chapter 4

Outlier Detection on Uncertain Data

The main goal of the research in this chapter is to identify distance-based outliers from uncertain data. Due to the surge in automated data collection technologies, data contain certain degree of inherent uncertainty. For example, data obtained from sensors, RFIDs, etc. may contain uncertainty due to the reasons discussed in Sec. 2.2. To obtain reliable results from such data, uncertainty must be considered in their processing. Since the main focus of this dissertation is outlier detection, in this chapter we will mainly focus on the outlier detection in the presence of uncertainty in data, where the uncertainty lies in the individual attributes of the dataset tuples and this uncertainty is modelled by the Gaussian distribution function.

4.1 Overview

Outlier detection is a fundamental problem in data mining. It has applications in many domains, credit card fraud detection [41], network intrusion detection [71], environment monitoring [49], medical sciences [14], etc. Several definitions of outlier have been given in past, but there exists no universally agreed definition. Hawkins [52] defined outlier as an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism. In statistics, one can find over 100 outlier detection approaches. These have been developed for different data distributions, parameters, desired

numbers of outliers and types of expected outliers [20, 73]. However, most statistical approaches are not useful due to several reasons. For example, most statistical approaches are univariate, in some approaches parameters are difficult to determine, and in other approaches outliers cannot be obtained until the underlying data distribution is known. In order to overcome these problems, several distance-based outlier detection approaches have been proposed in data mining [63, 65, 82, 112].

Most of the outlier detection approaches proposed in data mining are suitable only for deterministic data. However, due to the increasing usage of sensors, RFIDs and similar devices for data collection, data contains certain degree of inherent uncertainty [39, 54, 98]. The causes of uncertainty may include but are not limited to limitation of equipments, absence of data, inconsistent supply voltage and delay or loss of data in transfer [98]. In order to get reliable results from such data, uncertainty needs to be considered in calculation. Hence this chapter presents a distance-based outlier detection approach on uncertain data.

In order to obtain distance-based outliers from uncertain datasets, outlier score ($\#D$ -neighbors) computation is required for each dataset object. However, the $\#D$ -neighbors computation is very expensive. Therefore, a cell-based approach is proposed in this work. The use of cell-based approach is twofold in this dissertation, i.e., indexing and pruning. As a pruning approach, the proposed cell-based approach can identify and prune the cells containing only inliers using the cell bounds on $\#D$ -neighbors. Similarly it can also detect the cells containing outliers. Although the cell-based pruning is very effective, yet it may leave some cells undecided, i.e., they are neither identified as inlier cells nor as outlier cells. For the uncertain data objects in undecided cell, an object-wise bounds pruning approach is proposed. Finally nested-loop method is used for the un-pruned objects to compute their outlier scores ($\#D$ -neighbors). A property of the distance probability function used in this dissertation is that it produces higher values for objects located nearby and low values for two objects separated by a large distance. Hence, in the computation of $\#D$ -neighbors for the un-pruned objects, nearer objects are considered before the farther objects. In order to retrieve the nearer objects, cell-grid is used as an indexing approach.

In the cell-based approach, the unbounded nature of the Gaussian distribution prevents effective pruning. Moreover, $\#D$ -neighbors computation of un-pruned objects becomes expensive, since it needs to consider all the objects in

Table 4.1: Pruning techniques for outlier detection on uncertain data

Pruning Techniques	Outlier Detection Approaches	
	UDB(CG)	UDB(BG)
Cell-based pruning	✓	✓
Simple object-wise distance pruning	×	✓
Object-wise bounds pruning	✓	✓

the dataset. Therefore an approximate outlier detection approach is also proposed to reduce this cost. The basic idea is that the Gaussian distribution can be appropriately approximated by the bounded Gaussian distribution [88], and this bounded distribution allows to introduce strong pruning techniques. It saves a lot of computation cost at a small cost of accuracy.

Hence the work in this chapter presents two cell-based approaches for distance-based outlier detection on uncertain data. The exact approach using the conventional Gaussian uncertainty is denoted by $UDB(CG)$ and the approximate approach using the bounded Gaussian uncertainty is denoted by $UDB(BG)$ in the rest of the dissertation. Since each approach handles different nature of object's uncertainty (i.e., unbounded and bounded), different pruning techniques are proposed for both. Table 4.1 lists the pruning techniques proposed for both the approaches.

4.2 Problem Formulation

The distance-based outlier detection approach on deterministic data was introduced by Knorr et al. in [65]. They defined distance-based outliers as follows.

Definition 4.1 *An object o in a dataset DB is a distance-based outlier, if at least fraction p of the objects in DB lies greater than distance D from o .*

Def. 4.1 was given for deterministic datasets. However, the focus of this work is uncertain datasets whose attribute values are uncertain, where the uncertainty is given by the Gaussian distribution. The Gaussian distribution is chosen for representing uncertainty in this dissertation, because in statistics the Gaussian distribution (*or the normal distribution*) is the most important and the most commonly used.

Let o_i be d -dimensional uncertain objects, with attribute vector $\vec{\mathcal{A}}_i = (x_{i,1}, \dots, x_{i,d})^T$ following the Gaussian PDF with mean vector $\vec{\mu}_i = (\mu_{i,1}, \dots, \mu_{i,d})^T$ and co-variance matrix $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,d}^2)$, respectively. Namely, the vector $\vec{\mathcal{A}}_i$ is a random variable that follows the Gaussian distribution $\vec{\mathcal{A}}_i \sim \mathcal{N}(\vec{\mu}_i, \Sigma_i)$. Note that $\vec{\mu}_i$ denotes the observed coordinates (attribute values) of object o_i . The complete database consists of a set of such objects, $\mathcal{GDB} = \{o_1, \dots, o_N\}$, where $N = |\mathcal{GDB}|$ is the number of uncertain objects in \mathcal{GDB} . Hence Def. 4.1 can be extended naturally for uncertain datasets as follows.

Definition 4.2 *An uncertain object o in a database \mathcal{GDB} is a distance-based outlier, if the expected number of objects $o_i \in \mathcal{GDB}$ (including o itself) lying within D -distance of o is less than or equal to threshold $\theta = N(1 - p)$, where N is the number of uncertain objects in database \mathcal{GDB} , and p is the fraction of objects in \mathcal{GDB} that lies farther than D -distance of o .*

The objects that lie within the D -distance of o_i are called its D -neighbors, and the set of the D -neighbors of o_i and the number of D -neighbours are denoted by $DN(o_i)$ and $\#D\text{-neighbors}(o_i)$, respectively. In order to find the distance-based outliers in \mathcal{GDB} , the $\#D$ -neighbors of the un-pruned objects needs to be computed which requires the computation of distance probability. This distance probability is computed using the difference between two uncertain objects, which is given by another distribution known as the Gaussian difference distribution [114].

Let $\vec{\mathcal{A}}_i$ and $\vec{\mathcal{A}}_j$ be two independent d -dimensional normal random vectors with means $\vec{\mu}_i = (\mu_{i,1}, \dots, \mu_{i,d})^T$ and $\vec{\mu}_j = (\mu_{j,1}, \dots, \mu_{j,d})^T$ and diagonal covariance matrices $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,d}^2)$ and $\Sigma_j = \text{diag}(\sigma_{j,1}^2, \dots, \sigma_{j,d}^2)$, respectively. Then $|\vec{\mathcal{A}}_i - \vec{\mathcal{A}}_j| = \mathcal{N}(\vec{\mu}_i - \vec{\mu}_j, \Sigma_i + \Sigma_j)$ [114]. Let $Pr(o_i, o_j, D)$ denotes the probability that $o_j \in DN(o_i)$. Then,

$$Pr(o_i, o_j, D) = \int_R \mathcal{N}(\vec{\mu}_i - \vec{\mu}_j, \Sigma_i + \Sigma_j) d\vec{\mathcal{A}}, \quad (4.1)$$

where R is a sphere with centre $(\vec{\mu}_i - \vec{\mu}_j)$ and radius D . Lemma 4.1 gives the 2-dimensional expression for $Pr(o_i, o_j, D)$. However, $Pr(o_i, o_j, D)$ expressions for higher dimensions can be derived using Eq. 4.1.

Lemma 4.1 *Let o_i and o_j be two 2-dimensional uncertain objects with attributes $\vec{\mathcal{A}}_i \sim \mathcal{N}(\vec{\mu}_i, \Sigma_i)$ and $\vec{\mathcal{A}}_j \sim \mathcal{N}(\vec{\mu}_j, \Sigma_j)$, where $\vec{\mu}_i = (\mu_{i,1}, \mu_{i,2})^T$, $\vec{\mu}_j = (\mu_{j,1}, \mu_{j,2})^T$,*

$\Sigma_i = \text{diag}(\sigma_{i,1}^2, \sigma_{i,2}^2)$ and $\Sigma_j = \text{diag}(\sigma_{j,1}^2, \sigma_{j,2}^2)$. The $Pr(o_i, o_j, D)$ is given as follows.

$$Pr(o_i, o_j, D) = \frac{1}{2\pi \sqrt{(\sigma_{i,1}^2 + \sigma_{j,1}^2)(\sigma_{i,2}^2 + \sigma_{j,2}^2)}} \times \int_0^D \int_0^{2\pi} \exp \left\{ - \left(\frac{(r \cos \theta - \alpha_1)^2}{2(\sigma_{i,1}^2 + \sigma_{j,1}^2)} + \frac{(r \sin \theta - \alpha_2)^2}{2(\sigma_{i,2}^2 + \sigma_{j,2}^2)} \right) \right\} r \, d\theta \, dr, \quad (4.2)$$

where $\alpha_1 = \mu_{i,1} - \mu_{j,1}$ and $\alpha_2 = \mu_{i,2} - \mu_{j,2}$.

Proof. See Appendix A.

This work assumes that the attributes of uncertain objects are independent and the uncertainty of objects (standard deviation) is uniform in all dimensions, hence $\sigma_{i,1} = \sigma_{j,1} = \sigma_{i,2} = \sigma_{j,2} = \sigma$, and let $\alpha^2 = \alpha_1^2 + \alpha_2^2$. This results in a non-correlated diagonal covariance matrices, i.e., $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \sigma_{i,2}^2)$ and $\Sigma_j = \text{diag}(\sigma_{j,1}^2, \sigma_{j,2}^2)$ for the distance probability expression, $Pr(o_i, o_j, D)$. On the other hand, if the attributes of uncertain objects are dependent there exists a correlation between them and it results in a correlated covariance matrix. Appendix B shows that the series of transformations, Principal Component Analysis [100], is always possible to find alternative coordinates of an object, which transform a correlated Gaussian distribution into an uncorrelated one. Hence resulting in a diagonal, uncorrelated covariance matrix whose variance is uniform in all dimensions, which is consistent with the proposed probability function. Thus the proposed solution is equally applicable for the objects whose attributes are correlated. In the light of above assumptions, Eq. 4.2 is simplified as follows.

$$Pr(o_i, o_j, D) = \frac{1}{4\pi\sigma^2} \int_0^D \int_0^{2\pi} \exp \left\{ \frac{-1}{4\sigma^2} (r^2 - 2\alpha r \cos \theta + \alpha^2) \right\} r \, d\theta \, dr. \quad (4.3)$$

Note that $Pr(o_i, o_j, D)$ only depends on α^2 and not on the coordinates of o_i and o_j . Hence $Pr(o_i, o_j, D)$ is denoted by $Pr(\alpha, D)$ when there is no confusion, where α denotes the ordinary euclidean distance between the means of

uncertain objects. Computing this probability is usually very costly, and needs to be avoided as much as possible during the computation of outliers.

In the following part, the discussion focuses on 2-dimensional case. However, the discussion can be extended to higher dimensions without loss of generality. In addition, this work assumes $\sigma_{i,1} = \sigma_{j,1} = \sigma_{i,2} = \sigma_{j,2} = \sigma$ to keep the discussion simple.

Algorithm 4.1: UDB Outlier Detection: Naive Approach

Input: $\mathcal{GDB}, D, p, \sigma$

Output: Set of UDB Outliers \mathcal{O}

```

1:  $N \leftarrow$  number of objects in  $\mathcal{GDB}$ ;
2:  $\theta \leftarrow N(1 - p)$ ;
3: for each  $o$  in  $\mathcal{GDB}$  do
4:    $\#D$ -neighbors( $o$ )  $\leftarrow$  0;
5:   for each  $o_i$  in  $\mathcal{GDB}$  do
6:      $\#D$ -neighbors( $o$ )  $+= Pr(o, o_i, D)$ ;
7:     if  $\#D$ -neighbors( $o$ )  $> \theta$  then
8:       mark  $o$  as non-outlier, GOTO next  $o$ ;
9:     end if
10:  end for
11:  mark  $o$  as outlier, add  $o$  to  $\mathcal{O}$ ;
12: end for
13: return  $\mathcal{O}$ 

```

The Naive approach of the distance-based outlier detection uses Nested-loop to compute $\#D$ -neighbors of each object. The $\#D$ -neighbors computation of an object $o_i \in \mathcal{GDB}$ requires computation of the expensive distance probability with every other object in the \mathcal{GDB} until o_i can be decided as an outlier or inlier. In the worst case, this approach requires $O(N^2)$ evaluations of the distance probability, which is very expensive. The Naive approach of the distance-based outlier detection on uncertain data is given in Algorithm 4.1.

4.3 Cell-based Outlier Detection

In this section, we present a cell-based approach to index the dataset objects and to speed up the outlier detection process by pruning the cells as outliers and inliers. The proposed approach identifies and prunes the cells containing only

inliers depending on the cell-bounds on $\#D$ -neighbors. Similarly it can also detect the cells containing only outliers.

4.3.1 Cell-based Pruning

The cell-based technique is proposed to quickly identify and prune the cells containing only inliers. Similarly, it can also detect cells containing outliers like the cell-based approach of Knorr et al. [65]. Since the cell-based approach by Knorr et al. deals with deterministic data only, they considered two cell layers that lie within certain distances from a target cell for its pruning. However, in this work, objects are infinitely uncertain, hence all the cell-layers in the cell-grid need to be considered for the pruning of target cell.

Grid \mathcal{G} Structure

In order to identify distance-based outliers using cell-based technique, mean of each object $o_i \in \mathcal{GDB}$ is mapped to a 2-dimensional space that is partitioned into cells of length l . (The cell length is discussed in Sec. 4.3.5). Let $C_{x,y}$ be any cell in the grid \mathcal{G} , where positive integers x and y denote the cell indices. The layers (L_1, \dots, L_n) of $C_{x,y} \in \mathcal{G}$ are the neighbouring cells of $C_{x,y}$ as shown in Fig. 4.1 and are defined as follows.

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

$$L_2(C_{x,y}) = \{C_{u,v} | u = x \pm 2, v = y \pm 2, C_{u,v} \notin L_1(C_{x,y}), C_{x,y} \neq C_{u,v}\}.$$

$L_3(C_{x,y}), \dots, L_n(C_{x,y})$ are defined in a similar way. The considerable maximum number of layers depends on the position of the target cell in the cell-grid. A cell $C_{x,y}$ in \mathcal{G} can have the maximum number of layers if it exists at the corner of the \mathcal{G} and the minimum number of layers if it exists at the centre of the \mathcal{G} . Let n denotes the maximum number of layers, then the minimum number of layers is given by $\lceil n/2 \rceil$.

Cell Bounds

Like the cell-based approach by Knorr et al. [65], goal of the proposed cell-based technique is to identify and prune cells which are guaranteed to contain

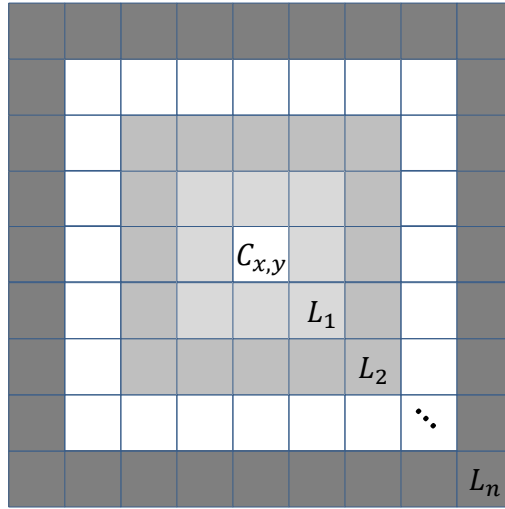


Figure 4.1: Cell layers

only inliers or outliers. A cell $C_{x,y}$ can be pruned as an *outlier cell* if the $\#D$ -neighbors for any object in $C_{x,y}$ according to Def. 4.2 is less than or equal to the threshold θ . Similarly a cell can be pruned as an *inlier cell* if the $\#D$ -neighbors for any object in cell $C_{x,y}$ is greater than the θ . Hence bounds on the $\#D$ -neighbors of $C_{x,y} \in \mathcal{G}$ are defined to prune them. The upper and lower bounds bind the possible $\#D$ -neighbors without expensive object-wise distance computation.

Upper Bound: The upper bound of a cell $C_{x,y}$, $UB(C_{x,y})$, binds the maximum $\#D$ -neighbors in grid \mathcal{G} for any object in cell $C_{x,y}$. Since the Gaussian distribution is infinite, two objects in the same cell may reside at the same coordinate. Hence the maximum $\#D$ -neighbors in $C_{x,y}$ for any object in cell $C_{x,y}$ itself is equal to the number of objects in $C_{x,y}$, denoted by $N(C_{x,y})$.

Similarly, the maximum $\#D$ -neighbors in cells in layer $L_m(C_{x,y}) (1 \leq m \leq n)$ for any object in $C_{x,y}$ can be obtained as follows.

$$\sum_{m=1}^n N(L_m(C_{x,y})) * Pr((m-1)l, D),$$

where $N(L_m(C_{x,y}))$ denotes the number of objects in layer $L_m(C_{x,y})$. Fig. 4.2 shows how the $\alpha = (m-1)l$ values are obtained for computation of the upper bound. Hence $UB(C_{x,y})$ of $C_{x,y} \in \mathcal{G}$ is derived as follows.

$$UB(C_{x,y}) = N(C_{x,y}) + \sum_{m=1}^n N(L_m(C_{x,y})) \times Pr((m-1)l, D). \quad (4.4)$$

Lower Bound: The lower bound of a cell $C_{x,y}$, $LB(C_{x,y})$, binds the minimum $\#D$ -neighbors in grid \mathcal{G} for any object in cell $C_{x,y}$. When two objects in the same cell reside at the opposite corners, the probability that they are D -neighbours takes the minimum value. Hence the minimum $\#D$ -neighbors in $C_{x,y}$ for any object in cell $C_{x,y}$ itself is equivalent to $1 + (N(C_{x,y}) - 1) \times Pr(\sqrt{2}l, D)$.

Similarly, the minimum $\#D$ -neighbors in cells in layer $L_m(C_{x,y})$ ($1 \leq m \leq n$) for any object in $C_{x,y}$ can be obtained as follows.

$$\sum_{m=1}^n N(L_m(C_{x,y})) \times Pr((m+1)\sqrt{2}l, D).$$

Fig. 4.2 shows how the $\alpha = (m+1)\sqrt{2}l$ values are obtained for the lower bounds. Hence $LB(C_{x,y})$ of $C_{x,y} \in \mathcal{G}$ is derived as follows.

$$LB(C_{x,y}) = 1 + (N(C_{x,y}) - 1) \times Pr(\sqrt{2}l, D) + \sum_{m=1}^n N(L_m(C_{x,y})) \times Pr((m+1)\sqrt{2}l, D). \quad (4.5)$$

Lookup Table: The bounds discussed above are required by each $C_{x,y} \in \mathcal{G}$ for pruning. Each bound computation requires evaluation of the costly distance probability, $Pr(\alpha, D)$, and the object counts of respective cell $C_{x,y}$ and its layers $L_m(C_{x,y})$. The number of distance probability computations for the bounds calculation can be reduced by pre-computing $Pr(\alpha, D)$ values for $C_{x,y}$ bounds. Since the $Pr(\alpha, D)$ values are decided only by the α -values and are independent from the locations of $C_{x,y}$, $Pr(\alpha, D)$ values need to be computed only for $\alpha = m\sqrt{2}l$ ($1 \leq m \leq n+1$) and $\alpha = ml$ ($0 \leq m \leq n-1$). The pre-computed values are stored in a lookup table to be used by the cell-based pruning technique.

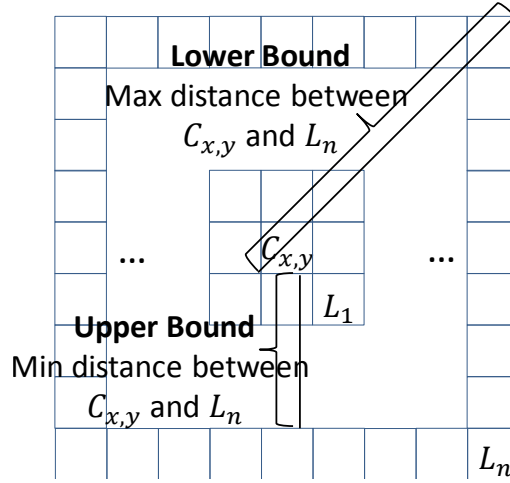


Figure 4.2: Cell and layers bounds

Cell Pruning

Having defined bounds and lookup table, a cell $C_{x,y} \in \mathcal{G}$ can be pruned as an inlier cell or identified as an outlier cell as follows.

If $LB(C_{x,y})$ is greater than θ , $C_{x,y}$ cannot contain outliers. Hence it can be pruned as an inlier cell. On the other hand if $UB(C_{x,y})$ is less than or equal to θ , $C_{x,y}$ is identified as an outlier cell. Lines 1 through 11 in Algorithm 4.2 show the cell-based pruning technique.

4.3.2 Object-wise Bounds Pruning

Although the cell-based technique is very effective, yet it may leave some cells undecided, i.e., they are neither pruned as inlier cells nor are identified as outlier cells. For the pruning of uncertain data objects in such cells an object-wise bounds pruning technique is proposed. This technique helps in the computation of bounds on $\#D$ -neighbors for the un-pruned objects from the cell-based approach. Using this approach, a lot of expensive distance function computations may be avoided.

In this technique, $Pr(\alpha, D)$ is pre-computed for some α values. In this work, $Pr(\alpha, D)$ is computed for several α values between 0 and $D + 3\sigma$. α is chosen in this range because $Pr(\alpha, D)$ values for $\alpha > D + 3\sigma$ are negligibly small and are usually not effective in pruning. The set of pre-computed $Pr(\alpha, D)$ values is denoted by ψ and $n_{bounds} = |\psi|$. These pre-computed values are used for

Algorithm 4.2: UDB Outlier Detection: Cell-based Approach**Input:** \mathcal{GDB}, D, p, l **Output:** Set of distance-based outliers \mathcal{O}

- 1: Create cell grid \mathcal{G} using the min. and max. dataset \mathcal{GDB} objects means and cell length l ;
- 2: Initialize $Count_k$ of each cell $C_k \in \mathcal{G}$;
- 3: Map each object o in \mathcal{GDB} to an appropriate C_k , and increment $Count_k$ by 1;
- 4: $\theta \leftarrow |\mathcal{GDB}|(1 - p)$, $\mathcal{O} = \{\}$; (θ correspond to the threshold)
/*Pruning cells using bounds*/
- 5: **for each** non-empty C_k in \mathcal{G} **do**
- 6: **if** $LB(C_k) > \theta$ **then**
- 7: C_k is an inlier cell, mark C_k green. GOTO Next C_k ;
- 8: **else if** $UB(C_k) \leq \theta$ **then**
- 9: C_k is an outlier cell, add objects of C_k to \mathcal{O} , mark C_k black. GOTO Next C_k ;
- 10: **end if**
- 11: **end for**
/*Object-wise pruning*/
- 12: $\mathcal{O} = \mathcal{O} \cup ObjectWisePruning(\mathcal{G}, D, \theta)$;
/*Unpruned objects processing*/
- 13: **for each** object o_i in non-empty, uncoloured $C_k \in \mathcal{G}$ **do**
- 14: **if** o_i is uncoloured **then** compute $\#D$ -neighbors(o_i) using objects in C_k and higher layers of $C_k \in \mathcal{G}$;
- 15: **if** $\#D$ -neighbors(o_i) $\leq \theta$ **then** o_i is outlier. Add o_i to \mathcal{O} ;
- 16: **end for**
- 17: **return** \mathcal{O} ;

the computation of bounds on $\#D$ -neighbors for the un-pruned objects. These bounds are denoted by $Pr(\alpha, D)_{LB}$ and $Pr(\alpha, D)_{UB}$, respectively. Algorithm 4.3 shows the object-wise bounds pruning.

For example, let $D = 90$ and $\sigma = 10$, then $D + 3\sigma = 120$. Therefore $Pr(\alpha, D)$ values need to be computed for $0 < \alpha \leq 120$. Assuming that $Pr(\alpha, D)$ is pre-calculated for $\alpha = 20, 40, 60, 80, 100, 120$ then $\psi = \{ 0.99, 0.9, 0.75, 0.5, 0.2, 0.001 \}$. If $\alpha = 70$ for o_i and o_j then $Pr(70, D)_{UB} = 0.75$ and $Pr(70, D)_{LB} = 0.5$.

Algorithm 4.3: UDB Outlier Detection: ObjectWisePruning**Input:** \mathcal{G}, D, θ **Output:** Set of distance-based outliers \mathcal{O}

```

1:  $\mathcal{O} = \{\}$ ;
2: for each non-empty uncoloured  $C_k$  in  $\mathcal{G}$  do
3:   for each  $o_i$  in  $C_k$  do
4:     for each  $o_j$  in  $D3\sigma(C_k)$  ( $D3\sigma(C_k)$  corresponds to the cells within
        $D + 3\sigma$  distance of cells  $C_k$ ) do
5:       if  $0 < \alpha \leq D + 3\sigma$  then
6:         Update  $\#D\text{-neighbors}(o_i)_{LB}$  and  $\#D\text{-neighbors}(o_i)_{UB}$  using
           precomputed bounds;
           ( $\#D\text{-neighbors}(o_i)_{LB}$  &  $\#D\text{-neighbors}(o_i)_{UB}$  corresponds the
           lower and upper bounds of  $\#D\text{-neighbors}(o_i)$  respectively.)
7:       end if
8:     end for
9:     if  $\#D\text{-neighbors}(o_i)_{LB} > \theta$  then  $o_i$  is inlier, mark  $o_i$  green. GOTO
       next  $o_i$ ;
10:    else if  $\#D\text{-neighbors}(o_i)_{UB} \leq \theta$  then  $o_i$  is outlier, mark  $o_i$  black.
       Add  $o_i$  to  $\mathcal{O}$ ;
11:    end for
12:  end for
13: return  $\mathcal{O}$ ;

```

4.3.3 Un-pruned Objects Processing and Grid File Index

There may be some undecided objects, i.e., they are neither pruned as inliers nor identified as outliers, even after the cell-based pruning and the object-wise bounds pruning. For all such uncertain objects, nested-loop computation follows. Usually the number of such objects is very small, yet it can be expensive due to the costly distance probability computation. According to the distance probability function, $Pr(o_i, o_j, D)$ is higher when o_i and o_j are close. Hence for an undecided object o_i , if $o_j \in \mathcal{GDB}$ nearer to o_i are chosen earlier for the computation of $\#D\text{-neighbours}$ of o_i , the number of $Pr(o_i, o_j, D)$ computations can be reduced. If an un-pruned object o_i is inlier, it will be pruned by considering only nearer objects. Since the objects are already in grid structure, the grid can be utilized as grid-file index [80] with no additional indexing cost to retrieve nearer objects for the undecided objects. This helps in deciding the un-pruned objects faster than using no index at all. Lines 13 through 16 of Algorithm 4.2 shows the processing of such objects.

4.3.4 Complexity Analysis

We will first analyse the complexity of the UDB outlier detection algorithm (Algorithm 4.2) for the 2D case. Line 1 creates the cell-grid by finding the minimum and maximum means of the dataset $\mathcal{G}DB$ objects and cell length l . It takes $O(N)$ time. Line 2 takes $O(m)$ time, where $m \ll N$ is the total number of cells in the cell-grid \mathcal{G} . Mapping N objects to the \mathcal{G} require $O(N)$ objects evaluations. Line 4 contains only the initializations of variables. The main loop of the algorithm in lines 5-11 is executed for all the cells in the \mathcal{G} . The loop computes the bounds of all the cells in \mathcal{G} , each of which takes $O(n_L)$ time (assuming that there are at-most n_L layers in the \mathcal{G}), because the cell bounds computation require the contribution of all the cell layers in the \mathcal{G} . Hence the overall loop takes $O(mn_L)$ time. Assuming that there are $n' \ll N$ un-pruned objects in un-pruned cells from the cell-based pruning, the object wise pruning takes $O(n'N)$ time because, for the object-wise pruning, the bounds on the $\#D$ -neighbours of each un-pruned object is computed using all the objects in the dataset. Finally, computation of the accurate $\#D$ -neighbours in Lines 13-16 takes $O(nN)$ time, where $n \ll N$ is the number of un-pruned objects from the cell-based and the object-wise prunings. Thus, the average case time complexity of the UDB outlier detection algorithm in 2D is $O(nN + mn_L)$. In the worst case, none of the object is pruned by the cell-based algorithm, hence the worst case time complexity of the UDB outlier detection algorithm in 2D is $O(N^2 + mn_L)$.

However, in the UDB outlier detection algorithm, the major cost lies in the evaluation of accurate $\#D$ -neighbours of the un-pruned objects (Lines 13-16). This cost is so high that it hides the cost of the rest of the algorithm. This is due to the expensive distance probability computation between uncertain objects. Therefore, we give the algorithm complexity in terms of the number of distance probability evaluations. Hence the average case and the worst case time complexities of the UDB outlier detection algorithm in 2D are $O(nN)$ and $O(N^2)$, respectively.

The complexities of the UDB outlier detection algorithm do not change with the increase in dimensions d , as long as only the number of distance probability evaluations are considered for the computation of the algorithm's complexities. Although with the increase in d , the number of grid cells increases exponen-

tially, yet the cost of evaluation of the $\#D$ -neighbours for the un-pruned objects remains dominant and hence the complexities remain same for the average case and the worst case, i.e., $O(nN)$ and $O(N^2)$, respectively for higher dimensional case.

From the above analysis, it is evident that the average case computational complexity of the UDB outlier detection algorithm is lower than the Naive algorithm, which is $O(N^2)$ in terms of the distance probability evaluations. Hence the execution times of the cell-based UDB outlier detection algorithm is far lower than the Naive algorithm. However, with the increase in d , the distance probability computation between uncertain objects becomes so expensive that it becomes impractical to detect outliers using the proposed approach from very high dimensional data.

4.3.5 Discussion: Determination of Values for Parameters D , p and l

Let us begin by stating that there is no universally correct value for parameters D , p or l . Parameter D has an effect on the $\#D$ -neighbors of an object and $\#D$ -neighbors are computed using $Pr(o_i, o_j, D)$ function. Larger D values result in larger $Pr(o_i, o_j, D)$ values and therefore larger $\#D$ -neighbors and vice versa. However very small or very large D value is not recommended as it results in very small or very large $\#D$ -neighbors respectively for all the state set objects and hides the difference between strong and weak outliers. Therefore an appropriate D value must not be too large to cover the entire dataset objects and not too small to cover only the object itself. Hence an appropriate D value may be decided by considering the dataset distribution by the end user.

Since the parameter p is used in the determination of threshold, $\theta = N(1 - p)$, it affects the number of outliers returned by the proposed approaches. Since an outlier occurs rarely, and therefore it is reasonable to select a value of p very close to unity. Consider a dataset of size $N = 1000$, and $p = 0.995$. Given the threshold expression $\theta = N(1 - p)$, where θ is the maximum $\#D$ -neighbors of an outlier object o , this means that the maximum value of $\#D$ -neighbors of o could be 5. If D is very large, very few or no outliers may occur. On the other hand, for very small D many or all dataset objects may be outliers. From experiments, we concluded that p should be close to unity. For example, for

$N = 10^3$, $p = 0.995$ may be appropriate, but for $N = 10^6$, may be far too small. For the latter case, $p = 0.99995$ may be more appropriate.

Varying l has an affect on the performance of the proposed outlier detection approaches rather than the accuracy. Smaller l values are good for cell pruning as they result in tighter bounds, however very small l increases the number of cells in the grid exponentially and the time required for the bounds computation. On the other hand, larger l values result in looser bounds and hence reduce the cell pruning capability. As the number of dimensions d increases, the number of grid cells increases exponentially and therefore larger l values are recommended for higher d . Therefore small l values are recommended for lower dimensions and relatively larger values are recommended for higher dimensions.

4.4 Cell-based Outlier Detection using the Bounded Gaussian Uncertainty

Despite the techniques presented in Sec. 4.3, unbounded nature of the Gaussian distribution prevents from computing outliers very efficiently. Hence in this section an approximate distance-based outlier detection approach using the bounded Gaussian uncertainty is presented. Approximating the Gaussian distribution by the bounded Gaussian distribution enables an approximate but more efficient cell-based pruning technique along with simple object-wise distance and bounds pruning techniques. According to this work's assumption, attributes of uncertain objects follow the Gaussian distribution. Therefore according to the 3-sigma rule there is a 95.45% probability that uncertain objects' attribute values lie within 2 standard deviations of the observed values and 99.73% probability that the values lie within 3 standard deviations of the observed values and so on [88]. Hence the conventional Gaussian distribution can be normalized within certain boundaries to increase efficiency of outlier detection at a small cost of accuracy.

In this section, we derive a distance probability function for the objects following the bounded Gaussian uncertainty. Given a conventional Gaussian function $g_{\vec{\lambda}}(x_1, x_2)$ with mean $\vec{\mu} = (\mu_1, \mu_2)^T$ and co-variance matrix $\Sigma = \text{diag}(\sigma^2, \sigma^2)$, the bounded Gaussian distribution $f_{\vec{\lambda}}(x_1, x_2)$ can be defined following the practise of [108], as follows.

$$f_{\vec{\mathcal{A}}}(x_1, x_2) = \begin{cases} \frac{g_{\vec{\mathcal{A}}}(x_1, x_2)}{\int_{(x_1, x_2) \in o.ur} g_{\vec{\mathcal{A}}}(x_1, x_2) dx_1 dx_2} & (x_1, x_2) \in o.ur \\ 0 & otherwise \end{cases} \quad (4.6)$$

where $o.ur$ denotes the uncertainty region of the bounded Gaussian distribution. This paper assumes that the uncertainty region is a sphere with centre at $(\mu_1, \mu_2)^T$ and radius r . Note that,

$$\int_{o.ur} f_{\vec{\mathcal{A}}}(x_1, x_2) dx_1 dx_2 = 1.$$

When two objects o_i and o_j follow the bounded Gaussian distribution, $Pr(o_i, o_j, D)$ is given as follows.

$$Pr(o_i, o_j, D) = \int_0^r \int_0^{2\pi} \int_0^D \int_0^{2\pi} f_{\vec{\mathcal{A}}_i}(r_1 \cos \theta_1, r_1 \sin \theta_1) \times f_{\vec{\mathcal{A}}_j}(r_1 \cos \theta_1 + r_2 \cos \theta_2, r_1 \sin \theta_1 + r_2 \sin \theta_2) r_1 r_2 d\theta_2 dr_2 d\theta_1 dr_1. \quad (4.7)$$

Hence the uncertain data objects following the conventional Gaussian uncertainty in dataset $\mathcal{GDB} = \{o_1, \dots, o_N\}$ can be approximated by the bounded Gaussian uncertainty. Namely, $\mathcal{GDB}_b = \{o_1, \dots, o_N\}$ denotes a set of objects whose attributes $\vec{\mathcal{A}}_i = (x_{i,1}, x_{i,2})^T$ follow the bounded Gaussian uncertainty with mean $\vec{\mu}_i = (\mu_{i,1}, \mu_{i,2})^T$ and co-variance matrix $\Sigma_i = \text{diag}(\sigma^2, \sigma^2)$ respectively and radius r .

4.4.1 Cell-based Pruning for the Bounded Gaussian

Approximating the Gaussian distribution by the bounded Gaussian distribution enables better cell-based pruning. The proposed technique prunes cells containing only inliers and identify outlier cells just like the cell-based approach for conventional Gaussian distribution. In contrast to computing bounds using many layers in the conventional Gaussian cell-based approach, cell size is set in such a way that a target cell can be pruned by just counting objects within the target cell and its neighbouring layers. Moreover, no pre-computation is required for the cell-based technique using the bounded Gaussian uncertainty.

In order to identify distance-based outliers using the cell-based technique,

mean of each object $o_i \in \mathcal{GDB}_b$ is mapped to a d -dimensional space that is partitioned into cells of length l (The cell length is discussed in Sec. 4.3.5). Let $C_{x,y}$ be a cell in the grid \mathcal{G} , then cells in region $R_1(C_{x,y})$ are those which completely lie within $D - 2r$ distance of the $C_{x,y}$, including the $C_{x,y}$ itself, as shown in Fig. 4.3. Let $n_{R1} = \lfloor \frac{D-2r}{l} \rfloor - 1$, then the region $R_1(C_{x,y})$ is derived as follows.

$$R_1(C_{x,y}) = \{C_{u,v} | u = x \pm n_{R1}, v = y \pm n_{R1}, \\ \sqrt{((|u| + 1)l)^2 + ((|v| + 1)l)^2} < D - 2r, C_{u,v} \neq C_{x,y}\}.$$

The number of cells in the region $R_1(C_{x,y})$ vary depending upon n_{R1} . Note that the $R_1(C_{x,y})$ satisfies the following property.

Property 1: If $C_{u,v} \in R_1(C_{x,y})$, then the objects $o_i \in C_{x,y}$ and $o_j \in C_{u,v}$ are at most $D - 2r$ distance apart.

From property 1, the $o_i \in C_{x,y}$ and the $o_j \in R_1(C_{x,y})$ are guaranteed to be D -neighbours mutually, hence the $Pr(o_i, o_j, D)$ is always equal to 1. Cells in region $R_2(C_{x,y})$ are those which fall within $D + 2r$ distance of the $C_{x,y}$. Let $n_{R2} = \lceil \frac{D+2r}{l} \rceil$, then the region $R_2(C_{x,y})$ is derived as follows.

$$R_2(C_{x,y}) = \{C_{u,v} | u = x \pm n_{R2}, v = y \pm n_{R2}, \\ \sqrt{((|u| - 1)l)^2 + ((|v| - 1)l)^2} < D + 2r, C_{u,v} \notin R_1(C_{x,y}), C_{u,v} \neq C_{x,y}\}.$$

Note that the $R_1(C_{x,y})$ and the $R_2(C_{x,y})$ satisfy following property.

Property 2: If $C_{u,v}$ is neither in $R_1(C_{x,y})$ nor in $R_2(C_{x,y})$ and $C_{u,v} \neq C_{x,y}$, then the objects $o_i \in C_{x,y}$ and $o_j \in C_{u,v}$ are greater than $D + 2r$ distance apart.

From property 2, it can be guaranteed that the $o_i \in C_{x,y}$ and $o_j \in C_{u,v}$ are greater than $D + 2r$ distance apart, hence the $Pr(o_i, o_j, D)$ is always equal to 0. Two more types of cells help in pruning. These cells are named *red cells* and *pink cells* and are denoted by $R_r(C_{x,y})$ and $R_p(C_{x,y})$ respectively. Let $n_{R1}^{diag} = \lfloor \frac{D-2r}{l\sqrt{2}} \rfloor - 1$ denotes number of diagonals within $D - 2r$ distance of a cell $C_{x,y}$.

Then the red and the pink cells are defined as follows.

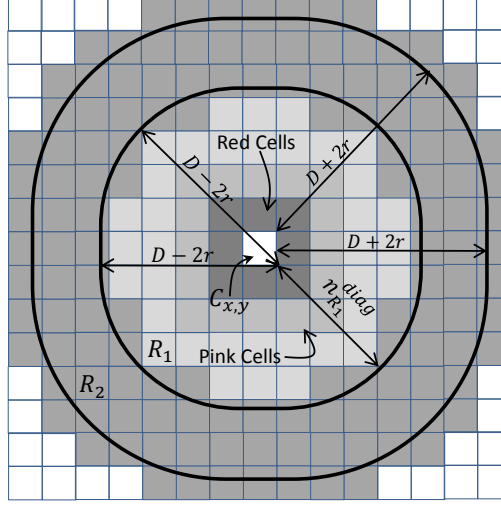


Figure 4.3: Bounded gaussian cell grid

$$R_r(C_{x,y}) = \{C_{u,v} | u = x \pm n_r, v = y \pm n_r, C_{u,v} \neq C_{x,y}\};$$

$$n_r = \min\{n \mid N(C_{x,y}) + \sum_{i=1}^n N(L_i(C_{x,y})) > \theta, 0 < n \leq \lfloor \frac{n_{R_1}^{diag}}{2} \rfloor\}.$$

where $N(L_i(C_{x,y}))$ denotes the number of objects in $C_{x,y}$ and its layer $L_i(C_{x,y})$ ($1 \leq i \leq n$). The n_r value which meets above condition may not exist. If it exists, $R_p(C_{x,y})$ is defined as follows.

$$R_p(C_{x,y}) = \{C_{u,v} | u = x \pm n_p, v = y \pm n_p, C_{u,v} \notin R_r(C_{x,y}), \\ C_{u,v} \neq C_{x,y}, n_r < n_p < n_{R_1}^{diag}\}.$$

For a $C_{x,y}$, $R_r(C_{x,y})$ is chosen in such a way that if the total number of objects in the $C_{x,y}$ and the $R_r(C_{x,y})$ are greater than threshold θ , then they can prune all objects in the $C_{x,y}$, the $R_r(C_{x,y})$ and the $R_p(C_{x,y})$ as inliers. n_r is smaller the better, since the smaller n_r results in larger n_p , hence more cells can be pruned as inliers.

For example, in Fig. 4.3, $n_{R_1}^{diag} = 3$, and assume that $n_r = 1$. Moreover, assume that the total number of objects in $C_{x,y}$ and $L_1(C_{x,y})$ are greater than θ . Then, $n_p = 2$, and all the objects in the $C_{x,y}$, the $L_1(C_{x,y})$ and the $L_2(C_{x,y})$ are

inliers. Note that combined thickness of $C_{x,y}$ and $R_r(C_{x,y})$ is always less than $n_{R_1}^{diag}$, hence they can prune cells in $R_p(C_{x,y})$, just like $C_{x,y}$ can prune $R_1(C_{x,y})$ cells according to Property 3a.

Algorithm 4.4: UDB Outlier Detection: Cell-based Approach (Bounded Gaussian)

Input: $\mathcal{GDB}_b, D, p, l, r, n_{bounds}$

Output: Set of distance-based outliers \mathcal{O}

- 1: Compute n_{bounds} bounds between $D - 2r$ and $D + 2r$;
 - 2: Create cell Grid \mathcal{G} using the min. and max. dataset \mathcal{GDB}_b objects means and cell length l and initialize the count of each cell $C_k \in \mathcal{G}$, $Count_k \leftarrow 0$;
 - 3: Map each object $o \in \mathcal{GDB}_b$ to an appropriate cell C_k , and increment $Count_k$ by 1;
 - 4: $\theta \leftarrow |\mathcal{GDB}_b|(1 - p)$, $\mathcal{O} = \{\}$;
/* Cell-based Pruning */
 - 5: For each non-empty $C_k \in \mathcal{G}$, If $Count_k > \theta$, C_k is an inlier cell, mark C_k green;
 - 6: For each green cell $C_k \in \mathcal{G}$, mark $R_1(C_k)$ cells blue, provided the neighbour has not already been marked green;
 - 7: For each non-empty, uncoloured cell $C_k \in \mathcal{G}$, If $Count_k + \sum_{m \in R_r(C_k)} Count_m > \theta$, then mark C_k , $R_r(C_k)$ and $R_p(C_k)$ blue;
 - 8: **for each** non-empty, uncoloured cell C_k in \mathcal{G} **do**
 - 9: $Count_{k2} \leftarrow Count_k + \sum_{m \in R_1(C_k)} Count_m$;
 - 10: **if** $Count_{k2} > \theta$ **then**
 - 11: C_k is an inlier cell, mark C_k blue. GOTO next C_k ;
 - 12: **else if** $Count_{k2} + \sum_{m \in R_2(C_k)} Count_m \leq \theta$ **then**
 - 13: C_k is an outlier cell, add objects of C_k to \mathcal{O} . GOTO next C_k ;
 - 14: **end if**
 - 15: **end for**
/*Object-wise pruning*/
 - 16: $\mathcal{O} = \mathcal{O} \cup ObjectWisePruning(\mathcal{G}, D, \theta)$;
/*Unpruned objects processing*/
 - 17: **for each** object o_i in non-empty, uncoloured $C_k \in \mathcal{G}$ **do**
 - 18: **if** o_i is uncoloured **then** compute $\#D$ -neighbors(o_i) using objects in C_k and higher layers (layers within $D + 3\sigma$ distance of o_i) of $C_k \in \mathcal{G}$;
 - 19: **if** $\#D$ -neighbors(o_i) $\leq \theta$ **then** o_i is outlier. Add o_i to \mathcal{O} ;
 - 20: **end for**
 - 21: **return** \mathcal{O} ;
-

Property 3: Cell pruning.

- (a) If $N(C_{x,y}) > \theta$, all the objects in $C_{x,y}$ and $R_1(C_{x,y})$ are inliers.
- (b) If $N(C_{x,y}) + N(R_1(C_{x,y})) > \theta$, all the objects in $C_{x,y}$ are inliers.
- (c) If $N(C_{x,y}) + N(R_r(C_{x,y})) > \theta$, all the objects in $C_{x,y}$, $R_r(C_{x,y})$ and $R_p(C_{x,y})$ are inliers.
- (d) If $N(C_{x,y}) + N(R_1(C_{x,y})) + N(R_2(C_{x,y})) \leq \theta$, every object in $C_{x,y}$ is an outlier.

where $N(\cdot)$ denotes the number of objects. Algorithm 4.4 shows the cell-based calculation for the bounded Gaussian distribution.

4.4.2 Simple Object-wise Distance Pruning

Although the cell-based technique is very effective, yet it may leave some cells undecided, i.e., they are neither pruned as the inlier cells nor are identified as the outlier cells. For pruning of uncertain data objects in such cells, an object-wise distance pruning technique is proposed. A similar technique was used in [78] for finding distance between an object and a cluster representative.

Since the uncertainty of objects in \mathcal{GDB}_b is bounded, it can be found whether two objects are within the D -distance only by calculating distance between their observed coordinates. The distance computation in this case is just ordinary Euclidean and is cheap. Since α denotes an ordinary Euclidean distance between observed coordinates of two objects. Let the objects be $o_i, o_j \in \mathcal{GDB}_b$. If $\alpha \leq D - 2r$, it is guaranteed that object o_j lies within the D -distance of o_i . In other words, $Pr(o_i, o_j, D) = 1$. On the other hand, if $\alpha > D + 2r$, then object o_j is guaranteed to lie outside the D -distance of o_i . In this case, $Pr(o_i, o_j, D) = 0$. For example, in Fig. 4.4, $\alpha < D - 2r$ for o_i and o_p and $\alpha > D + 2r$ for o_i and o_q . Therefore $Pr(o_i, o_p, D) = 1$ and $Pr(o_i, o_q, D) = 0$. This object-wise distance pruning helps in computing $\#D$ -neighbours of the un-pruned objects. Using this approach, a lot of expensive distance probability computations may be avoided.

4.4.3 Object-wise Bounds Pruning

Using simple object-wise distance pruning of Sec. 4.4.2, $Pr(o_i, o_j, D)$ can be computed only for the objects whose $D + 2r < \alpha \leq D - 2r$. However in order

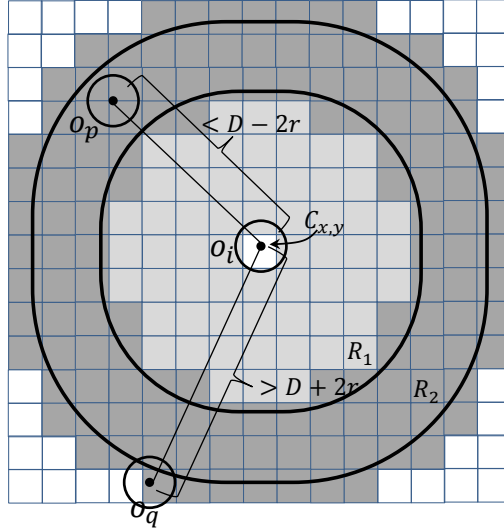


Figure 4.4: Simple object-wise distance pruning

to compute the $\#D$ -neighbours for an object $o_i \in \mathcal{GDB}_b$, the $Pr(o_i, o_j, D)$ needs to be computed for all $o_j \in \mathcal{GDB}_b$ within the regions $R_1(C_{x,y})$ and $R_2(C_{x,y})$. Here $C_{x,y}$ is the cell containing o_i . Hence a technique similar to the object-wise bounds pruning of Sec. 4.3.2 can be used to compute bounds of the $Pr(o_i, o_j, D)$ for $D - 2r < \alpha \leq D + 2r$. The object-wise bounds pruning for the bounded Gaussian is exactly similar to that of the conventional Gaussian with an exception that in the bounded Gaussian, the bounds need to be computed for $D - 2r < \alpha \leq D + 2r$ only.

4.4.4 Un-pruned Objects Processing for the Bounded Gaussian

Un-pruned objects processing for the bounded Gaussian distribution is same as the one presented in Algorithm 4.2. The only difference between the un-pruned objects processing of the conventional Gaussian uncertainty and the bounded Gaussian uncertainty is that the later needs to consider objects in only 2 regions, i.e., $R_1(C_{x,y})$ and $R_2(C_{x,y})$ for the computation of the $\#D$ -neighbours of an un-pruned object $o_i \in C_{x,y}$. In contrast, the conventional Gaussian needs to consider objects in the complete grid \mathcal{G} .

4.4.5 Complexity Analysis

The major part of the UDB(BG) algorithm is same as that of the UDB(CG) algorithm, hence the computational complexity of the UDB(BG) algorithm is also similar to that of the UDB(CG) algorithm. Just like the UDB(CG) algorithm, the major cost of the UDB outlier detection lies in the evaluation of the exact $\#D$ -neighbours of the un-pruned objects. This cost is so high that it hides the cost of the rest of the algorithm. This is due to the expensive distance probability computation between the uncertain objects. Therefore we give the complexities of the UDB(BG) algorithm in terms of the number of distance probability evaluations, which are $O(nN')$ and $O(N^2)$ in 2-dimensional case, for the average case and the worst case respectively; where $n \ll N$ is the number of un-pruned objects and $N' \leq N$ is the number of objects that lie within $D + 2r$ distance of the un-pruned objects. Although the number of distance probability evaluations required for the processing of un-pruned objects in the UDB(BG) algorithm is far less than the UDB(CG), its worst case complexity is still $O(N^2)$ in 2D case, since it assumes that none of the objects are pruned through the cell-based approach and all the dataset objects lie within $D + 2r$ distance of the un-pruned objects.

The complexities of the UDB(BG) algorithm do not change with the increase in dimensions d , as long as only the number of distance probability evaluations are considered for the computation of the algorithm's complexity. Although with the increase in d , the number of grid cells increases exponentially, yet the cost of evaluation of the $\#D$ -neighbours for the un-pruned objects remains dominant and hence the complexities remain same for the average case and the worst case, i.e., $O(nN')$ and $O(N^2)$, respectively for higher dimensional case.

4.5 Discussion: Outlier Detection in High-dimensional Data and Sub-space Outlier Detection

High dimensional data in Euclidean space pose challenges to data mining algorithms, mainly due to the data sparsity. Based on this observation, Bayer et al. in [22] proved that in high dimensional space, all pairs of points are almost

equidistant from one another for a wide range of data distributions and distance functions. Therefore the task of outlier detection has found new specialized solutions for tackling high dimensional data in Euclidean space. These approaches fall under mainly two categories [121], namely 1) Considering subspaces (subsets of attributes) for the outlier detection, 2) Not considering subspaces for the detection of outliers.

In the first category, irrelevant attributes are usually not filtered and all the subspaces (subset of attributes) are included in the outlier detection process. The second category filters the irrelevant attributes and is more concerned with general issues of efficiency and effectiveness. Nevertheless, both types of specialized outlier detection algorithms tackle challenges specific to high dimensional data.

The work presented in this dissertation can be used for the first category, i.e. subspace outlier detection. Subspace outlier detection aims at finding outliers in relevant subspaces that are not outliers in the full-dimensional space (where they are covered by irrelevant attributes) [121]. Predominant issues in subspace outlier detection are: 1) identification of subspaces: Which subspace is relevant and why?, 2) comparability of outlier scores: How to compare outlier results from different subspaces.

First approach for high-dimensional (subspace) outlier detection was given by Aggarwal and Yu [12]. Their approach resembles a grid-based subspace clustering approach but not searching dense but sparse grid cells and report objects contained within sparse grid cells as outliers.

Zhang et al. [118] also studied the identification of outliers in the subspaces. They defined the outlying degree of a point w.r.t. a certain space s in terms of the sum of distances to the k nearest neighbors in this (sub-)space s . For a fixed subspace s , this is the outlier model of Angiulli and Pizzuti [17].

In our work, subspace outlier detection can be incorporated using the subspace distance-based outlier detection approach presented by Knorr et al. in [64] for deterministic data. With the increase in dimensions, the number of subspaces need to be considered for outlier detection increases exponentially and due to this reason identification of relevant subspaces for outlier detection is a known research problem. Authors in [64] proposed to select subspaces at random and search for outliers in them. Using this approach, if a subspace \mathcal{A} does not contain any outlier, then none of its subspaces $\mathcal{B} \subset \mathcal{A}$ can contain an outlier, in

this way a lot of subspaces can be pruned from consideration. Experimentally, authors in [64] found $d = 3$ (d denotes the number of subspace dimensions) as a good starting point for random subspace selection, however further research is needed to identify relevant subspaces.

4.6 Experiments

Extensive experiments are conducted on synthetic and real datasets to evaluate the effectiveness and efficiency of the proposed approaches. In the experiments, Knorr et al. [63] approach of outlier detection on deterministic data is used as the baseline to compare the accuracy of outlier detection. All algorithms are implemented in C++, GNU compiler. All experiments are performed on a system with an Intel Core 2 Duo CPU E8400 3.00GHz CPU and 2GB main memory running Ubuntu 12.04 OS. All programs run in main memory and no I/O cost is considered. Each experiment is performed 3 times and the average values are used in the graphs. We have also used error bars in the graphs, showing the standard error in the execution time measurements of each approach. However, in majority of the graphs they are not visible due to very small standard error in the execution times. Therefore, a table is used to show the percentage of standard error in the execution times.

Pre-computation time is not included in the measurements. Unless specified, the following parameter values are used in experiments: $D = 100$, $\sigma = 10$, $l = 10$, $n_{bounds} = 10$, $r = 2\sigma$, and $p = 0.998$. In the figures, the Knorr et al. [65] approach is denoted by *Knorr* and the approaches proposed in this chapter are denoted by *UDB(CG)* and *UDB(BG)*, respectively.

4.6.1 Datasets

In this work two synthetic and three real datasets are used for experiments, as shown in Figs. 4.5. Unless specified, synthetic datasets, the unimodal Gaussian (UG) and the trimodal Gaussian (TG) are 2-dimensional each and are generated using the BoxMuller method [110]. This method generates pair of independent, standard, normally distributed (zero mean, unit variance) random numbers, given a source of uniformly distributed random numbers. Higher-dimensional unimodal Gaussian datasets (for up to 5 dimensions) are also generated and used

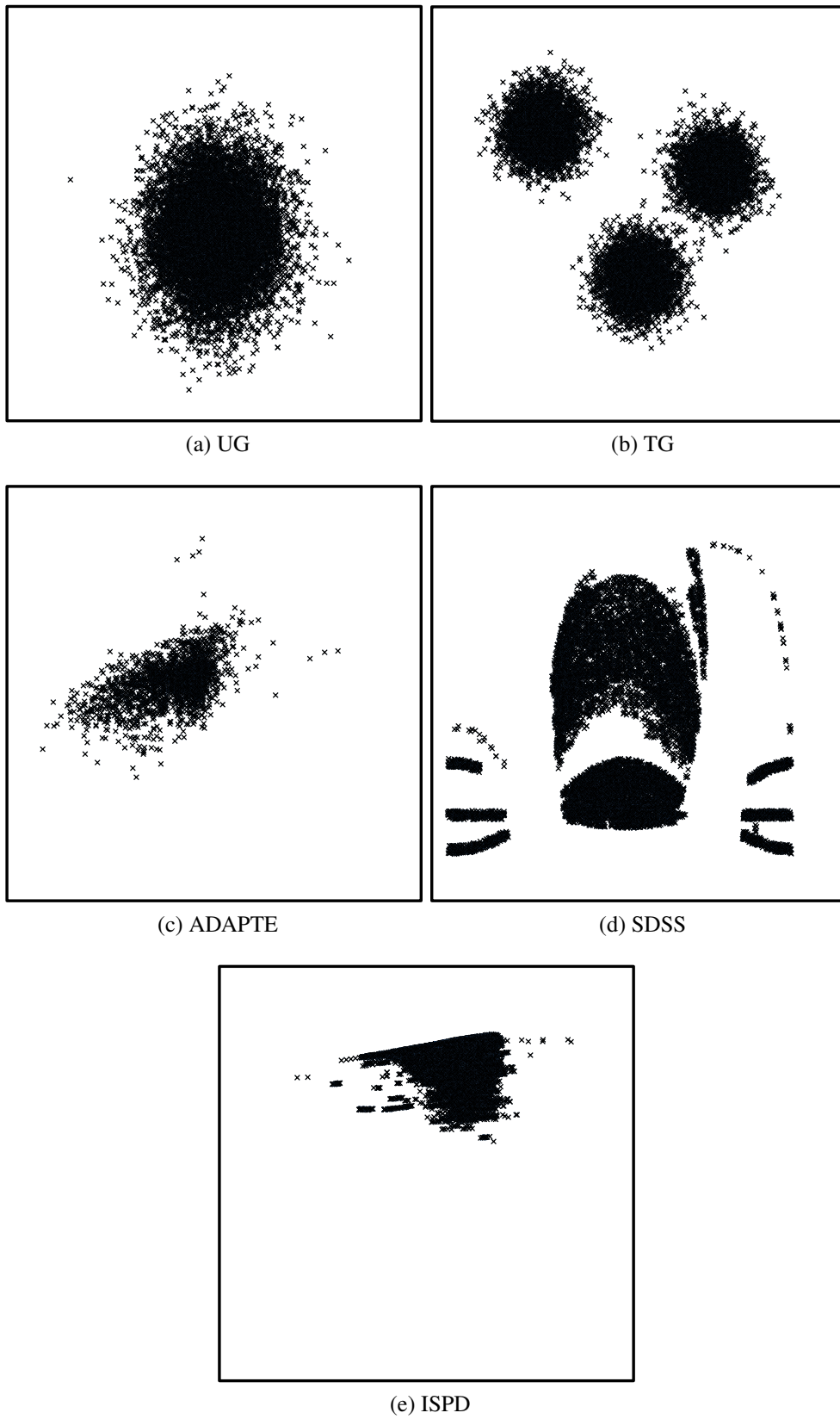


Figure 4.5: Datasets used in the experiments.

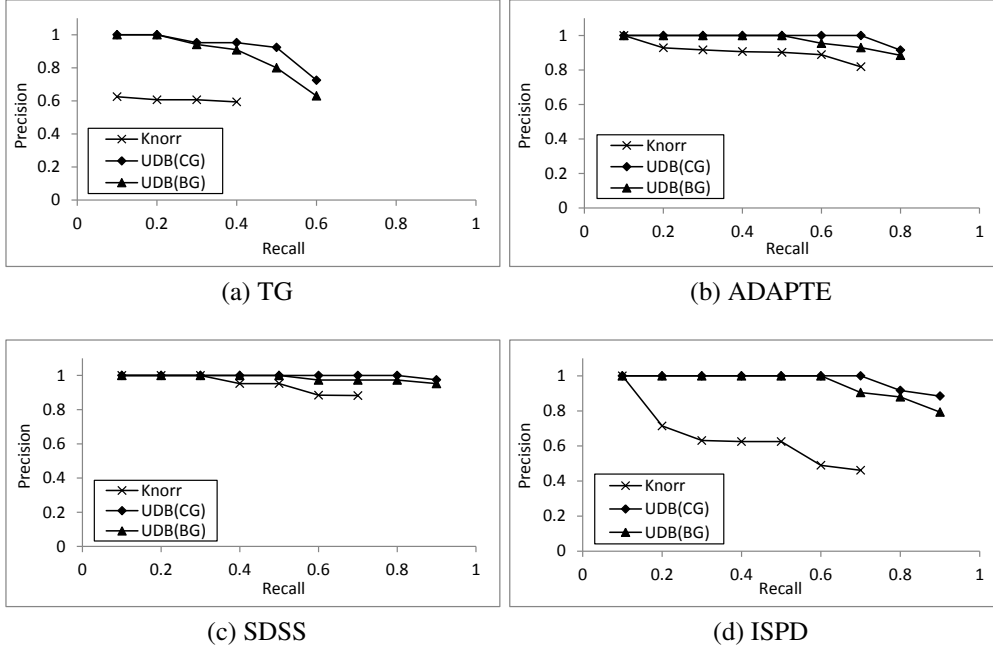


Figure 4.6: Precision-recall trade-off curves ($D = 100$, $\sigma = 10$, $\sigma_p = 20$, $l = 10$, $n_{bounds} = 10$, $r = 3\sigma$, and p is selected in such a way that approximately 0.5 % outliers are returned by all the algorithms.)

for the evaluation of the proposed approaches on higher-dimensional datasets. Unless specified, synthetic datasets consist of 5,000 tuples each.

As for real-world data, three datasets are used: ADAPTE, SDSS and ISPD. ADAPTE and ISPD are obtained from CISL Research data archive [1] and SDSS is obtained from Sloan Digital Sky Survey [3]. ADAPTE consists of about 1,851 maximum and minimum temperature values collected from the National Polytechnic Institute of Mexico and National Meteorological System. SDSS dataset contains 10,136 Right Ascension and Declination coordinates of stars and galaxies. SDSS dataset used in the experiments is a subset of SDSS Data Release 7 (DR7), which includes a huge collection of more than 6 million stars, 8 million galaxies, and 4,500 quasars [3]. The International Surface Pressure Databank (ISPD) dataset consists of 108,015 values of sea level pressure and surface pressure, which is the world's largest collection of pressure observations [2].

All the datasets are normalized to have a domain of $[0,1000]$ on every dimension. For each point z in any dataset, an uncertain object o is created, whose

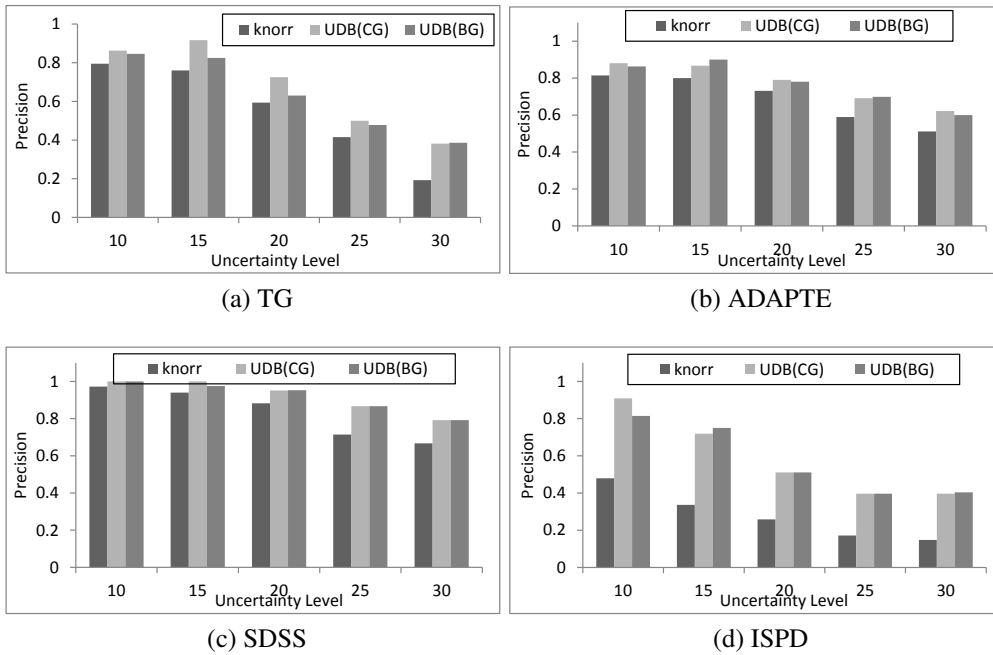


Figure 4.7: Precision with increasing σ_p ($D = 100$, $\sigma = 10$, $l = 10$, $n_{bounds} = 10$, $r = 3\sigma$, and p is selected in such a way that approximately 0.5 % outliers are returned by all the algorithms.)

uncertainty is given by the Gaussian distribution with mean z and standard deviation σ in all the dimensions.

4.6.2 Accuracy

Firstly, experiments are performed to evaluate the accuracy of the proposed approaches. Since there are no known approaches for the distance-based outlier detection on uncertain data, the deterministic distance-based outlier detection approach given by Knorr et al. in [65] is used as a baseline. Since the outliers are not known, for both the synthetic and the real datasets, the baseline approach is used to determine the outliers on the original datasets. The results obtained from the baseline approach are used as the ground truth. In order to judge the accuracy of the proposed approaches, the precision and recall are measured on the perturbed dataset for the baseline approach and the proposed approaches. The precision is defined as the ability of the algorithm to present only true outliers. The recall is defined as the ability of the algorithm to present all true outliers. In order to compute the precision and recall, the $\#D$ -neighbors are computed

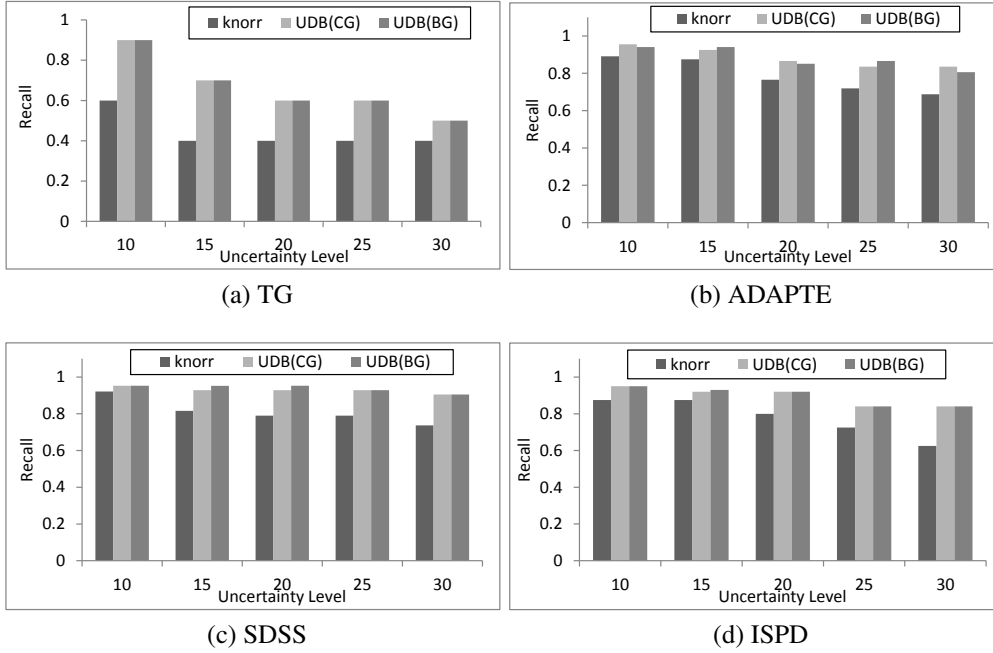


Figure 4.8: Recall with increasing σ_p ($D = 100$, $\sigma = 10$, $l = 10$, $n_{bounds} = 10$, $r = 3\sigma$, and p is selected in such a way that approximately 0.5 % outliers are returned by all the algorithms.)

for the outliers returned by each of the approach used in the experiments. The outliers are then listed in an increasing order of their $\#D$ -neighbors for the computation of their precision and recall. The perturbed dataset is obtained by adding normal random numbers with zero mean and standard deviation σ_p to each of the tuple values of the original dataset. The σ_p was varied from 10 to 30 (with a step of 5) to generate perturbed datasets of five different levels. Experiments show that the proposed approaches are superior than the baseline approach, since they do not degrade quite as much with increasing uncertainty.

Unless specified, the following parameter values are used for the experiments in this subsection: $D = 100$, $\sigma = 10$, $\sigma_p = 20$, $l = 10$, $n_{bounds} = 10$, $r = 3\sigma$, and p is selected in such a way that the baseline approach and the proposed approaches return approximately 0.5% outliers.

Firstly the precision-recall trade-off curves are presented for the different datasets. In all the graphs in Fig. 4.6, the trade-off curves are higher for the proposed approaches, showing the higher precision and recall of the proposed approaches. In Figs. 4.6b and 4.6c, the precision-recall curves are somewhat

closer for all three approaches due to the presence of obvious outliers in these datasets. Addition of perturbation could not change a lot of outliers in these datasets. Yet the recall, i.e. the number of true outliers retrieved, of the proposed approaches is better than the baseline approach in both the datasets. The number of false positive outliers returned by the baseline approach in Figs. 4.6a and 4.6d is very large. This can be observed from the very low precision of the trade-off curves of the baseline approach. On the other hand, the trade-off curves for the proposed approaches are far higher, which shows the higher accuracy of the proposed approaches.

The precision of the proposed approaches is also tested with increasing level of uncertainty. From Fig. 4.7, it is clear that the precision falls with increasing uncertainty level. Moreover, the precision results for the proposed approaches are superior than the results of the baseline approach for all the uncertainty levels in Fig. 4.7. Furthermore, the difference between the baseline approach and the proposed approaches increases with increasing uncertainty. Please note the sharp decrease in precision in Fig.4.7d. Since the dataset of Fig. 4.7d is very large and dense, addition of even low level of perturbation produced a lot of false positive outliers in case of the baseline approach. On the other hand, the proposed approaches produced better precisions for all the datasets. Similar results are illustrated for recall in Fig. 4.8. In all four plots of Fig. 4.8, the recall is somewhat consistent with increasing uncertainty level for the proposed approaches. Which proves that the proposed approaches are capable of retrieving true outliers, even from the noisy data.

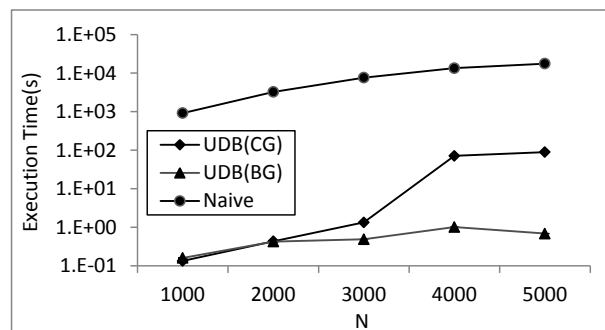


Figure 4.9: Naive vs. proposed Approaches

Table 4.2: Standard error percentage in execution times

N	Naive		UDB(CG)		UDB(BG)	
	Avg.Exec. Time(s)	Std. Error(%)	Avg.Exec. Time(s)	Std. Error(%)	Avg.Exec. Time(s)	Std. Error(%)
1000	918.355	0.00038	0.13433	0.40520	0.15967	0.34092
2000	3236.21	0.00248	0.43133	0.12619	0.424	0
3000	7622.91	9.44E-05	1.34	0	0.48833	0.05573
4000	13460.5	0.00020	71.4207	0.00101	1.01167	0.02690
5000	17493.9	0.00031	88.9793	0.00061	0.68666	0.03964

4.6.3 Efficiency

In this subsection, experiments are conducted to evaluate efficiency of the proposed outlier detection approaches presented in sections 4.3 and 4.4. The time taken by the Naive approach is too high. It takes several hours even on the smallest sample (of 5000 tuples) of the synthetic dataset as can be observed from Fig. 4.9. On the other hand, the proposed approaches, i.e., the UDB(CG) and the UDB(BG) are several times faster than the Naive approach. Note that the UDB(BG) approach require far less time to detect outliers than the UDB(CG), since the computation of $\#D$ -neighbors in the UDB(BG) is far less expensive than the UDB(CG) due to the reasons discussed in Sec. 4.4.

As stated previously, each experiment is performed 3 times and the average values are used in the graphs. Besides, error bars are also used in the graphs, showing the standard error in the execution time measurements of each approach. However, due to very small standard error in the execution times, it is not visible in the figures. Hence for the Fig. 4.9, we have used a table to present the average execution times and the standard error percentages used to plot the graphs in Fig. 4.9. Due to the usage of dedicated computers for the experiments, the deviations in execution times is very small as can be observed from Table 4.2. Hence, in the rest of the experiments only figures are used for the evaluation of the proposed approaches.

Graphs in Fig. 4.10 show the effect of varying cell lengths on the execution times of the proposed approaches. The smaller cell lengths l require the lower execution times and vice versa, which can be observed from Fig. 4.10, specially the UDB(BG) curves. Smaller l values are good for cell pruning as they result in tighter bounds, however very small l increases the number of cells in the grid

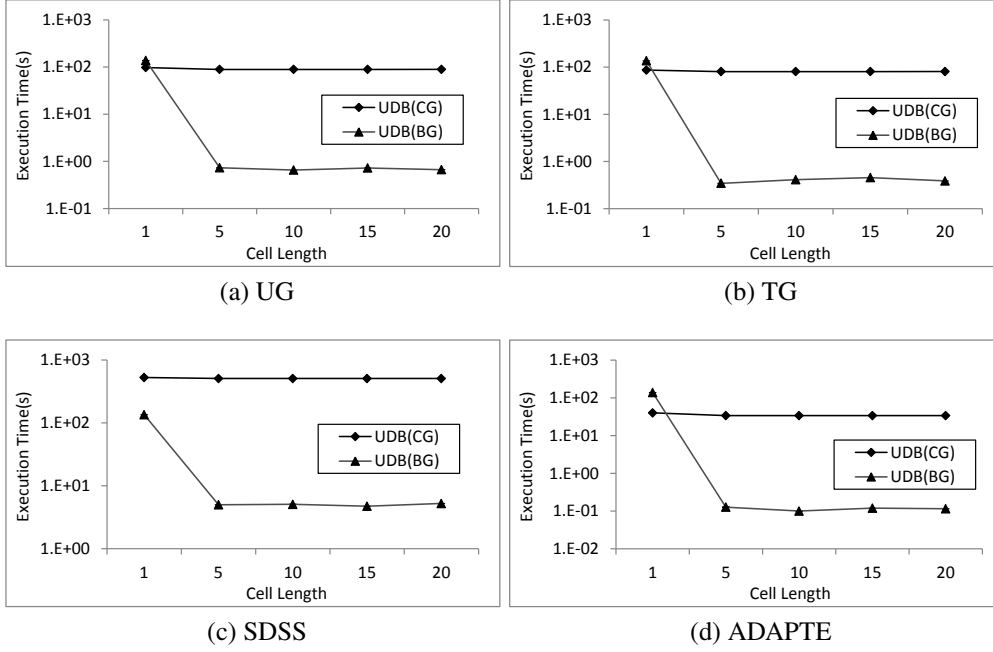


Figure 4.10: Varying parameter l ($D = 100$, $\sigma = 10$, $n_{bounds} = 10$, $r = 2\sigma$, $p = 0.998$)

exponentially and the time required for the bounds computation. On the other hand, larger l values result in looser bounds and hence reduce the cell pruning capability. Moreover, as the number of dimensions d increases, the number of grid cells increases exponentially. Therefore, smaller l values are recommended for lower dimensions and relatively larger values are recommended for higher dimensions. Furthermore it can be observed from the graphs in Fig. 4.10 that the UDB(BG) approach is more efficient than the UDB(CG) approach. The sharp increase of the UDB(BG) curves at cell length 1 is due to the exponential increase in the number of cells in grid.

Next, experiments are performed by varying the dataset objects' uncertainty which is denoted by σ . As σ increases, the uncertainty of the dataset objects also increases. This increase in uncertainty results in smaller $Pr(o_i, o_j, D)$ values even if o_i and o_j are located nearby. Hence the number of distance function evaluations required increases for un-pruned objects during the processing of un-pruned objects, which results in higher execution times as can be observed from Fig. 4.11. However in Fig. 4.11b, the fall in execution time of the UDB(CG) curve is due to the decrease in number of un-pruned objects. In

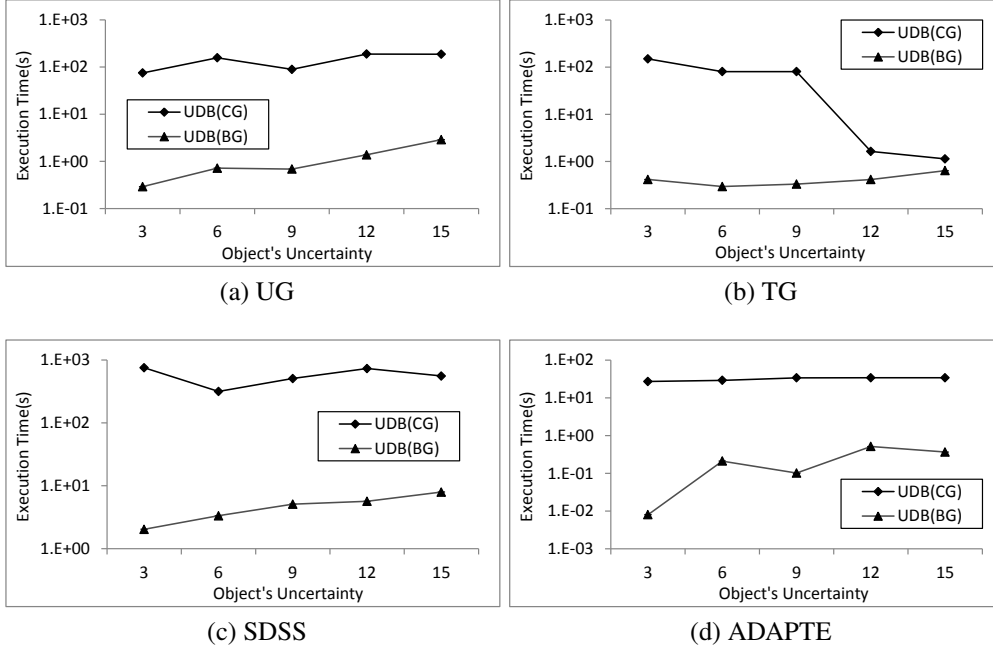


Figure 4.11: Varying object's uncertainty σ ($D = 100$, $l = 10$, $n_{bounds} = 10$, $r = 2\sigma$, $p = 0.998$)

our proposed approaches, the main execution cost lies in the processing of un-pruned objects (This phenomenon is discussed in Fig. 4.16). Hence the decrease in the execution time of the UDB(CG) curve is due to the decrease in number of un-pruned objects with the increase in the dataset objects' uncertainty.

Fig. 4.12 shows the effect of varying the distance parameter D . Increase in D results in an increase in D -neighbours of dataset objects. As a result, objects are more easily pruned as inliers, bringing down execution time of the overall algorithm for larger values of D . Unusual curves in Fig. 4.12d is due to the variation in the number of un-pruned objects for $D = 100$ and $D = 120$.

In Fig. 4.13, the number of outliers are varied by varying the parameter p . As can be observed from the graphs in Fig. 4.13, increase in p results in decrease in execution times of the algorithms. This is due to the fact that increase in p , results in decrease of the threshold value θ . Hence the dataset objects are pruned more easily, bringing down the algorithm execution time. Unusual curves in Fig. 4.13d is again due to the variation in the number of un-pruned objects for $p = 0.998$. In experiments, the number of un-pruned objects are quite small or even zero for some parameter values, resulting in dramatic decrease in execution

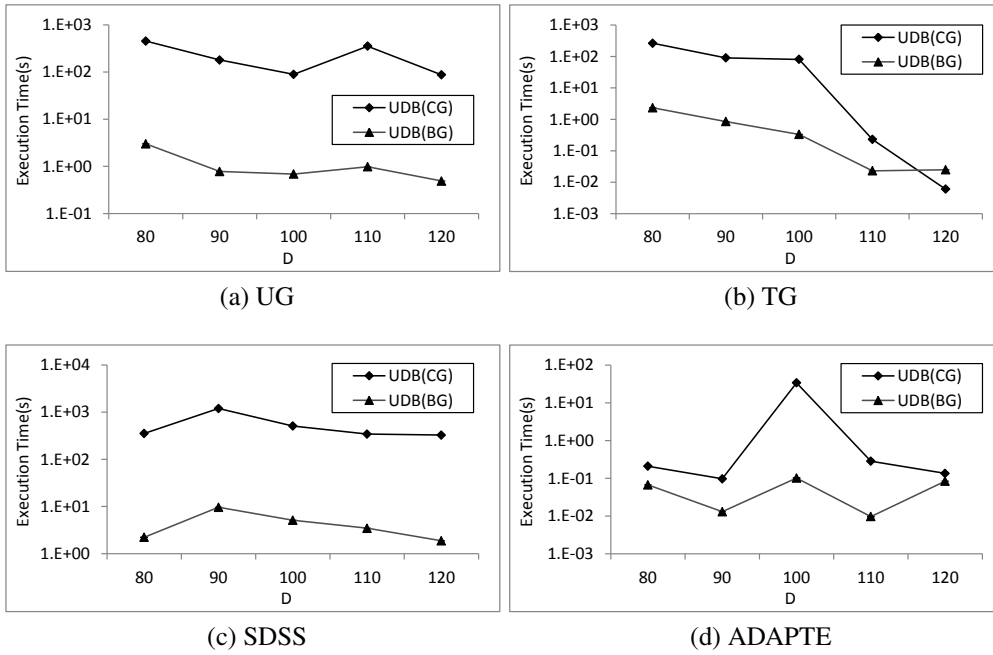


Figure 4.12: Varying parameter D ($l = 10, \sigma = 10, n_{bounds} = 10, r = 2\sigma, p = 0.998$)

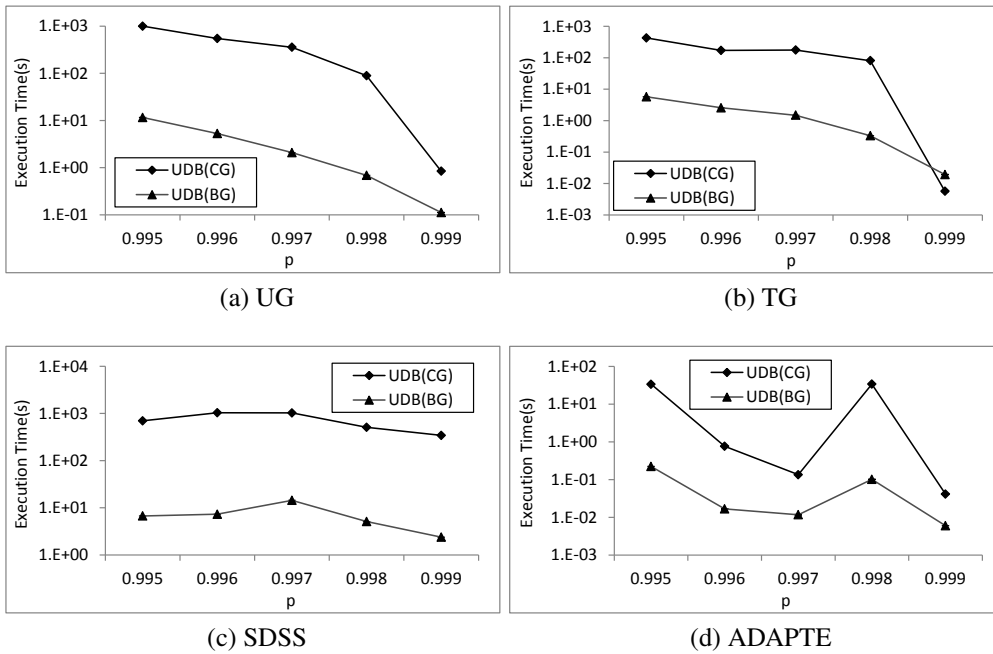


Figure 4.13: Varying parameter p ($D = 100, l = 10, \sigma = 10, n_{bounds} = 10, r = 2\sigma$)

times of the algorithms, while in some experiments the number of un-pruned objects are quite large for some parameter values, resulting in sharp increase in the execution times of the algorithms.

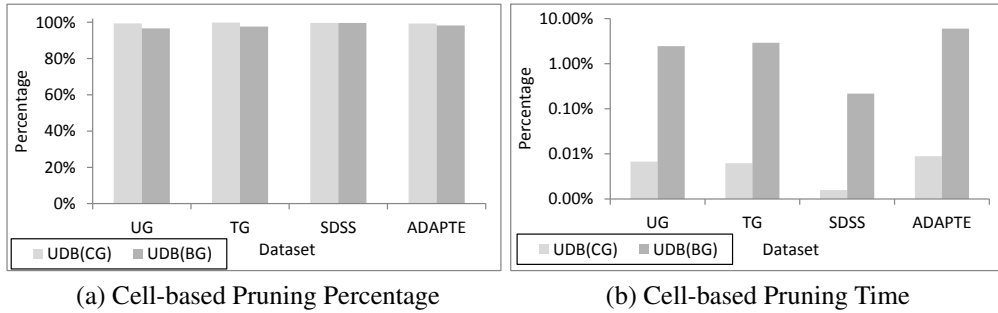


Figure 4.14: Cell-based pruning

Next we present some graphs discussing the effect of different pruning approaches for both the proposed approaches i.e., the UDB(CG) and the UDB(BG). Fig. 4.14a shows the percentage of objects pruned by the cell-based approach for different datasets. From the figure, it is very clear that the cell-based approach is very effective and capable of pruning more than 95% of the dataset objects in all the datasets. Fig. 4.14b shows the percentage of the execution time taken by the cell-based approach by the different datasets.

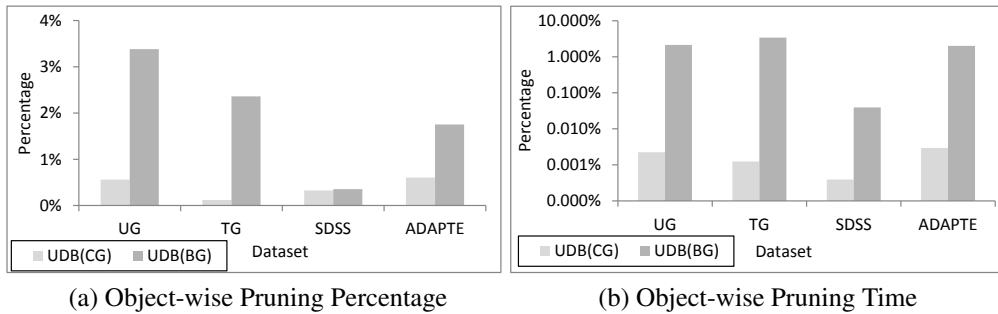


Figure 4.15: Object-wise pruning

Fig. 4.15 shows the percentage of objects pruned and the time required, respectively by the object-wise pruning approach. Since the object-wise pruning approach is executed after the cell-based pruning approach, it is capable of pruning very small percentage of object. Fig. 4.16a shows the percentage of un-pruned objects. Since the un-pruned objects require exact $\#D$ -neighbors

evaluation, which is computationally very expensive, the percentage of algorithms' time required by both the approaches is quite high, as can be observed from Fig. 4.16b. Although the percentage of un-pruned objects is very small and the percentages of objects pruned by the cell-based and the object-wise pruning approaches is quite large, however the percentage of time taken by the pruning approaches is quite small as compared to the un-pruned objects' processing time.

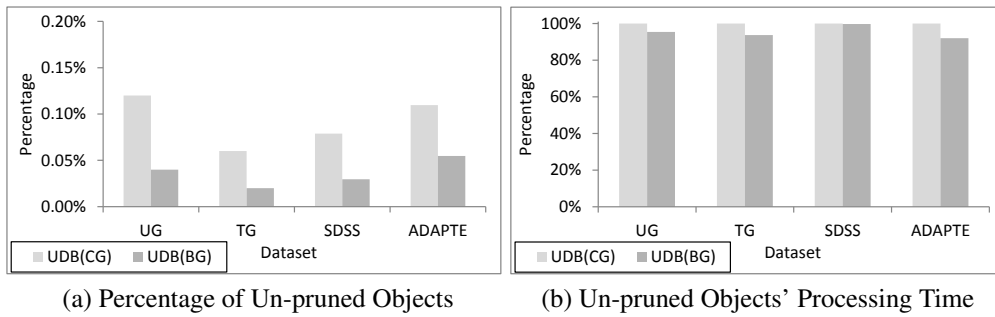


Figure 4.16: Un-pruned objects' processing

Next, we perform experiments by varying the number of dimensions d . We varied the number of dimensions from 2 to 5 with a step of 1. For this experiment, dataset UG of size 500 tuples is used. As d increases the number of cells in the grid increases exponentially and the memory and the processing time required to hold and process the large number of cells, respectively increases dramatically. Therefore, the proposed cell-based approach is not much effective for the high-dimensional data as can be observed from Fig. 4.17. Moreover, from Fig. 4.17 we can observe that as the number of d increases, the difference in execution times of the Naive approach and the CUDB(CG) approach decreases. This is due to the fact that with the increase in d , objects get sparse and their $\#D$ -neighbors become very small. Hence, the cell-based approach can not prune objects in higher dimensional data resulting in an increase in un-pruned objects. For very high d , none of the objects is pruned by the cell-based approach and the cell-based approach become as expensive as that of the Naive approach.

We also performed experiments by varying the number of dimensions d and the parameter D in parallel. In the experiments shown in Fig. 4.17, parameter D was kept constant, due to which the number of outliers increases dramatically

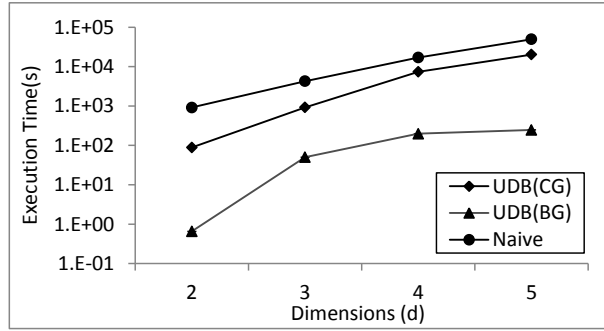


Figure 4.17: Vary dimensions d ($N = 500$, $D = 100$, $l = 10$, $\sigma = 10$, $n_{bounds} = 10$, $r = 2\sigma$ and $p = 0.997$)

with the increase in d . With the increase in d , objects get sparse and the parameter D must be increased accordingly to find the limited number of outliers. Hence in Fig. 4.18, we increased the parameter D with the increase in d , in proportion to the average distance between the dataset objects. By doing so, we obtained almost same percentage of outliers from all the dimensions. Moreover, the execution times of all the approaches fall dramatically as can be observed from Fig. 4.18. In addition, the difference between the execution times of the Naive approach and the CUDB(CG) approach continues to decrease with the increase in d . This is mainly due to the exponential increase in the number of cells with the increase in d , whose processing require a major percentage of the overall algorithm execution time.

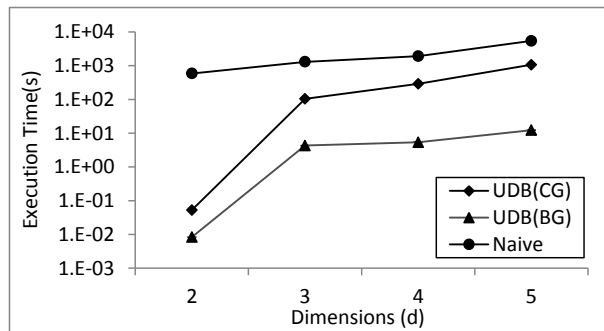


Figure 4.18: Vary dimensions d and parameter D ($N = 500$, $l = 10$, $\sigma = 10$, $n_{bounds} = 10$, $r = 2\sigma$ and $p = 0.997$)

Finally, Fig. 4.19 compares the pre-computation times of the CUDB(CG) and the CUDB(BG) approaches. Since the CUDB(CG) approach needs to compute two types of bounds, i.e., $Pr(\alpha, D)$ values for cell bounds and object-

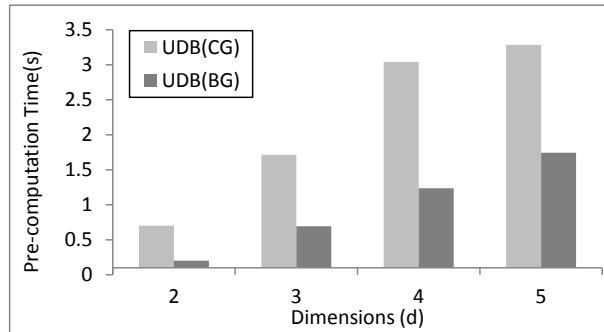


Figure 4.19: Pre-computation time

wise bounds, the pre-computation cost of the CUDB(CG) is higher than the CUDB(BG). Furthermore, the pre-computation time increases with the increase in number of dataset's dimensions. This is due to the fact that the distance probability computation $Pr(o_i, o_j, D)$, which is required for pre-computation, becomes expensive with the increase in dimensions.

4.7 Summary

In this chapter, two approaches of distance-based outlier detection on uncertain datasets are proposed. Firstly a cell-based approach of distance-based outlier detection on uncertain objects following the conventional Gaussian uncertainty is proposed. Secondly an approximate cell-based approach of outlier detection using the bounded Gaussian uncertainty is proposed to increase the efficiency of outlier detection. Experiments prove that the accuracy of the proposed UDB outlier detection approaches (UDB(CG) and UDB(BG)) is far higher than the baseline algorithm. The cell-based pruning is quite effective and is able to prune more than 95% objects in almost all the experiments, due to which the UDB(CG) and the UDB(BG) are several times faster than the Nave approach. In addition the UDB(BG) saves a lot more computation time than the UDB(CG) at a small loss of accuracy.

Chapter 5

Tok- k Outlier Detection on Uncertain Data

The main goal of the research in this chapter is to find top- k distance-based outliers or in other words to obtain k strongest outliers along with their ranking from uncertain datasets. In the outlier detection approach presented in chapter 4 and in other existing works on outlier detection from uncertain data [13, 60, 77, 112], an object can be either classified as outlier or inlier. Since there is no universally agreed definition of outliers, different algorithms return different outliers depending upon the combination of parameter values. Some combinations return a very few while others return a lot of outliers. Moreover, no outlier ranking is available and users are unable to differentiate between strong and weak outliers. Hence in this chapter we present a top- k approach of distance-based outlier detection on uncertain data.

5.1 Overview

Existing algorithms on outlier detection from uncertain data [13, 60, 77, 112] can only perform binary classification, i.e., they can either classify an object as an inlier or an outlier. However, end users are usually interested in strong outliers and their ranking. Hence in this work, we present a top- k approach of distance-based outliers.

To obtain the top- k distance-based outliers from uncertain datasets efficiently we have proposed a novel data structure, PC-list (populated-cells list). The PC-

list is a sorted list of non-empty cells of a d -dimensional grid, where grid is used to index data objects [80]. Using PC-list, the top- k outlier detection algorithm needs to consider only a fraction of the dataset objects and hence quickly identifies candidate objects for top- k outliers. Finally exact outlier score ($\#D$ -neighbors) is computed for each candidate object to find the top- k outliers and their ranking. Furthermore, two approximate top- k outlier detection algorithms are presented in this work to increase the efficiency of the top- k outlier detection algorithm. The first approximate algorithm, approximates only the candidate objects' $\#D$ -neighbors. According to our distance probability function, major contribution in $\#D$ -neighbors computation of an object is made by the nearer objects. Hence the first approximate algorithm, approximates the candidate objects' $\#D$ -neighbors computation by considering only the objects which can affect the $\#D$ -neighbors in a dramatic way. The second approximate algorithm makes use of the bounded Gaussian uncertainty to increase the efficiency of the top- k outlier detection algorithm. The exact top- k outlier detection approach is denoted by $kUDB(CG)$ and the approximate approaches are denoted by $kUDB(Approx)$ and $kUDB(BG)$, respectively in the rest of the dissertation.

5.2 Problem Formulation

Since the focus of this work is the detection of top- k distance-based outliers from uncertain data and to obtain their ranking, the definition 4.2 of distance-based outlier detection on uncertain data given in chapter 4 can be modified for the top- k outliers as follows.

Definition 5.1 *The top- k distance-based outliers are the k uncertain objects in the dataset \mathcal{GDB} for which the expected number of objects $o_i \in \mathcal{GDB}$ lying within their D -distance ($\#D$ -neighbors) is smallest.*

The objects that lie within the D -distance of o_i are called its D -neighbors, and the set of the D -neighbors of o_i and the number of D -neighbours are denoted by $DN(o_i)$ and $\#D\text{-neighbors}(o_i)$ or $\#D(o_i)$, respectively. In order to find the top- k distance-based outliers in \mathcal{GDB} , the $\#D$ -neighbors of candidate outlier objects needs to be computed which requires the computation of distance probability. This distance probability is computed using the difference

between two uncertain objects, which is given by another distribution known as the Gaussian difference distribution [114].

Let $\vec{\mathcal{A}}_i$ and $\vec{\mathcal{A}}_j$ be two independent d -dimensional normal random vectors with means $\vec{\mu}_i = (\mu_{i,1}, \dots, \mu_{i,d})^T$ and $\vec{\mu}_j = (\mu_{j,1}, \dots, \mu_{j,d})^T$ and diagonal covariance matrices $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,d}^2)$ and $\Sigma_j = \text{diag}(\sigma_{j,1}^2, \dots, \sigma_{j,d}^2)$, respectively. Then $|\vec{\mathcal{A}}_i - \vec{\mathcal{A}}_j| = \mathcal{N}(\vec{\mu}_i - \vec{\mu}_j, \Sigma_i + \Sigma_j)$ [114]. Let $Pr(o_i, o_j, D)$ denotes the probability that $o_j \in DN(o_i)$. Then,

$$Pr(o_i, o_j, D) = \int_R \mathcal{N}(\vec{\mu}_i - \vec{\mu}_j, \Sigma_i + \Sigma_j) d\vec{\mathcal{A}}, \quad (5.1)$$

where R is a sphere with centre $(\vec{\mu}_i - \vec{\mu}_j)$ and radius D . Please refer to 4.1 for the 2-dimensional derivation for $Pr(o_i, o_j, D)$. $Pr(o_i, o_j, D)$ expressions for higher dimensions can be derived using Eq. 5.1.

This work assumes that the attributes of uncertain objects are independent and the uncertainty of objects (standard deviation) is uniform in all dimensions, hence $\sigma_{i,1} = \sigma_{j,1} = \sigma_{i,2} = \sigma_{j,2} = \sigma$, and let $\alpha^2 = \alpha_1^2 + \alpha_2^2$. This results in a non-correlated diagonal covariance matrices, i.e., $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \sigma_{i,2}^2)$ and $\Sigma_j = \text{diag}(\sigma_{j,1}^2, \sigma_{j,2}^2)$ for the distance probability expression, $Pr(o_i, o_j, D)$. On the other hand, if the attributes of uncertain objects are dependent there exists a correlation between them and it results in a correlated covariance matrix. Appendix B shows that the series of transformations is always possible to find alternative coordinates of an object, which eliminates the correlation among an object's coordinates. Hence resulting in a diagonal covariance matrix whose variance is uniform in all dimensions, which is consistent with the proposed probability function. Thus the proposed solution is equally applicable for the objects whose attributes are correlated. In the light of above assumptions, the 2-dimensional expression for $Pr(o_i, o_j, D)$ is given as follows.

$$Pr(o_i, o_j, D) = \frac{1}{4\pi\sigma^2} \int_0^D \int_0^{2\pi} \exp\left\{\frac{-1}{4\sigma^2} (r^2 - 2\alpha r \cos \theta + \alpha^2)\right\} r d\theta dr. \quad (5.2)$$

Note that $Pr(o_i, o_j, D)$ only depends on α^2 and not on the coordinates of o_i and o_j . Hence $Pr(o_i, o_j, D)$ is denoted by $Pr(\alpha, D)$ when there is no confusion, where α denotes the ordinary euclidean distance between the means of

uncertain objects. Computing this probability is usually very costly, and needs to be avoided as much as possible during the computation of outliers.

The Naive approach of the top- k outlier detection given in Algorithm 5.1 uses Nested-loop. In order to find whether an object $o_i \in \mathcal{GDB}$ is a top- k outlier, its $\#D$ -neighbours ($\#D(o_i)$) are computed. Computation of $\#D(o_i)$ for an object $o_i \in \mathcal{GDB}$ requires N expensive distance probability evaluations. During the computation of $\#D(o_i)$, if it becomes greater than threshold θ , o_i is an inlier and the computation of $\#D(o_i)$ is stopped. On the other hand, if $\#D(o_i)$ is less than or equal to θ , o_i is added to the candidate list of outliers \mathbb{C}_{obj} , along with its $\#D$ -neighbours. The \mathbb{C}_{obj} is kept sorted in ascending order of $\#D$ -neighbours and the k objects in it with the lowest $\#D$ -neighbours are selected as outliers. In the worst case, this approach requires $O(N^2)$ evaluations of the costly distance probability, which is computationally very expensive.

Algorithm 5.1: k UDB Outlier Detection: Naive Approach

Input: \mathcal{GDB} , D , k

Output: Top- k Distance-based Outliers

- 1: $N \leftarrow |\mathcal{GDB}|$, $\theta \leftarrow \infty$, $\mathbb{C}_{obj} \leftarrow \phi$
 - 2: **for each** o_i in \mathcal{GDB} **do**
 - 3: $\#D(o_i) \leftarrow 0$; ($\#D$ -neighbours of o_i)
 - 4: **for each** o_j in \mathcal{GDB} **do**
 - 5: $\#D(o_i)+ = Pr(o_i, o_j, D)$;
 - 6: **if** $\#D(o_i) > \theta$ **then** GOTO next o_i ;
 - 7: **end for**
 - 8: Insert o_i and its $\#D(o_i)$ into \mathbb{C}_{obj} (Keep \mathbb{C}_{obj} sorted of $\#D(o_i)$);
 - 9: **if** $|\mathbb{C}_{obj}| > k$ **then**
 - 10: Set $\theta = \#D(o')$, where o' is the k^{th} object in \mathbb{C}_{obj} ;
 - 11: Remove all $o'' \in \mathbb{C}_{obj}$, such that $\#D(o'') > \theta$;
 - 12: **end if**
 - 13: **end for**
 - 14: **return** \mathbb{C}_{obj} ;
-

5.3 PC-list-based Outlier Detection

The Naive approach requires a lot of computation time to detect top- k outliers even from a small dataset due to the costly distance probability calculation. To overcome this problem a populated-cells list (PC-list) based approach of the top-

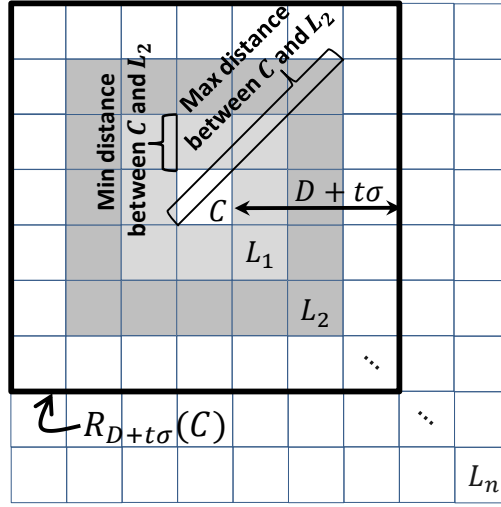


Figure 5.1: Cell layers and bounds

k distance-based outlier detection is proposed. PC-list is an array of non-empty cells of a d -dimensional grid, where the grid is used to index the dataset objects. The PC-list helps in the detection of top- k distance-based outliers by identifying the grid cells containing candidate outliers.

Lemma 5.1 *Let $o_i, o_j \in \mathcal{GDB}$ be two d -dimensional uncertain data objects following the Gaussian distribution and α denotes an ordinary Euclidean distance between the means of o_i and o_j . Then for $t \in \mathcal{R}$, denoting the number of standard deviations required to enclose a large probability (say $> 99\%$) of a d -dimensional Gaussian difference distribution, following statements hold.*

1. *If $\alpha \leq D - t\sigma'$, $Pr(o_i, o_j, D) \approx 1$.*
2. *If $\alpha \geq D + t\sigma'$, $Pr(o_i, o_j, D) \approx 0$.*

where σ' is the standard deviation of the Gaussian difference distribution in any one dimension (assuming that the standard deviation is uniform in all the dimensions).

Proof. See Appendix C.

5.3.1 Grid (\mathcal{G}) Structure

In order to find the top- k distance-based outliers from uncertain dataset using the PC-list, mean of each object in \mathcal{GDB} is quantized to a d -dimensional grid

\mathcal{G} that is partitioned into cells of length l (The cell length is discussed in Sec. 4.3.5). Let $C_{\psi_1, \dots, \psi_d}$ be any cell in \mathcal{G} , where positive integers ψ_1, \dots, ψ_d denote the cell indices. The layers (L_1, \dots, L_n) of $C_{\psi_1, \dots, \psi_d} \in \mathcal{G}$ are the neighbouring cells of $C_{\psi_1, \dots, \psi_d}$, as shown in Fig. 5.1 and are derived as follows.

$$\begin{aligned} L_1(C_{\psi_1, \dots, \psi_d}) &= \{C_{x_1, \dots, x_d} | x_1 = \psi_1 \pm 1, \dots, x_d = \psi_d \pm 1, C_{x_1, \dots, x_d} \neq C_{\psi_1, \dots, \psi_d}\}. \\ L_2(C_{\psi_1, \dots, \psi_d}) &= \{C_{x_1, \dots, x_d} | x_1 = \psi_1 \pm 2, \\ &\quad \dots, x_d = \psi_d \pm 2, C_{x_1, \dots, x_d} \notin L_1(C_{\psi_1, \dots, \psi_d}), C_{x_1, \dots, x_d} \neq C_{\psi_1, \dots, \psi_d}\}. \end{aligned}$$

$L_3(C_{\psi_1, \dots, \psi_d}), \dots, L_n(C_{\psi_1, \dots, \psi_d})$ are derived in a similar way. We will use C to denote $C_{\psi_1, \dots, \psi_d}$ when there is no confusion.

Let $R_{D-t\sigma}(C)$ denotes a region formed by $\lfloor \frac{D-t\sigma}{l\sqrt{d}} - 1 \rfloor$ neighbouring layers of $C \in \mathcal{G}$ as shown in Fig. 5.2. The region $R_{D-t\sigma}(C)$ is chosen in such a way that for each $o_i \in C$ and $o_j \in R_{D-t\sigma}(C)$, $Pr(o_i, o_j, D) \approx 1$. Similarly $R_{D+t\sigma}(C)$ denotes a region formed by $\lceil \frac{D+t\sigma}{l} \rceil$ neighbouring layers of cell $C \in \mathcal{G}$ as shown in Fig. 5.1. Region $R_{D+t\sigma}(C)$ is chosen in such a way that for each $o_i \in C$ and $o_j \notin R_{D+t\sigma}(C)$, $Pr(o_i, o_j, D)$ approaches zero.

5.3.2 PC-list Structure

Populated-cells list (PC-list) is an array of non-empty cells of a d -dimensional grid. Let $N(C)$ be the number of objects in C , and $N_{D-t\sigma}(C)$ is the number of objects within cells in region $R_{D-t\sigma}(C)$ (including C itself). Then the PC-list (PC) is a sorted list containing $N(C)$ and $N_{D-t\sigma}(C)$ for each non-empty cell $C \in \mathcal{G}$ as shown in Fig. 5.2. The tuples in the PC-list are sorted in an ascending order of $N_{D-t\sigma}(C)$ column. The idea behind sorting is that outliers tend to exist in sparse regions. Sorting tuples in the PC-list, lets us identify cells with few number of neighbouring objects or cells in sparse regions.

The PC-list constructed in such a way that the majority of cells at the top of the PC-list contain candidate outlier objects. To prune the cells in the PC-list which cannot contain top- k outliers, cell bounds are computed. In practise, only small percentage of cells at the top of the PC-list require bounds computation. Rest of the cells are pruned as inlier cells i.e., the cells containing only inlier objects or the cells which do not contain the top- k outliers.

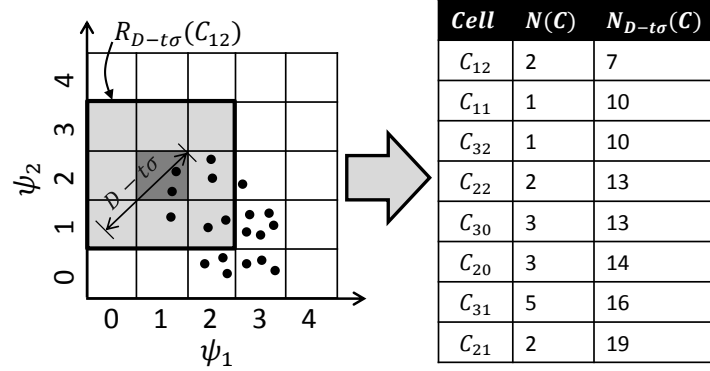


Figure 5.2: PC-list building

5.3.3 Cell Bounds

In order to identify cells $C \in PC$, containing only inliers or candidate top- k outliers, their bounds on the $\#D$ -neighbours are used. A cell C can be pruned as an inlier cell if the minimum $\#D$ -neighbours for any object in C is greater than threshold θ (θ is discussed shortly). Similarly a cell can be identified as containing top- k outliers (candidate outlier cell) if the maximum $\#D$ -neighbours for any object in C is less than θ . Since the Gaussian distribution is unbounded, $Pr(o_i, o_j, D)$ is always greater than zero for $o_i, o_j \in \mathcal{GDB}$. Therefore all the cells in the PC-list need to be considered for the computation of bounds of $C \in PC$. To compute cell bounds, the minimum and the maximum ordinary Euclidean distances between cells are required. Beside this, object count of each $C \in PC$ and $Pr(\alpha, D)$ values for α ranging from the minimum to the maximum ordinary Euclidean distances between cells in \mathcal{G} are also required. The $Pr(\alpha, D)$ values are precomputed and stored in a look-up table to be used by the top- k outlier detection algorithm.

Distance between Cells

Let C_p and C_q are two cells in PC with indices $\psi_{p1}, \dots, \psi_{pd}$ and $\psi_{q1}, \dots, \psi_{qd}$ respectively. Let $\Delta_{min}(C_p, C_q)$ and $\Delta_{max}(C_p, C_q)$ denote the minimum and the maximum ordinary Euclidean distances between C_p and C_q respectively. Distance between C_p and C_q depends on their positions in the grid \mathcal{G} and can be derived as follows.

$$\Delta_{min}(C_p, C_q) = l \times \left(\sum_{s=1}^d \delta_{min,s}^2 \right)^{1/2} \quad (5.3)$$

$$\text{where } \delta_{min,s} = \begin{cases} \psi_{ps} - (\psi_{qs} + 1) & \psi_{ps} > \psi_{qs} \\ (\psi_{ps} + 1) - \psi_{qs} & \psi_{ps} < \psi_{qs} \\ \psi_{ps} - \psi_{qs} & \psi_{ps} = \psi_{qs} \end{cases}$$

$$\Delta_{max}(C_p, C_q) = l \times \left(\sum_{s=1}^d \delta_{max,s}^2 \right)^{1/2} \quad (5.4)$$

$$\text{where } \delta_{max,s} = \begin{cases} (\psi_{ps} + 1) - \psi_{qs} & \psi_{ps} \geq \psi_{qs} \\ \psi_{ps} - (\psi_{qs} + 1) & \psi_{ps} < \psi_{qs} \end{cases}$$

Now the bounds for the PC-list cells can be obtained using pre-computed $Pr(\alpha, D)$ values and the information available in the PC-list. Let $LB(Pr(C_p, C_q))$ and $UB(Pr(C_p, C_q))$ denote $Pr(\alpha, D)$ values at minimum $\alpha \geq \Delta_{max}(C_p, C_q)$ and maximum $\alpha \leq \Delta_{min}(C_p, C_q)$, respectively. Then for a $C \in PC$, its lower bound $LB(C)$ and upper bound $UB(C)$ are defined as follows.

$$LB(C) = \sum_{C' \in PC} LB(Pr(C, C')) \times N(C'). \quad (5.5)$$

$$UB(C) = \sum_{C' \in PC} UB(Pr(C, C')) \times N(C'). \quad (5.6)$$

Since the major contribution in the bounds for $C \in PC$ is done by the cells in region $R_{D+t\sigma}(C)$, the bounds for $C \in PC$ can be redefined to reduce the number of pre computations and the bounds computation time, as follows.

$$LB(C) = \sum_{C' \in \{PC \cap R_{D+t\sigma}(C)\}} LB(Pr(C, C')) \times N(C'). \quad (5.7)$$

$$UB(C) = \sum_{C' \in \{PC \cap R_{D+t\sigma}(C)\}} UB(Pr(C, C')) \times N(C') + Pr(l\sqrt{d}(\lceil \frac{D+t\sigma}{l} \rceil + 1), D) \times (N - \sum_{C' \in \{PC \cap R_{D+t\sigma}(C)\}} N(C')). \quad (5.8)$$

Number of $Pr(\alpha, D)$ Pre-computations

Since the bounds of $C \in PC$ are computed using the cells in region $R_{D+t\sigma}(C)$, $Pr(\alpha, D)$ values need to be computed only for the neighbouring layers within $D + t\sigma$ distance of a cell. For $\lceil \frac{D+t\sigma}{l} \rceil$ neighbouring layers, we require $2\lceil \frac{D+t\sigma}{l} \rceil$ pre-computations. Two more pre-computations are required for the cell C itself and the objects that lie greater than $D + t\sigma$ distance of a cell. Hence the total number of pre-computations of $Pr(\alpha, D)$ required are only $2\lceil \frac{D+t\sigma}{l} \rceil + 2$.

5.3.4 Candidate Outlier Cells Detection

Using the bounds discussed in Sec. 5.3.3, a cell can be pruned as an inlier cell i.e., a cell containing only inlier objects or can be identified as containing top- k outlier candidates. Let \mathbb{C}_{cell} is a list for holding candidate outlier cells from PC-list, sorted in ascending order of $UB(C)$. Let $C^k \in \mathbb{C}_{cell}$ is a cell with the minimum upper bound containing the k^{th} object. A $C \in PC$ is a candidate outlier cell whenever $\sum_{C' \in \mathbb{C}_{cell}} N(C') < k$ or $LB(C) \leq \theta$, where $\theta = UB(C^k)$ denotes the threshold.

For a $C \in PC$, if $LB(C) > \theta$, C cannot contain any of the top- k outliers and can be pruned. On the other hand, if $LB(C) \leq \theta$, C may contain top- k outliers. C is added to \mathbb{C}_{cell} , such that \mathbb{C}_{cell} remain sorted of its $UB(C)$ attribute. Set $\theta = UB(C^k)$ and remove C' from \mathbb{C}_{cell} , such that $LB(C') > \theta$, as they cannot contain the top- k outliers.

Stopping Condition: The PC-list is scanned from top for candidate outlier cells. During the scanning, if a $C' \in PC$ is found such that $Pr(D - t\sigma, D) \times N_{D-t\sigma}(C') > \theta$, which is a lower bound on $\#D$ -neighbours of C' , C' can-

not contain top- k outliers and can be pruned. Since the PC-list is sorted of $N_{D-t\sigma}(C)$, any cell after C' must have $N_{D-t\sigma}(C) \geq N_{D-t\sigma}(C')$. Hence the lower bound of $C \in PC$ after C' must be greater than or equal to the lower bound of $C' \in PC$ and cannot contain top- k outliers. Hence the PC-list scanning can be stopped at this point.

Algorithm 5.2: k UDB Outlier Detection: PC-list Approach

Input: \mathcal{GDB}, D, l, k

Output: Top- k Distance-based Outliers

- 1: $N \leftarrow |\mathcal{GDB}|, \theta \leftarrow \infty;$
 - 2: $\mathbb{C}_{cell} \leftarrow \phi, \mathbb{C}_{obj} \leftarrow \phi;$ (Candidate outlier cells list and top- k candidate outlier objects list respectively)
 - 3: Map each $o \in \mathcal{GDB}$ to an appropriate cell C of grid \mathcal{G} ;
 - 4: Create PC-list PC , using non-empty cells of \mathcal{G} ;
 - 5: Sort PC w.r.t. $N_{D-t\sigma}(C)$ column;
/*Searching candidate outlier cells*/
 - 6: **for each** C in $|PC|$ **do**
 - 7: /*Stopping condition*/
 - 8: **if** $N_{D-t\sigma}(C) \times Pr(D - t\sigma, D) > \theta$ **then** Exit for loop.
 - 9: Compute $LB(C)$ and $UB(C)$;
 - 10: **if** $LB(C) \leq \theta$ **then**
 - 11: Add C to \mathbb{C}_{cell} (keep \mathbb{C}_{cell} sorted of $UB(C)$ attribute);
 - 12: **if** \mathbb{C}_{cell} contains $\geq k$ objects **then**
 - 13: Set $\theta = UB(C^k)$, such that C^k contain the k^{th} object;
 - 14: Remove all C from \mathbb{C}_{cell} , such that $LB(C) > \theta$;
 - 15: **end if**
 - 16: **end if**
 - 17: **end for**
/*Calculating $\#D$ -neighbors(o) of candidate top- k outliers*/
 - 18: The computation of $\#D$ -neighbors(o) is similar to that of the Naive approach. The only difference between the Naive algorithm and in this algorithm is that in this algorithm, $\#D$ -neighbors(o) are computed for the candidate objects in \mathbb{C}_{cell} only, however in the Naive case, all the objects in the dataset are considered for the computation of $\#D$ -neighbors.
-

5.3.5 The k UDB(CG) and the k UDB(Approx) Algorithms

In this section, we present two algorithms to detect top- k distance-based outliers from uncertain datasets. The first algorithm (k UDB(CG)) computes accurate $\#D$ -neighbours for all the un-pruned objects, however the second algo-

rithm (k UDB(Approx)) approximates the $\#D$ -neighbours to reduce the algorithm computation cost. In Sec. 5.4, we will present another approximate top- k algorithm using the bounded Gaussian uncertainty (k UDB(BG)).

The k UDB(CG) Algorithm

The Algorithm 5.2 first maps dataset objects to appropriate grid cells and creates the PC-list in lines 4 and 5 respectively. Since the PC-list is sorted in the ascending order of its $N_{D-t\sigma}(C)$ column, it guarantees that cells in the sparse regions of the grid \mathcal{G} are at the top of the PC-list. Hence the candidate outlier cells are expected to be at the top of the list. We scan the PC-list and add the candidate outlier cells in \mathbb{C}_{cell} until the stopping condition on line 8 becomes true. The number of objects in \mathbb{C}_{cell} may be greater than k , hence their $\#D$ -neighbours are computed to find the top- k outliers and their ranking. The object is then added to the \mathbb{C}_{obj} (set of candidate outlier objects) along with its $\#D$ -neighbors(o). The objects in \mathbb{C}_{obj} are sorted in ascending order of $\#D$ -neighbors(o) column. As the k^{th} object's $\#D$ -neighbors(o) is found, threshold θ is set (refer line 13 of Algorithm 5.2). During the calculation of $\#D$ -neighbors(o), if for some o' , $\#D$ -neighbors(o') becomes greater than θ , then o' can not be among the top- k outliers and is removed from further consideration.

The k UDB(Approx) Algorithm

In the k UDB(CG) algorithm, the minimum number of distance probability computations required for the evaluation of k $\#D$ -neighbors(o) is kN , however the candidate outlier objects which require the evaluation of $\#D$ -neighbors(o), may be greater than k . When the distance probability is expensive to compute (as in our case), computation of even k $\#D$ -neighbors(o) is very expensive. According to our distance probability function, the major contribution in the evaluation of $\#D$ -neighbors(o) is done by the nearer objects. Hence $\#D$ -neighbors(o) for each un-pruned o can be approximated with high accuracy by considering objects only within $D + t\sigma$ distance of o according to Lemma 5.1, rather than considering all the objects in dataset. It saves a lot of computation time. Rest of the algorithm is same as that of the accurate top- k algorithm.

Maximum Approximation Error: For any $o \in \mathcal{GDB}$, maximum approximation error (ε_{max}) happens if all the $o' \in \mathcal{GDB} \setminus o$ are at a distance slightly greater than $D + t\sigma$ from o . Hence $\varepsilon_{max} = (N - 1) \times Pr(D + t\sigma + \beta, D)$, where $\beta \in \mathbb{R}$ is a very small real value to make distance greater than $D + t\sigma$. For example for $t = 9$, $d = 2$ and $N = 10^5$ objects, $\varepsilon_{max} \approx 10^{-5}$. ε_{max} depends mainly on t . In practice $t \geq 3$ gives sufficiently accurate $\#D$ -neighbors(o) for $d = 2$ and 3. For higher d values, we need to increase t value according to Lemma 5.1.

5.3.6 Complexity Analysis

We will first analyse the complexity of the *top-k* algorithm for the 2D case. Lines 1 and 2 contain only the initializations of variables. Since there are N objects in the dataset \mathcal{GDB} , line 3 takes $O(N)$ time. Line 4 takes $O(m)$ time, where $m \ll N$ is the total number of populated cells in the cell-grid. Sorting m cells in the PC-list in line 5 takes $O(m \log(m))$ time. The main loop of the algorithm in lines 6-17 is executed for all the cells in the PC-list in the worst case. The loop computes the lower and upper cell bounds, each of which takes $O(m)$ time because the cell bounds computation require the contribution of all the cells in the PC-list. Keeping the \mathbb{C}_{cell} sorted in Line 11 takes $O(m)$ time in the worst case. Lines 13 and 14 within the loop takes at-most $O(m)$ each. Hence the overall loop takes $O(m^2)$ time. Finally, computation of the $\#D$ -neighbours in Line 18 takes $O(nN)$ time, where $n \ll N$ is the number of candidate objects for the *top-k* outliers. Thus, the average case time complexity of the k UDB outlier detection algorithm in 2D is $O(nN + m^2)$. In the worst case, none of the objects is pruned by the cell and PC-list based pruning, hence the worst case time complexity of the k UDB(CG) outlier detection algorithm in 2D is $O(N^2 + m^2)$.

However, in the k UDB(CG) algorithm, the major cost lies in the evaluation of $\#D$ -neighbours of the candidate outlier objects (Line 18). This cost is so high that it hides the cost of the rest of the algorithm. This is due to the expensive distance probability computation between uncertain objects. Therefore, we give the k UDB(CG) algorithm complexity in terms of the number of distance probability evaluations. Hence the average case and the worst case time complexities of the k UDB(CG) outlier detection algorithm in 2D are $O(nN)$ and

$O(N^2)$, respectively.

Since the k UDB(Approx) approach considers objects only within $D+t\sigma$ distance of the candidate outlier objects for the computation of their $\#D$ -neighbours, the average case and the worst case time complexities of the k UDB(Approx) outlier detection algorithm in 2D are $O(nN'')$ and $O(N^2)$, respectively; where $N'' \leq N$ is the number of objects that lie within $D + t\sigma$ distance of the candidate objects. In the k UDB(Approx) algorithm, the worst case time complexity assumes that none of the objects is pruned by the cell and PC-list based pruning and all the dataset objects lie within $D + t\sigma$ distance of the candidate outlier objects.

The complexities of the proposed algorithms do not change with the increase in dimensions d , as long as only the number of the distance probability computations are considered for the computation of algorithms' complexities. Although with the increase in d , the number of grid cells increases exponentially, yet the cost of the evaluation of the $\#D$ -neighbours for the candidate outlier objects remains dominant and hence the complexities remain same as discussed above for higher dimensional case.

From the above analysis, it is evident that the computational complexity of the proposed algorithms is lower than the Naive algorithm, which is $O(N^2)$ in terms of the distance probability computations. Hence the execution times of the proposed algorithms is far lower than the Naive algorithm. However, with the increase in d , the distance probability computation between uncertain objects becomes very expensive and it becomes impractical to detect outliers using the proposed approach from very high dimensional data.

5.4 PC-list-based Outlier Detection using the Bounded Gaussian Uncertainty

Approximating the Gaussian uncertainty by the bounded Gaussian uncertainty enables an approximate but more efficient outlier detection. According to this paper's assumption, attributes of uncertain objects follow the Gaussian distribution. Therefore according to the 3-sigma rule there is a 95.45% chance that uncertain objects' attribute values lie within 2 standard deviations of the observed values and 99.73% chance that the values lie within 3 standard deviations of the

observed values [88]. Hence the conventional Gaussian distribution can be normalized within certain boundaries to increase the efficiency of the top- k outlier detection at a small cost of accuracy.

Given a two dimensional conventional Gaussian function $g_{\vec{\lambda}}(x_1, x_2)$ with mean $\vec{\mu} = (\mu_1, \mu_2)$ and co-variance matrix $\Sigma = \text{diag}(\sigma^2, \sigma^2)$, the bounded Gaussian distribution $f_{\vec{\lambda}}(x_1, x_2)$ can be defined following the practise of Tao et al. [108], as follows.

$$f_{\vec{\lambda}}(x_1, x_2) = \begin{cases} \frac{g_{\vec{\lambda}}(x_1, x_2)}{\int_{(x_1, x_2) \in o.ur} g_{\vec{\lambda}}(x_1, x_2) dx_1 dx_2} & (x_1, x_2) \in o.ur \\ 0 & otherwise \end{cases} \quad (5.9)$$

where *o.ur* denotes the uncertainty region of the bounded Gaussian distribution. This work assumes that the uncertainty region is a sphere with centre (μ_1, μ_2) and radius $r = t\sigma$ (t is discussed in Lemma 5.1).

By bounding the Gaussian uncertainty, a cell can be pruned by simply counting the number of objects in its neighbouring cells. Moreover the major cost of outlier detection, that is, the processing of un-pruned objects also reduces significantly. This is because, with the bounded Gaussian uncertainty, the outlier detection algorithm needs to consider limited number of objects for the computation of an object's $\#D$ -neighbors rather than all the objects in the dataset. For details on the bounded Gaussian uncertainty, please refer Sec. 4.4.

5.4.1 Grid (\mathcal{G}) and PC-list Structures for the Bounded Gaussian

In order to identify distance-based outliers using the PC-list, mean of each object in \mathcal{GDB} is mapped to a d -dimensional space that is partitioned into cells of length l (l is discussed in Sec. 4.3.5). Let $C_{\psi_1, \dots, \psi_d}$ be a cell in the Grid \mathcal{G} , then cells in region $R_{D-2r}(C_{\psi_1, \dots, \psi_d})$ are those which completely lie within $D - 2r$ distance of the $C_{\psi_1, \dots, \psi_d}$, including the $C_{\psi_1, \dots, \psi_d}$ itself. Let $n_{D-2r} = \lfloor \frac{D-2r}{l} \rfloor - 1$, then the region $R_{D-2r}(C_{\psi_1, \dots, \psi_d})$ is derived as follows.

$$R_{D-2r}(C_{\psi_1, \dots, \psi_d}) = \{C_{x_1, \dots, x_d} | x_1 = \psi_1 \pm n_{D-2r}, \dots, x_d = \psi_d \pm n_{D-2r},$$

$$\sqrt{\sum_{i=1}^d ((x_i + 1)l)^2} < D - 2r, C_{x_1, \dots, x_d} \neq C_{\psi_1, \dots, \psi_d} \}. \quad (5.10)$$

The number of cells in the region $R_{D-2r}(C_{\psi_1, \dots, \psi_d})$ vary depending upon n_{D-2r} . Note that the region $R_{D-2r}(C_{\psi_1, \dots, \psi_d})$ satisfies the following property.

Property 1: If $C_{x_1, \dots, x_d} \in R_{D-2r}(C_{\psi_1, \dots, \psi_d})$, then the objects $o_i \in C_{\psi_1, \dots, \psi_d}$ and $o_j \in C_{x_1, \dots, x_d}$ are at most $D - 2r$ distance apart.

From property 1, the $o_i \in C_{\psi_1, \dots, \psi_d}$ and the $o_j \in R_{D-2r}(C_{\psi_1, \dots, \psi_d})$ are guaranteed to be D -neighbours mutually, hence the $Pr(o_i, o_j, D)$ is always equal to 1. Cells in region $R_{D+2r}(C_{\psi_1, \dots, \psi_d})$ are those which fall within $D + 2r$ distance of the $C_{\psi_1, \dots, \psi_d}$. Let $n_{D+2r} = \lceil \frac{D+2r}{l} \rceil$, then the region $R_{D+2r}(C_{\psi_1, \dots, \psi_d})$ is derived as follows.

$$R_{D+2r}(C_{\psi_1, \dots, \psi_d}) = \{C_{x_1, \dots, x_d} | x_1 = \psi_1 \pm n_{D+2r}, \dots, x_d = \psi_d \pm n_{D+2r},$$

$$\sqrt{\sum_{i=1}^d ((x_i - 1)l)^2} < D + 2r, C_{x_1, \dots, x_d} \notin R_{D-2r}(C_{\psi_1, \dots, \psi_d}), C_{x_1, \dots, x_d} \neq C_{\psi_1, \dots, \psi_d} \}. \quad (5.11)$$

Note that the region $R_{D-2r}(C_{\psi_1, \dots, \psi_d})$ and the region $R_{D+2r}(C_{\psi_1, \dots, \psi_d})$ satisfy the following property.

Property 2: If C_{x_1, \dots, x_d} is neither in $R_{D-2r}(C_{\psi_1, \dots, \psi_d})$ nor in $R_{D+2r}(C_{\psi_1, \dots, \psi_d})$ and $C_{x_1, \dots, x_d} \neq C_{\psi_1, \dots, \psi_d}$, then the objects $o_i \in C_{\psi_1, \dots, \psi_d}$ and $o_j \in C_{x_1, \dots, x_d}$ are greater than $D + 2r$ distance apart.

From property 2, it can be guaranteed that the $o_i \in C_{\psi_1, \dots, \psi_d}$ and $o_j \in C_{x_1, \dots, x_d}$ are greater than $D + 2r$ distance apart, hence the $Pr(o_i, o_j, D)$ is always equal to 0. In the following, C is used to denote $C_{\psi_1, \dots, \psi_d}$ when there is no confusion.

The PC-list structure for the bounded Gaussian is similar to that of the conventional Gaussian (refer Sec. 5.3.2) except the column $N_{D-t\sigma}(C)$. Instead of $N(C)$ and $N_{D-t\sigma}(C)$, the PC-list contains $N(C)$ and $N_{D-2r}(C)$ for each non-empty cell of \mathcal{G} . The tuples in the PC-list are sorted in an ascending order of $N_{D-2r}(C)$ column.

5.4.2 Cell Bounds for the Bounded Gaussian

In order to identify cells $C \in PC$, containing only inliers or candidate top- k outliers, their bounds on the $\#D$ -neighbours are used. A cell C can be pruned as an inlier cell if the minimum $\#D$ -neighbours for any object in C is greater than threshold θ (θ is discussed in Sec. 5.3.4). Similarly a cell can be identified as containing top- k outliers if the maximum $\#D$ -neighbours for any object in C is less than θ . In case of the bounded Gaussian distribution, $Pr(o_i, o_j, D) = 0$ if the means of o_i and o_j are greater than $D + 2r$ distance. Hence only cells within regions R_{D-2r} and R_{D+2r} of a $C \in PC$ need to be considered for the computation of its lower and upper bounds respectively.

Thus for a $C \in PC$, its lower bound $LB(C)$ and upper bound $UB(C)$ are defined as follows.

$$LB(C) = \sum_{C' \in \{PC \cap R_{D-2r}(C)\}} N(C') \quad (5.12)$$

$$UB(C) = \sum_{C' \in \{PC \cap R_{D+2r}(C)\}} N(C') \quad (5.13)$$

5.4.3 Candidate Outlier Cells Detection for the Bounded Gaussian

For the bounded Gaussian case, the procedure of candidate outlier cell detection is similar to that discussed in Sec. 5.3.4. However the stopping condition is slightly different. During the scanning of PC-list, if a $C' \in PC$ is found such that $N_{D-2r}(C') > \theta$, which is a lower bound on $\#D$ -neighbors of C' , C' cannot contain top- k outliers and can be pruned. Since the PC-list is sorted of $N_{D-2r}(C)$, any cell after C' must have $N_{D-2r}(C) \geq N_{D-2r}(C')$. Hence the PC-list scanning can be stopped safely at this position.

5.4.4 The k UDB(BG) Algorithm

Major part of the k UDB(BG) algorithm is same as that of the k UDB(CG) algorithm (Algorithm 5.2). The main difference lies in the construction of the PC-list and the computation of bounds as discussed in Secs. 5.4.1 and 5.4.2, respectively. Moreover evaluation of the candidate outlier objects now only require objects within $D+2r$ distance of the target object rather than all the objects in the dataset, bringing down the overall cost of execution.

5.4.5 Complexity Analysis

As discussed in Sec. 5.4.4, the major part of the k UDB(BG) algorithm is same as that of the k UDB(CG) algorithm, the computational complexity of the k UDB(BG) algorithm is also similar to that of the k UDB(CG) algorithm. Just like the k UDB(CG) algorithm, the major cost of the top- k outlier detection lies in the evaluation of the $\#D$ -neighbours of the candidate outlier objects. This cost is so high that it hides the cost of the rest of the algorithm. This is due to the expensive distance probability computation between the uncertain objects. Therefore we give the complexities of the k UDB(BG) algorithm in terms of the number of distance probability evaluations, which are $O(nN')$ and $O(N^2)$ in 2-dimensional case, for the average case and the worst case respectively; where $n \ll N$ is the number of candidate outlier objects and $N' \leq N$ is the number of objects that lie within $D+2r$ distance of the candidate outlier objects. Although the number of distance probability evaluations required for the processing of unpruned objects in the k UDB(BG) algorithm is far less than the k UDB(CG), its worst case complexity is still $O(N^2)$ in 2D case, since it assumes that none of the objects are pruned through the cell and PC-list based pruning and all the dataset objects lie within $D+2r$ distance of the candidate outlier objects.

The complexities of the k UDB(BG) algorithm do not change with the increase in dimensions d , as long as only the number of distance probability evaluations are considered for the computation of the algorithm's complexity. Although with the increase in d , the number of grid cells increases exponentially, yet the cost of evaluation of the $\#D$ -neighbours for the candidate outlier objects remains dominant and hence the complexities remain same for the average case and the worst case, i.e., $O(nN')$ and $O(N^2)$, respectively for higher dimensional case.

From the above analysis, it is evident that the computational complexities of the proposed algorithms is lower than the Naive algorithm, which is $O(N^2)$ in terms of the distance probability evaluation. Hence the execution time of the k UDB(BG) algorithm is far lower than the Naive algorithm. However, with the increase in d , the distance probability computation between uncertain objects becomes very expensive and it becomes impractical to detect outliers using the proposed approach from very high dimensional data.

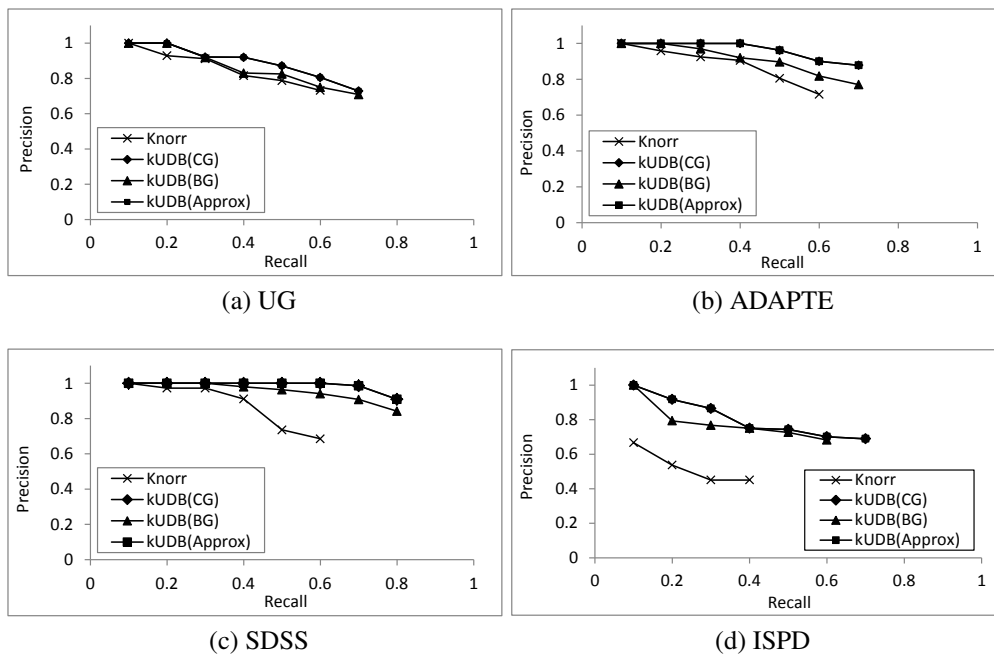


Figure 5.3: Precision-recall trade-off curves ($D = 70$, $\sigma = 10$, $\sigma_p = 30$, $l = 10$, $t = 3$ and $k = 50$)

5.5 Experiments

Extensive experiments are conducted on synthetic and real datasets to evaluate the effectiveness and efficiency of the proposed approaches. In the experiments, Knorr et al. [63] approach of outlier detection on deterministic data is used as the baseline to compare the accuracy of outlier detection. All algorithms are implemented in C++, GNU compiler. All experiments are performed on a system with an Intel Core 2 Duo CPU E8400 3.00GHz CPU and 2GB main memory running Ubuntu 12.04 OS. All programs run in main memory and no I/O cost

is considered. Each experiment is performed 3 times and the average values are used in the graphs. We have also used error bars in the graphs, showing the standard error in the execution time measurements of each approach. However, in majority of the graphs they are not visible due very small standard error in the execution times.

Pre-computation time is not included in the measurements. Unless specified, the following parameter values are used in experiments: $D = 100$, $\sigma = 10$, $l = 10$, $t = 2$, $r = t\sigma$ and $k = 15$. In figures, the Knorr et al. [65] approach is denoted by *Knorr* and the proposed approaches, i.e., the top- k , the top- k approximate and the top- k approximate using the bounded Gaussian uncertainty are denoted by k UDB(CG), k UDB(Approx) and k UDB(BG) respectively. The datasets used in the experiments are discussed in Sec. 4.6.1 of chapter 4.

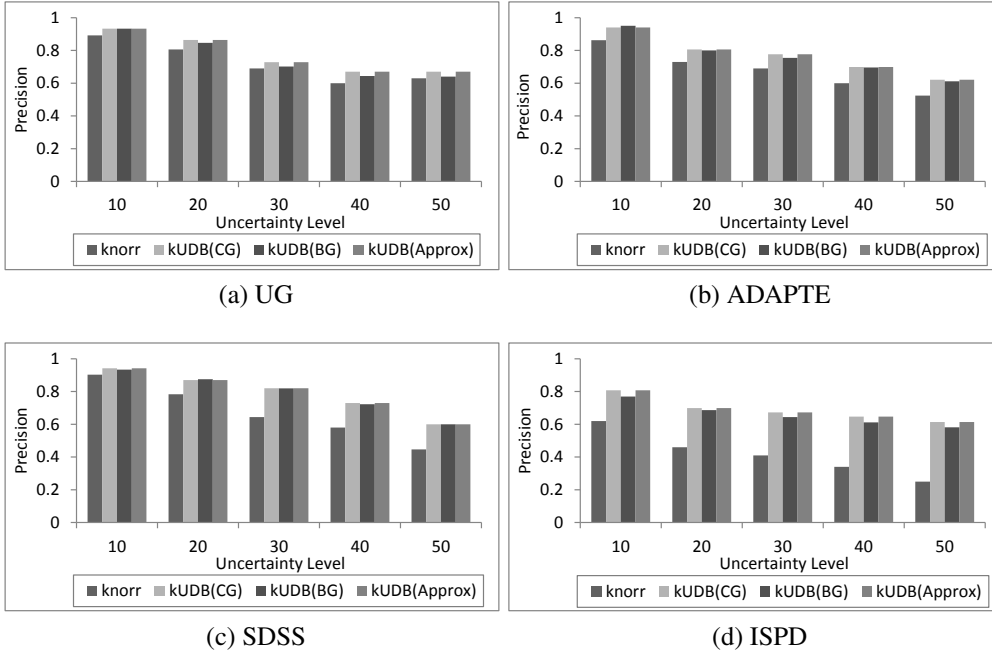


Figure 5.4: Precision with increasing σ_p ($D = 70$, $\sigma = 10$, $l = 10$, $t = 3$ and $k = 50$)

5.5.1 Accuracy

Firstly, experiments are performed to evaluate the accuracy of the proposed approaches. Since there are no known approaches for the top- k distance-based

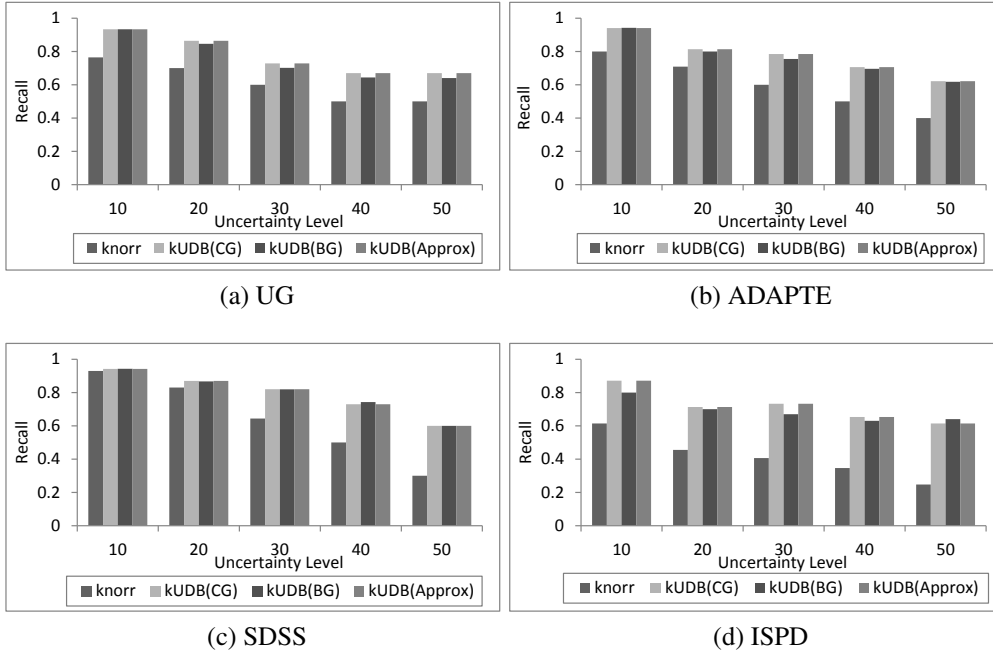
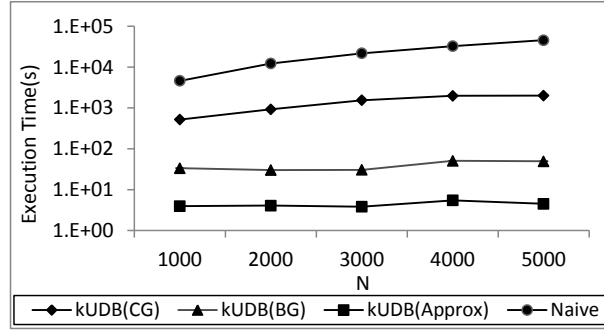


Figure 5.5: Recall with increasing σ_p ($D = 70$, $\sigma = 10$, $l = 10$, $t = 3$ and $k = 50$)

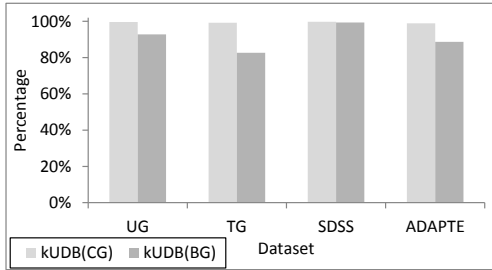
outlier detection on uncertain data, the deterministic approach for the distance-based outlier detection given by Knorr et al. [65] is used as a baseline. Slight changes are made in the Knorr's algorithm to obtain top- k outliers from it. Since the outliers are not known, for both the synthetic and the real datasets, the baseline algorithm is used to determine the outliers on the original datasets. The results obtained from the baseline approach are used as the ground truth. In order to judge the accuracy of the proposed approaches, the precision and recall are measured on the perturbed dataset for the baseline approach and the proposed approaches. The precision is defined as the ability of the algorithm to present only true outliers. The recall is defined as the ability of the algorithm to present all true outliers. The perturbed dataset is obtained by adding normal random numbers with zero mean and standard deviation σ_p to each of the tuple values of the original dataset. The σ_p was varied from 10 to 50 (with a step of 10) to generate perturbed datasets of five different levels. Experiments show that the proposed approaches are superior than the baseline approach, since they do not degrade quite as much with increasing uncertainty. Unless specified, the following parameter values are used for the experiments in this subsection: $D = 70$,

$\sigma = 10, \sigma_p = 30, l = 10, t = 3, r = t\sigma$ and $k = 50$.

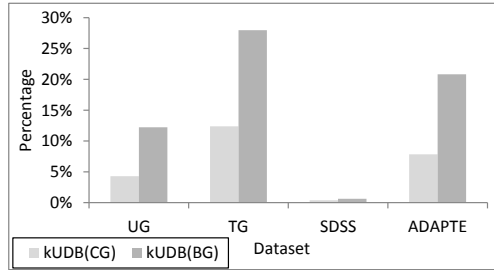
Firstly the precision-recall trade-off curves are presented for different datasets. In all the graphs in Fig. 5.3, both the precisions and recalls of the proposed approaches are higher than the baseline approach. Moreover the precision-recall curves of the k UDB(CG) and the k UDB(Approx) are exactly same. This is due to the fact that both the approaches returned same outliers. Although there was a slight difference in the $\#D$ -neighbours of the outliers returned by both the proposed approaches, but this difference was not big enough to change the top- k outlier objects or their ranking. The precision-recall of the k UDB(BG) approach is also better than the baseline approach and in most of the datasets is equal to the k UDB(CG) approach.



(a) Naive vs. PC-list



(b) Pruning Percentage



(c) Stopping Condition

Figure 5.6: Effectiveness of the PC-list based approach ($D = 100, l = 10, \sigma = 10, t = 3, r = t\sigma$, and $k = 15$)

In Fig. 5.3a, the precision-recall curves are almost same for all the approaches, however in Figs. 5.3b, 5.3c and 5.3d the precision-recall curves of the proposed approaches are comparatively higher than the baseline approach. Specially the low recall in Figs. 5.3c and 5.3d shows the presence of a large number of false positive outliers in the outliers obtained from the baseline approach.

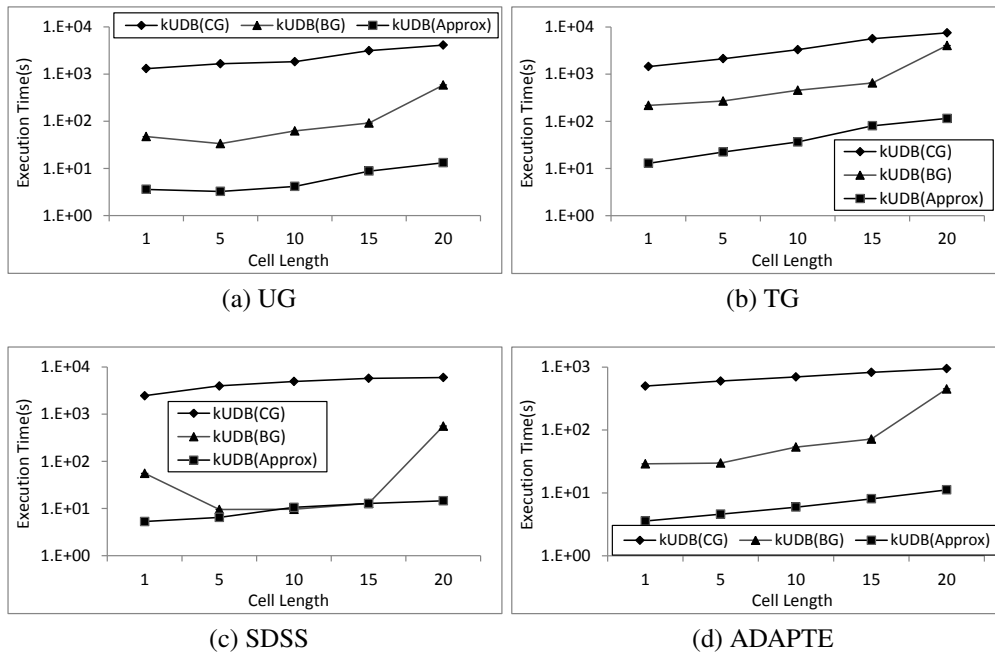


Figure 5.7: Varying parameter l ($D = 100$, $\sigma = 10$, $t = 3$, $r = t\sigma$, and $k = 15$)

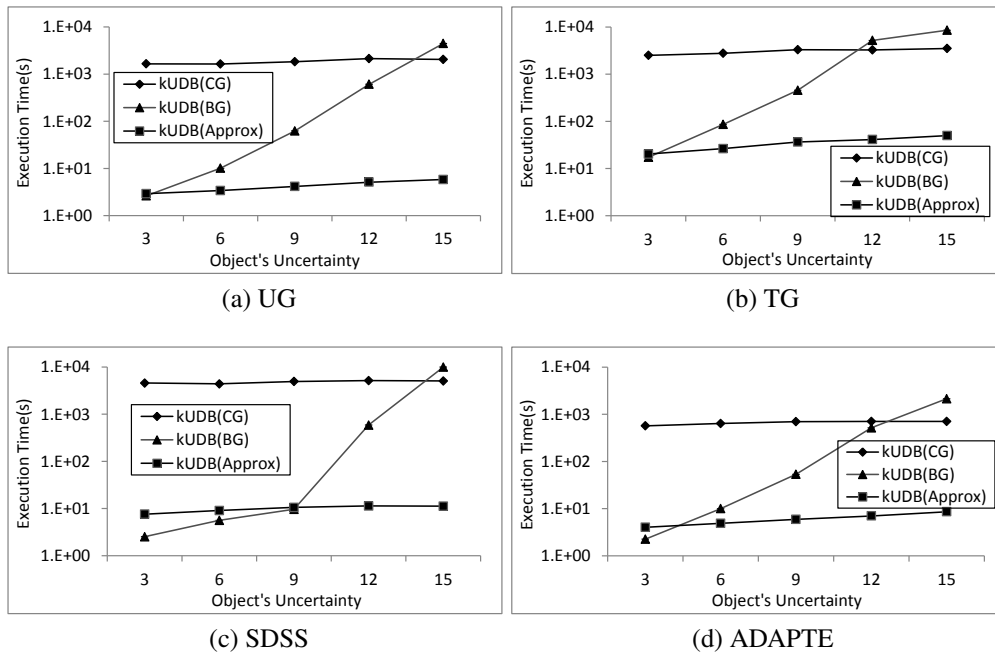


Figure 5.8: Varying object's uncertainty σ ($D = 100$, $l = 10$, $t = 3$, $r = t\sigma$, and $k = 15$)

The accuracy of the proposed approaches is also evaluated with the increasing level of uncertainty. From Fig. 5.4, it is clear that the precision falls with increasing uncertainty level. Moreover, the precision of the proposed approaches is always higher than the baseline approach in Fig. 5.4, which means that fewer false-positive outliers were returned by the proposed approaches than the baseline approach. Similar results are illustrated for recall in Fig. 5.5. In all four plots of Fig. 5.5, the recall is somewhat consistent with increasing uncertainty level for the proposed approaches. Which proves that the proposed approaches are better than the baseline approach in retrieving only true outliers, even from the noisy data.

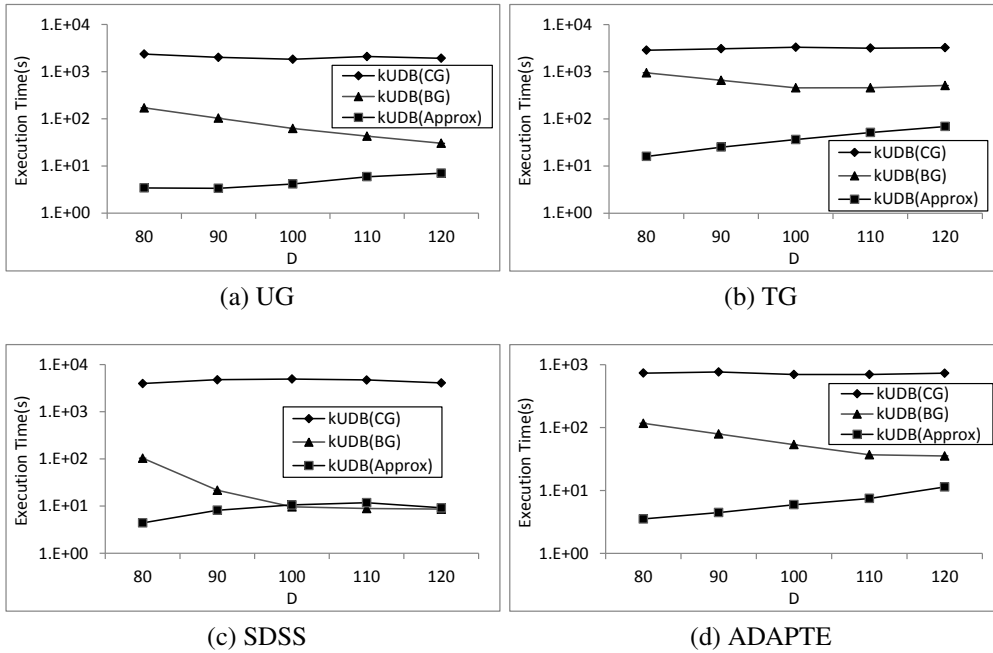


Figure 5.9: Varying parameter D ($l = 10$, $\sigma = 10$, $t = 3$, $r = t\sigma$, and $k = 15$)

5.5.2 Efficiency

In this subsection, experiments are conducted to evaluate the efficiency of the proposed top- k outlier detection approaches presented in Secs. 5.3 and 5.4. Fig. 5.6a compares the execution times of the Naive and the proposed approaches on the UG dataset. Please note the use of logarithmic scale in all the efficiency graphs to keep the graph lines visible. The proposed approaches are several

times faster than its Naive counterpart due to their strong pruning capability as can be observed from Fig.5.6b. Stopping condition discussed in Secs. 5.3.4 and 5.4.3 helps identify candidate outlier cells very quickly. Fig. 5.6c shows the percentage of cells considered in the PC-list to identify candidate outlier cells. The percentage is comparatively higher for trimodal Gaussian dataset because the dataset is relatively sparse and hence results in larger number of candidate outlier cells. Moreover, the k UDB(Approx) and the k UDB(BG) approaches are several times faster than the k UDB(CG) approach. This is due to the fact that these approaches, in contrast to the k UDB(CG) approach, do not consider all the dataset objects for the computation of $\#D$ -neighbors of the candidate objects. From theoretical analysis in Sec. 5.3.5 and experiments, we found that the k UDB(Approx) approach gives an accuracy of up to several decimal digits in the evaluation of $\#D$ -neighbors(o) and hence the outliers obtained from the k UDB(CG) and the k UDB(Approx) are same. In addition the execution time of the k UDB(Approx) approach in all the experiments is several times lower than its exact counterpart. On the other hand, the accuracy of the k UDB(BG) approach is not as high as that of the k UDB(Approx) approach, however it does not require any pre-computation.

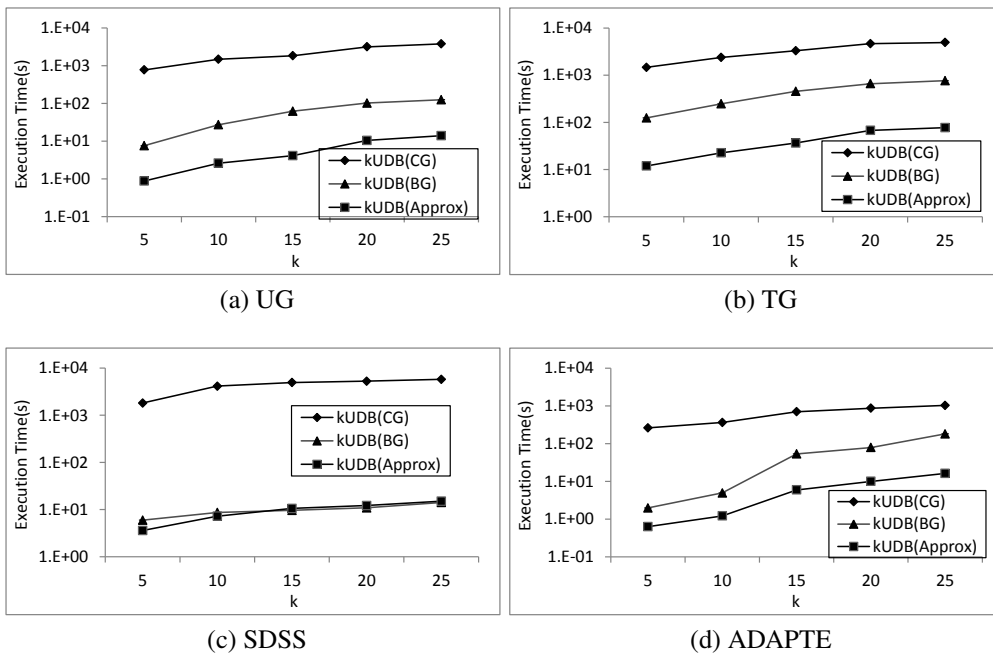


Figure 5.10: Varying parameter k ($D = 100$, $\sigma = 10$, $t = 3$, $r = t\sigma$, and $l = 15$)

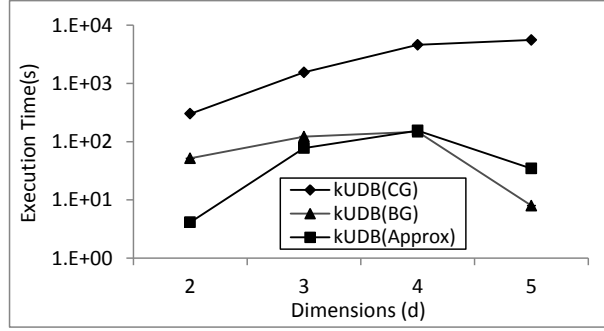


Figure 5.11: Vary dimensions d ($N = 500$, $D = 100$, $l = 10$, $\sigma = 10$, $t = 3$, $r = t\sigma$, and $k = 15$)

Graphs in Figs. 5.7, 5.8, 5.9 and 5.10 show the affect of varying different parameters on the execution times. Firstly, consider the variation of parameter l in Fig. 5.7. The smaller cell lengths l require the lower execution times and vice versa, which can be observed from Fig. 4.10. Smaller l values are good for cell pruning as they result in tighter bounds, however very small l increases the number of cells in the grid exponentially and the time required for the bounds computation. On the other hand, larger l values result in looser bounds and hence reduce the cell pruning capability. Moreover, as the number of dimensions d increases, the number of grid cells increases exponentially. Therefore, smaller l values are recommended for lower dimensions and relatively larger values are recommended for higher dimensions.

Next, experiments are performed by varying the dataset objects' uncertainty which is denoted by σ . As σ increases, the uncertainty of objects also increases. This increase in uncertainty results in smaller $Pr(o_i, o_j, D)$ values even if o_i and o_j are located nearby. Hence the number of distance probability evaluations required increases for un-pruned objects, which results in higher execution times as can be observed from graphs in Fig. 5.8. Moreover it can be observed from Fig. 5.8 that for smaller σ , the computation cost is lowest for the k UDB(BG) approach. Please recall that the bounded Gaussian uncertainty is bounded by radius $r = t\sigma$. Hence smaller σ results in smaller r and it helps in early pruning of objects, brining down the overall cost of the algorithm. However large σ results in larger r and in an increase in the number of objects required for the computation of $\#D$ -neighbors. Which is the cause of higher execution times for the k UDB(BG) for higher σ values.

Graphs in Fig. 5.9 show the affect of varying parameter D . For each un-

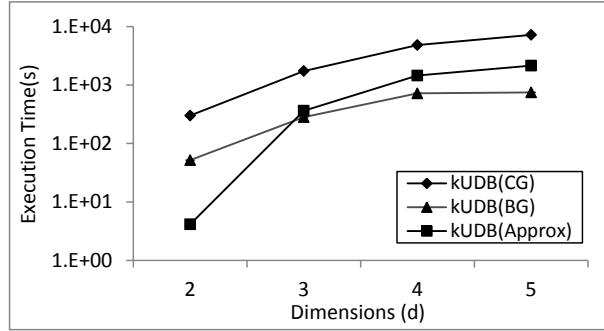


Figure 5.12: Vary dimensions d and parameter D ($N = 500$, $l = 10$, $\sigma = 10$, $t = 3$, $r = t\sigma$, and $k = 15$)

pruned o from the PC-list-based pruning, increase in D results in an increase in the $\#D$ -neighbours which need to be considered for the approximation of $\#D$ -neighbors(o). Therefore it increases the execution time of the $kUDB(Approx)$ approach. However, in case of the $kUDB(CG)$ and $kUDB(BG)$ approaches, algorithm execution times decreases with the increase in D . This is due to the fact that for larger D , $Pr(o_i, o_j, D)$ is higher. Hence objects are more easily pruned if it is an inlier, reducing the overall cost of the algorithms.

From graphs in Fig. 5.10, increase in k results in an increase in execution times of the algorithms, which is quite obvious behaviour of the algorithms.

Finally, experiments are performed by varying the number of dimensions. Experiments in Fig. 5.11 are performed on the synthetic dataset UG with $N = 500$. Computation cost of all the algorithms increases with the increase in dimensions as can be observed from Fig. 5.11, however the computation cost of the $kUDB(BG)$ and the $kUDB(Approx)$ decreases for $d = 5$. This is due to the fact that as d increases, objects get sparse and very few objects lie within the $D + 2r$ and the $D + t\sigma$ distance of the candidate objects in the $kUDB(BG)$ and the $kUDB(Approx)$ approaches, respectively. Hence, $\#D$ -neighbors computation of the candidate objects in the $kUDB(BG)$ and the $kUDB(Approx)$ approaches require relatively less time than the $kUDB(CG)$.

We also performed experiments by varying the number of dimensions d and the parameter D in parallel. In the experiments shown in Fig. 5.11, parameter D was kept constant, due to which the number of outliers increases dramatically with the increase in d . With the increase in d , objects get sparse and the parameter D must be increased accordingly to find the limited number of outliers. Hence in Fig. 5.12, we increased the parameter D with the increase in d , in

proportion to the average distance between the dataset objects. By doing so, we obtained almost same percentage of outliers from all the dimensions. Moreover, the execution times of all the approaches are far low in Fig. 5.11 as compared to the Fig. 5.11, specially for dimensions 3 and higher. Computation cost of all the algorithms increases with the increase in dimensions due to the exponential increase in the number of cells in the grid. In addition, the distance probability becomes expensive with the increase in d , which is another cause of the increase in algorithms' execution times.

5.6 Summary

In this chapter, an exact (k UDB(CG)) and two approximate (k UDB(Approx) and k UDB(BG)) approaches on top- k distance-based outlier detection from uncertain datasets are proposed. All the approaches make use of a cell grid and a PC-list (populated-cells list) to quickly identify the candidate outlier objects. The only difference between the k UDB(CG) and the k UDB(Approx) approaches is the computation of $\#D$ -neighbours. The k UDB(CG) approach computes the $\#D$ -neighbours of the top- k candidate objects by considering all the objects in the dataset, however the k UDB(Approx) approach considers only nearer objects for its $\#D$ -neighbour computation to reduce its computation cost. The k UDB(BG) approach makes use of the bounded Gaussian uncertainty to reduce the computation cost of outlier detection. Experiments prove that the accuracy of all the proposed algorithms (k UDB(CG), k UDB(Approx) and k UDB(BG)) is far higher than the baseline algorithm. The PC-list based approach is effective in reducing the number of objects need to be considered for the top- k outlier detection, due to which the proposed approaches are several times faster than the Naive approach. In addition, the approximate approaches save a lot more computation time at a small loss of accuracy.

Chapter 6

Continuous Outlier Detection on Uncertain Data Streams

The main goal of the research in this chapter is to obtain distance-based outliers from uncertain time series data streams. In data streams, data arrive continuously at high rate. Such streams are common due to the incremental usage of automated data collection devices (e.g., WSNs, monitoring cameras, etc.), which generate streams of uncertain data. The uncertainty in data from such devices is unavoidable and its causes are discussed in chapter 2. Hence an approach is required which can detect outliers continuously and at high speed. Hence, in this chapter we present an incremental outlier detection approach on uncertain data.

6.1 Overview

Outlier detection on uncertain static data is a challenging research problem in data mining. Moreover, the continuous and high speed arrival of data makes it more challenging. The problem of outlier detection from uncertain data streams is an important research problem in data mining but there exists very few works related to this problem as discussed in Sec. 3.2. None of the proposed works so far deals with the attribute-level uncertainty, which is the focus of this dissertation.

In this chapter, we propose a continuous outlier detection approach for uncertain time series data streams. Namely, a distance-based approach is proposed

to detect outliers continuously from a set of uncertain objects' states that are originated synchronously from a group of data sources (e.g., sensors in WSN). A set of objects' states at a timestamp is called a state set. Usually, the duration between two consecutive timestamps is very short and the state of all the objects may not change much in this duration. Therefore, to eliminate the unnecessary computation at every timestamp, an incremental approach of outlier detection is proposed which makes use of outlier detection results obtained from previous timestamp to detect outliers in current timestamp. Moreover, an approximate continuous outlier detection approach using the bounded Gaussian uncertainty is proposed to further reduce the cost of incremental outlier detection. A cell-based approach, similar to one discussed in Sec. 4.3 is utilized to reduce the cost of distance-based outlier detection within a state set. The exact continuous outlier detection approach using the conventional Gaussian uncertainty is denoted by *CUDB(CG)* and the approximate continuous outlier detection approach using the bounded Gaussian uncertainty is denoted by *CUDB(BG)* in the rest of the dissertation.

6.2 Problem Formulation

We defined the distance-based outliers on uncertain static datasets in chapter 4 as follows.

Definition 6.1 *An uncertain object o_i in a database \mathcal{GDB} is a distance-based outlier, if the expected number of objects lying within D -distance of o_i are less than or equal to threshold $\theta = N(1 - p)$, where N is the number of uncertain objects in \mathcal{GDB} , and p is the fraction of \mathcal{GDB} objects that lie farther than D -distance of o_i .*

However, the focus of this work is uncertain time series data streams, where streams are the sequences of objects' states generated over time. This work assumes that the states of all the objects are generated synchronously at every timestamp and the set of states at a timestamp is called a state set. It is further assumed that the objects' uncertainty follows the Gaussian distribution.

Formally, in this work, d -dimensional uncertain objects o_i are considered, with attribute vector $\vec{A}_i = (x_{i1}, \dots, x_{id})$ following the Gaussian distribution with mean $\vec{\mu}_i = (\mu_{i1}, \dots, \mu_{id})$ and co-variance matrix $\Sigma_i = \text{diag}(\sigma_{i1}^2, \dots, \sigma_{id}^2)$.

Namely, $\vec{\mathcal{A}}_i$ is a random variable that follows the Gaussian distribution $\vec{\mathcal{A}}_i \sim \mathcal{N}(\vec{\mu}_i, \Sigma_i)$. Assuming that there are N objects whose states may change over time, $S^j = \{\vec{\mathcal{A}}_1^j, \dots, \vec{\mathcal{A}}_N^j\}$ denotes a state set of N objects at time t_j . Note that the $\vec{\mu}_i^j$ denotes the observed coordinates (attribute values) of an object o_i at time t_j . Hence, Def. 6.1 can be extended naturally for uncertain time-series data streams as follows.

Definition 6.2 *An uncertain object o_i is a distance-based outlier at time t_j , if the expected number of objects in S^j lying within D -distance of o_i are less than or equal to threshold $\theta = N(1 - p)$, where p is the fraction of objects that lie farther than D -distance of $o_i \in S^j$.*

The objects that lie within the D -distance of o_i are called its D -neighbors, and the set of the D -neighbors of o_i and the number of D -neighbours are denoted by $DN(o_i)$ and $\#D\text{-neighbors}(o_i)$, respectively. In order to find the distance-based outliers from the state set S^j at time t_j , the $\#D$ -neighbors of the un-pruned objects needs to be computed which requires the computation of distance probability. This distance probability is computed using the difference between two uncertain objects, which is given by another distribution known as the Gaussian difference distribution [114]. For example, if attributes $\vec{\mathcal{A}}_p$ and $\vec{\mathcal{A}}_q$ of objects o_p and o_q , respectively follow the Gaussian distribution, then $|\vec{\mathcal{A}}_p - \vec{\mathcal{A}}_q| = \mathcal{N}(\vec{\mu}_p - \vec{\mu}_q, \Sigma_p + \Sigma_q)$ also follows the Gaussian distribution [114]. Let $Pr(o_p, o_q, D)$ denotes the probability that $o_q \in DN(o_p)$. Then,

$$Pr(o_p, o_q, D) = \int_R \mathcal{N}(\vec{\mu}_p - \vec{\mu}_q, \Sigma_p + \Sigma_q) d\vec{\mathcal{A}}, \quad (6.1)$$

where R is a sphere with centre $(\vec{\mu}_p - \vec{\mu}_q)$ and radius D . For the expression and derivation of $Pr(o_p, o_q, D)$, please refer Sec. 4.2. Furthermore, we will use $Pr(\alpha, D)$ to denote $Pr(o_p, o_q, D)$ when there is no confusion, where α is an ordinary Euclidean distance between the means of o_p and o_q . Computing this probability is usually very costly, and it gets more expensive with the increase in data dimensionality.

In the following part, the discussion focuses on 2-dimensional case. However, the solution can be extended to higher dimensional cases without loss of generality. In addition, this work assumes that the standard deviations are uniform in all dimensions of an object, to keep the discussion simple. The proposed

incremental outlier detection approaches make use of results obtained from previous state set (S^{j-1}) to detect outliers in current state set (S^j). However, to reduce the cost of distance-based outlier detection within a state set, a cell-based approach similar to our previous work [96] is utilized. Hence in Sec. 6.3, a quick overview of the cell-based outlier detection approach [96] is presented. The proposed continuous outlier detection and the approximate continuous outlier detection approaches are presented in Secs. 6.4 and 6.5, respectively.

6.3 Cell-based Outlier Detection

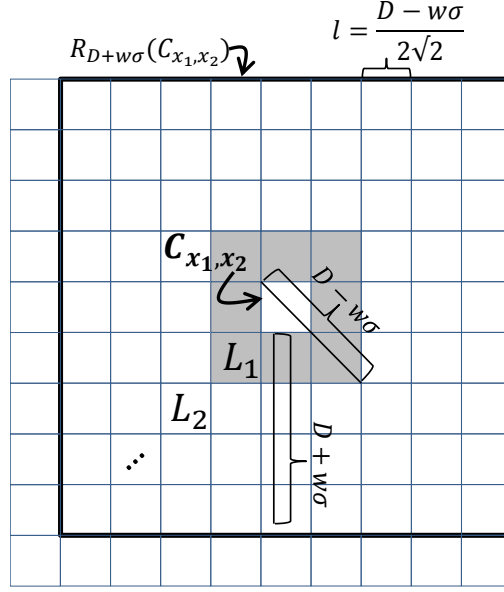
The Naive approach of the distance-based outlier detection within a state set S^j , is the use of nested loop to find the $\#D$ -neighbors of each $o_i \in S^j$. This approach is very expensive and on average it requires $O(N^2)$ expensive probability ($Pr(o_i, o_j, D)$) evaluations. To reduce this costly computation, a cell-based approach [96] is used in this work.

The cell-based approach is aimed at reducing the number of costly probability evaluations. It maps a dataset objects to a cell-grid and identify the cells containing only inliers or outliers based on the bounds on the $\#D$ -neighbors. This approach will be utilized in Secs. 6.4 and 6.5 for the proposed continuous outlier detection approaches.

6.3.1 Grid (\mathcal{G}) Structure

To identify distance-based outliers using the cell-based approach, the objects in the state set S^j are mapped to a 2-dimensional space that is partitioned into cells of length $l = \frac{D-w\sigma}{2\sqrt{2}}$ as shown in Fig. 6.1. Let C_{x_1, x_2} be a cell at the intersection of row x_1 and column x_2 . The Layer 1 (L_1) neighbors of C_{x_1, x_2} are the immediate neighbouring cells of C_{x_1, x_2} , defined as follows and satisfies property 1.

$$L_1(C_{x_1, x_2}) = \{C_{u_1, u_2} | u_1 = x_1 \pm 1, u_2 = x_2 \pm 1, C_{u_1, u_2} \neq C_{x_1, x_2}\}.$$

Figure 6.1: Cell Grid (\mathcal{G})

Property 1: If $C_{u_1, u_2} \in L_1(C_{x_1, x_2})$, then $o_p \in C_{x_1, x_2}$ and $o_q \in C_{u_1, u_2}$ are at most $D - w\sigma$ apart.

From Property 1 and Lemma 5.1, $Pr(o_p, o_q, D) \approx 1$. The Layer 2 (L_2) neighbors of C_{x_1, x_2} are the immediate neighbouring cells of $L_1(C_{x_1, x_2})$ and are defined as follows.

$$L_2(C_{x_1, x_2}) = \{C_{u_1, u_2} | u_1 = x_1 \pm 2, u_2 = x_2 \pm 2, C_{u_1, u_2} \notin L_1(C_{x_1, x_2}), \\ C_{u_1, u_2} \neq C_{x_1, x_2}\}.$$

$L_3(C_{x_1, x_2})$ and higher layers are defined in a similar way. Let $n_{D+w\sigma} = \lceil \frac{D+w\sigma}{l} \rceil$, then the region $R_{D+w\sigma}$ of C_{x_1, x_2} is defined as follows and satisfies property 2.

$$R_{D+w\sigma}(C_{x_1, x_2}) = \{C_{u_1, u_2} | u_1 = x_1 \pm n_{D+w\sigma}, u_2 = x_2 \pm n_{D+w\sigma}, \\ C_{u_1, u_2} \notin L_1(C_{x_1, x_2}), C_{u_1, u_2} \neq C_{x_1, x_2}\}.$$

Property 2: If C_{u_1, u_2} is neither an L_1 nor an $R_{D+w\sigma}$ neighbour of C_{x_1, x_2} and $C_{u_1, u_2} \neq C_{x_1, x_2}$, then $o_p \in C_{x_1, x_2}$ and $o_q \in C_{u_1, u_2}$ are at least $D + w\sigma$ apart.

From Property 2 and Lemma 5.1, $Pr(o_p, o_q, D) \approx 0$.

6.3.2 Cell Bounds

Cell bounds on $\#D$ -neighbors are computed to prune a cell as inlier or outlier, without expensive object-wise distance computation. The upper bound of C_{x_1, x_2} , $UB(C_{x_1, x_2})$, binds the maximum $\#D$ -neighbors in the \mathcal{G} for any object in C_{x_1, x_2} , and is defined as follows.

$$UB(C_{x_1, x_2}) = N(C_{x_1, x_2}) + \sum_{m=1}^{n_{D+w\sigma}} N(L_m(C_{x_1, x_2})) \times Pr((m-1)l, D) + \\ (N - N(C_{x_1, x_2}) - \sum_{m=1}^{n_{D+w\sigma}} N(L_m(C_{x_1, x_2}))) \times Pr(n_{D+w\sigma}l, D),$$

where $N(\cdot)$ denotes the number of objects. On the other hand, the lower bound of C_{x_1, x_2} , $LB(C_{x_1, x_2})$, binds the minimum $\#D$ -neighbors in the \mathcal{G} for any object in C_{x_1, x_2} and is defined as follows.

$$LB(C_{x_1, x_2}) = 1 + (N(C_{x_1, x_2}) - 1) \times Pr(\sqrt{2}l, D) + \\ \sum_{m=1}^{n_{D+w\sigma}} N(L_m(C_{x_1, x_2})) \times Pr((m+1)\sqrt{2}l, D).$$

Since $Pr(\alpha, D)$ value is dependent on α and is independent from the location of C_{x_1, x_2} , the $Pr(\alpha, D)$ values required for the bounds computation are pre-computed to reduce the cost of the bounds computation. The $Pr(\alpha, D)$ values need to be computed only for $\alpha = m\sqrt{2}l$ ($1 \leq m \leq n_{D+w\sigma} + 1$) and $\alpha = ml$ ($0 \leq m \leq n_{D+w\sigma} - 1$). The pre-computed values are stored in a lookup table to be used for the bounds computation.

6.3.3 Cell Pruning

Let $\theta' = \left\lceil \frac{\theta}{Pr(D-w\sigma, D)} \right\rceil$, where θ is threshold and is dependent on parameter p , then the grid cells can be pruned using the following property.

Property 3:

1. If $N(C_{x_1, x_2}) > \theta'$, all the objects in C_{x_1, x_2} and $L_1(C_{x_1, x_2})$ are inliers.
2. If $N(C_{x_1, x_2}) + N(L_1(C_{x_1, x_2})) > \theta'$, all the objects in C_{x_1, x_2} are inliers.

3. If $LB(C_{x_1, x_2}) > \theta$, all the objects in C_{x_1, x_2} are inliers.
4. If $UB(C_{x_1, x_2}) \leq \theta$, all the objects in C_{x_1, x_2} are outliers.

For all the un-pruned objects in the un-pruned cells, the Naive approach is used to find their $\#D$ -neighbors, to determine whether the un-pruned objects are inliers or outliers.

6.4 Incremental Outlier Detection

This section presents the proposed continuous outlier detection approach for time series data streams. Streams are the sequence of objects' states generated over time. We assume that the states of all the objects are generated synchronously and the set of states at a timestamp t_j is called a state set S^j , as shown in Fig. 6.2. The straightforward approach to detect outliers from each state set is to use the cell-based approach discussed in Sec. 6.3 for every timestamp. However, it is the wastage of computational resources since the duration between two consecutive timestamps is very short and the state of all the objects may not change in this duration. Hence, we propose an incremental approach of outlier detection, which makes use of outlier detection results obtained from state set S^{j-1} at t_{j-1} to detect outliers in state set S^j at t_j . This eliminates the need to process all the objects' states at every timestamp and saves a lot of computation time.

Time	State set	o_1	o_2	...	o_N
t_1	S^1	$\vec{\mathcal{A}}_1^1$	$\vec{\mathcal{A}}_2^1$...	$\vec{\mathcal{A}}_N^1$
t_2	S^2	$\vec{\mathcal{A}}_1^2$	$\vec{\mathcal{A}}_2^2$...	$\vec{\mathcal{A}}_N^2$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
t_{j-1}	S^{j-1}	$\vec{\mathcal{A}}_1^{j-1}$	$\vec{\mathcal{A}}_2^{j-1}$...	$\vec{\mathcal{A}}_N^{j-1}$
t_j	S^j	$\vec{\mathcal{A}}_1^j$	$\vec{\mathcal{A}}_2^j$...	$\vec{\mathcal{A}}_N^j$

Figure 6.2: State sets

Let SC^j at timestamp t_j denotes a set of objects whose states change between timestamps t_{j-1} and t_j . We call such objects, SC-objects (state-change

objects). Note that $SC^j \subseteq S^j$. The main idea of the incremental outlier processing is to process only the objects which are either SC-objects or are affected by the SC-objects. We will utilize the cell-based algorithm discussed in Sec. 6.3 to process only the SC-objects. The proposed incremental approach targets all state sets except the initial state set (S^1). For the S^1 , no results are available from the previous state set, hence all the objects in the S^1 need to be processed using the cell-based approach. To simplify the problem, consider the case with one SC-object, o_p . Let C_{x_1, x_2}^j represents a cell C_{x_1, x_2} at time t_j . As a result of state change, $o_p \in \mathcal{G}$ can move in the following two ways.

[Case 1] o_p moved to a different cell:

$$o_p \in C_{x_1, x_2}^{j-1}, o_p \in C_{x_1, x_2}^j, C_{x_1, x_2}^{j-1} \neq C_{x_1, x_2}^j.$$

[Case 2] o_p moved within a cell:

$$o_p \in C_{x_1, x_2}^{j-1}, o_p \in C_{x_1, x_2}^j, C_{x_1, x_2}^{j-1} = C_{x_1, x_2}^j.$$

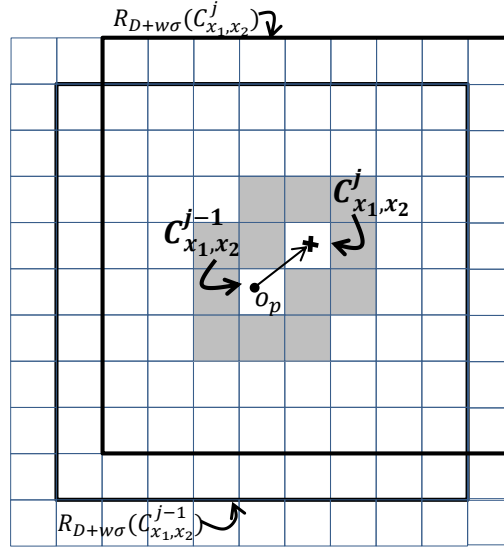


Figure 6.3: o_p moved among cells

Recall that in the cell-based approach, for the computation of cell bounds on $\#D$ -neighbors, the cells within region $R_{D+w\sigma}$ are considered and for the upper bound, the cells outside the region $R_{D+w\sigma}$ are also considered. Hence, when an object moves among cells (case 1), it affects the cell bounds of all the cells within region $R_{D+w\sigma}$ of the C_{x_1, x_2}^{j-1} and C_{x_1, x_2}^j and the $\#D$ -neighbors of all the

un-pruned objects in the \mathcal{G} . Namely in case 1, o_p affects cells C_{x_1, x_2}^{j-1} , C_{x_1, x_2}^j , their L_1 and $R_{D+w\sigma}$ neighbors and all the objects in un-pruned cells in the \mathcal{G} . This movement does not affect the cell-based pruned cells outside $R_{D+w\sigma}$ region, because the number of objects outside the region $R_{D+w\sigma}$ is not affected by this movement. Fig. 6.3 shows the movement of o_p between C_{x_1, x_2}^{j-1} and C_{x_1, x_2}^j and their L_1 and $R_{D+w\sigma}$ neighbors.

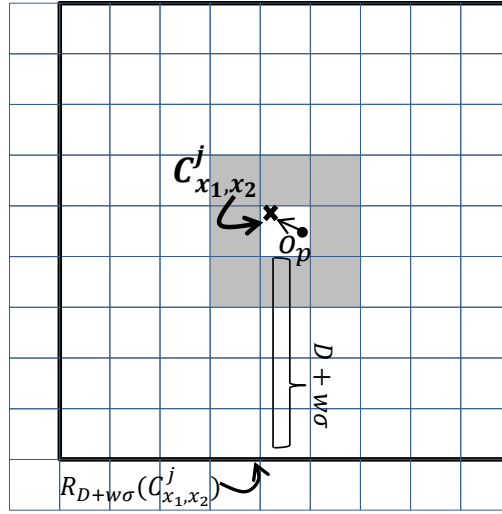


Figure 6.4: o_p moved within cell

On the other hand, when an object moves within a cell (case 2), it does not affect the bounds of any grid cell, however, this movement affects the $\#D$ -neighbors of all the un-pruned objects in the \mathcal{G} . Fig. 6.4 shows the movement of o_p between C_{x_1, x_2}^{j-1} and C_{x_1, x_2}^j where $C_{x_1, x_2}^{j-1} = C_{x_1, x_2}^j$. We call the cells affected by the SC-objects *target cells*. Target cells require re-outlier detection with the arrival of new state set.

Target Cells

In practise, there are more than one SC-objects between t_{j-1} to t_j . Therefore, we expand the idea to more than one SC-objects. Hence, the target cells can be classified into following 3 types.

Type A: Cells containing SC-objects which have moved to or from another cell at time t_j (C_{x_1, x_2}^{j-1} and C_{x_1, x_2}^j in Fig. 6.3).

Algorithm 6.1: CUDB Outlier Detection: Incremental Approach

Input: S^j, \mathcal{G}, θ
Output: Set of distance-based outliers \mathcal{O}
 /*Identifying state change objects*/

- 1: **for each** $o_i \in S^j$ **do**
- 2: **if** o_i is case-1 SC-object **then**
- 3: Add o_i to appropriate cell C^j , increase $N(C^j)$ by 1;
- 4: Delete o_i from its old cell C^{j-1} , decrease $N(C^{j-1})$ by 1;
- 5: Label C^j and C^{j-1} **A** and their L_1 and $R_{D+w\sigma}$ neighbouring cells **B**;
- 6: **end if**
- 7: **end for**
- 8: Label each $C^j \in \mathcal{G}$ **C**, if C^j is non-empty, un-pruned and not labelled **A** or **B**;
- /*Processing cells of types A, B and C*/
- 9: **for each** $C \in \mathcal{G}$ **do**
- 10: **if** C is labelled **A**, **B** or **C**, process them using cell-based approach of Sec. 6.3 (Algorithm 4.2) and obtain set of outliers \mathcal{O} ;
- 11: **end for**
- 12: **for each** o_i in un-pruned cells **do**
- 13: Compute $\#D$ -neighbors(o_i) using the $Pr(o_i, o, D)$ values available in Hash table; ($Pr(o_i, o, D)$ computation is required if either o_i or o or both are SC-objects or $Pr(o_i, o, D)$ is not available in the Hash table)
- 14: **if** $\#D$ -neighbors(o_i) $\leq \theta$ **then** o_i is outlier. Add o_i to \mathcal{O} ;
- 15: **end for**
- 16: **return** \mathcal{O} ;

Type B: L_1 and $R_{D+w\sigma}$ neighbouring cells of Type A cells except those classified as Type A.

Type C: Un-pruned cells of the grid \mathcal{G} . Type C cells may include Type A and B cells.

All three cell types, i.e., A, B and C require re-outlier detection with the arrival of new state set, while rest of the cells do not need to be processed. Lines 1-11 of the incremental algorithm (Algorithm 6.1) is used for the processing of target cells.

Un-pruned Objects Processing

Due to expensive $\#D$ -neighbors computation, the main cost of our proposed incremental algorithm (CUDB(CG)) lies in the processing of un-pruned objects (Type C cells). The $\#D$ -neighbors computation of an object o_p requires distance probability computation, $Pr(o_p, o_q, D)$, between o_p and each $o_q \in S^j$. In the CUDB(CG) algorithm, this cost can be reduced by utilizing the $Pr(o_p, o_q, D)$ values computed in processing the previous state set. Namely, a Hash table is used to store $Pr(o_p, o_q, D)$ values computed at time t_{j-1} . At time t_j , these values are retrieved from the Hash table in $O(1)$ time. Hence at time t_j , $Pr(o_p, o_q, D)$ values need to be computed only in two cases; 1) States of o_p, o_q or both have changed, 2) $Pr(o_p, o_q, D)$ is not available in the Hash table. Since un-pruned objects form a fraction of the state set, the memory required to hold the Hash table is not significant. However it saves a lot of computation time. Lines 12-15 of Algorithm 6.1 shows the processing of un-pruned objects.

6.5 Incremental Outlier Detection using the Bounded Gaussian Uncertainty

Approximating the Gaussian uncertainty by the bounded Gaussian uncertainty enables an approximate but more efficient outlier detection. According to this paper's assumption, attributes of uncertain objects follow the Gaussian distribution. Therefore according to the 3-sigma rule there is a 95.45% chance that uncertain objects' attribute values lie within 2 standard deviations of the observed values and 99.73% chance that the values lie within 3 standard deviations of the observed values [88]. Hence the conventional Gaussian distribution can be normalized within certain boundaries to increase the efficiency of outlier detection at a small cost of accuracy.

Given a two dimensional conventional Gaussian function $g_{\vec{\lambda}}(x_1, x_2)$ with mean $\vec{\mu} = (\mu_1, \mu_2)$ and co-variance matrix $\Sigma = \text{diag}(\sigma^2, \sigma^2)$, the bounded Gaussian distribution $f_{\vec{\lambda}}(x_1, x_2)$ can be defined following the practise of [108],

as follows.

$$f_{\vec{\mathcal{A}}}(x_1, x_2) = \begin{cases} \frac{g_{\vec{\mathcal{A}}}(x_1, x_2)}{\int_{(x_1, x_2) \in o.ur} g_{\vec{\mathcal{A}}}(x_1, x_2) dx_1 dx_2} & (x_1, x_2) \in o.ur \\ 0 & otherwise \end{cases} \quad (6.2)$$

where $o.ur$ denotes the uncertainty region of the bounded Gaussian distribution. This paper assumes that the uncertainty region is a sphere with centre (μ_1, μ_2) and radius r .

By bounding the Gaussian uncertainty, a cell can be pruned by simply counting the number of objects in its neighbouring layers. Moreover the major cost of outlier detection, that is, the processing of un-pruned objects also reduces significantly. This is because, with the bounded Gaussian, the outlier detection algorithm needs to consider limited number of objects for the computation of an object's $\#D$ -neighbors rather than all the objects, as in the case of the conventional Gaussian uncertainty. For details on the bounded Gaussian uncertainty, please refer Sec. 4.4.

6.5.1 Bounded Gaussian Cell-based Outlier Detection

Like the grid structure of the conventional Gaussian uncertainty in Sec. 6.3.1, objects in the state set S^j are mapped to a 2-dimensional space that is partitioned into cells of length $l = \frac{D-2r}{2\sqrt{2}}$. The cell layers are defined in a similar manner as that of Sec. 6.3.1 and cell length l is chosen in such a way to satisfy the property 4.

Property 4: If $C_{u_1, u_2} \in L_1(C_{x_1, x_2})$, then $o_p \in C_{x_1, x_2}$ and $o_q \in C_{u_1, u_2}$ are at most $D - 2r$ distance apart.

From property 4, an $o_p \in C_{x_1, x_2}$ and an $o_q \in C_{u_1, u_2}$ are guaranteed to be D -neighbors mutually, with the $Pr(o_p, o_q, D) = 1$. Cells in region $R_{D+2r}(C_{x_1, x_2})$ are those which fall within $D + 2r$ distance of C_{x_1, x_2} . Let $n_{D+2r} = \lceil \frac{D+2r}{l} \rceil$, then the region R_{D+2r} of C_{x_1, x_2} is defined as follows and satisfies property 5.

$$R_{D+2r}(C_{x_1, x_2}) = \{C_{u_1, u_2} | u_1 = x_1 \pm n_{D+2r}, u_2 = x_2 \pm n_{D+2r}, \\ C_{u_1, u_2} \notin L_1(C_{x_1, x_2}), C_{u_1, u_2} \neq C_{x_1, x_2}\}.$$

Property 5: If C_{u_1, u_2} is neither an L_1 nor an R_{D+2r} neighbour of C_{x_1, x_2} and $C_{u_1, u_2} \neq C_{x_1, x_2}$, then $o_p \in C_{x_1, x_2}$ and $o_q \in C_{u_1, u_2}$ are at least $D + 2r$ apart.

From property 5, it can be guaranteed that an $o_p \in C_{x_1, x_2}$ and an $o_q \in C_{u_1, u_2}$ are greater than $D + 2r$ distance apart, hence $Pr(o_p, o_q, D) = 0$. Using the properties 4 and 5, grid cells can be pruned as follows.

Property 6:

1. If $N(C_{x_1, x_2}) > \theta$, all the objects in C_{x_1, x_2} and $L_1(C_{x_1, x_2})$ are inliers.
2. If $N(C_{x_1, x_2}) + N(L_1(C_{x_1, x_2})) > \theta$, all the objects in C_{x_1, x_2} are inliers.
3. If $N(C_{x_1, x_2}) + N(L_1(C_{x_1, x_2})) + N(R_{D+2r}(C_{x_1, x_2})) \leq \theta$, all the objects in C_{x_1, x_2} are outliers.

For the un-pruned objects in the un-pruned cells from the cell-based pruning, $\#D$ -neighbors are computed using only the objects within $D + 2r$ distance of the target object, to find whether the un-pruned objects are inliers or outliers.

6.5.2 Bounded Gaussian Incremental Outlier Detection

As a result of state change, an object can either move within the cell or among the cells. Hence the two cases are similar to the one discussed in Sec. 6.4 for the conventional Gaussian uncertainty. Since the uncertainty of an object is bounded, as a result of change in their state, fewer number of neighbouring cells are affected as compared to the conventional Gaussian uncertainty, reducing the number of target cells requiring re-outlier detection. Similarly, *Type C* cells now do not contain all the un-pruned cells of the grid. However, they contain only the un-pruned cells within region R_{D+2r} of the cell containing an SC-object. As a result, the computational cost of the incremental outlier detection algorithm using the bounded Gaussian uncertainty reduces significantly than the conventional Gaussian uncertainty.

6.6 Complexity Analysis

Since the basic cell-based algorithms used by the incremental outlier detection approaches is UDB outlier detection algorithms (Algorithms 4.2 and 4.4), we

will not derive the complexities of the CUDB outlier detection algorithms from scratch. However, we will use the complexities of the UDB outlier detection algorithms to derive the complexities of the CUDB outlier detection algorithms in this section. From the UDB outlier detection algorithms, the major cost lies in the evaluation of accurate $\#D$ -neighbours of the un-pruned objects. This cost is so high that it hides the cost of the rest of the algorithms and therefore the complexities of the UDB outlier detection algorithms were given in terms of the number of distance probability evaluations. Hence the average case and the worst case complexities of the UDB(CG) outlier detection algorithm obtained in Sec. 4.3.4 are $O(nN)$ and $O(N^2)$ respectively, where N is the number of dataset objects and $n \ll N$ is the number of un-pruned objects. The average case and the worst case time complexities of the UDB(BG) outlier detection algorithm are $O(nN')$ and $O(N^2)$ respectively, where $N' \leq N$ is the number of objects that lie within $D + 2r$ distance of the un-pruned objects.

In the CUDB algorithms (the CUDB(CG) and the CUDB(BG)), the UDB outlier detection algorithms are utilized only for the objects affected by the SC-objects. Assuming that there are n_{sc} objects affected by the SC-objects and $n_{sc} \leq n$, then the average case computation complexities of the CUDB(CG) and the CUDB(BG) algorithms are given by $O(n_{sc}N)$ and $O(n_{sc}N')$, respectively for the 2D case. However, the worst case complexity remains same, i.e., $O(N^2)$ for the CUDB(CG) and the CUDB(BG) algorithms, assuming that all the objects are affected by the SC-objects. Hence, the average computation complexities of the CUDB algorithms remain less than the UDB algorithms, as long as the $n_{sc} < n$.

The complexities of the CUDB outlier detection algorithms do not change with the increase in dimensions d , as long as only the number of distance probability evaluations are considered for the computation of the algorithms' complexities. Although with the increase in d , the number of grid cells increases exponentially, yet the cost of evaluation of the $\#D$ -neighbours of the un-pruned objects remains dominant and hence the complexities remain same (as discussed above for the 2D case) for the higher dimensional case for both the CUDB algorithms, i.e., the CUDB(CG) and the CUDB(BG).

The stream data arrive continuously and at high speed and has high memory requirements, therefore the memory complexity of the CUDB algorithms is also presented. According to the problem formulation of this chapter, streams are

the sequences of objects' states generated over time and a state set is a set of uncertain objects' states that are originated synchronously from a group of data sources. The proposed incremental algorithms make use of the results obtained from the previous state set to efficiently detect outliers in the current state set. Hence, two state sets need to be kept in memory always, a current and a previous, each of which require $O(N)$ memory. In addition, a hash table is used by all the un-pruned objects, each of which requires $O(N)$ memory. Hence, on average the memory complexity of the CUDB algorithms is $O(nN)$, where N is the number of dataset objects and $n \ll N$ is the number of un-pruned objects. In the worst case, none of the dataset objects is pruned by the cell-based pruning, resulting in $O(N^2)$ memory complexity.

6.7 Experiments

We conducted extensive experiments on synthetic and real datasets to evaluate the effectiveness of the proposed approaches. All algorithms were implemented in C++, GNU compiler. All experiments were performed on a system with an Intel Core 2 Duo CPU E8400 3.00GHz CPU and 2GB main memory running Ubuntu 12.04 OS. All programs run in main memory and no I/O cost is considered. Each experiment is performed 3 times and the average values are used in the graphs. We have also used error bars in the graphs, showing the standard error in the execution time measurements of each approach. However, in majority of the graphs they are not visible due very small standard error in the execution times.

Pre-computation time is not included in the measurements. Unless specified, the following parameter values are used in experiments: $D = 100$, $\sigma = 10$, $t = 2$, $r = t\sigma$ and $p = 0.998$. SC-objects ratio = 30% is used for the UG dataset (SC-objects ratio for the MOW dataset depends on the number of objects changing states between two forecasts). In figures, the incremental approaches using the conventional Gaussian and the bounded Gaussian uncertainties are denoted by the CUDB(CG) and the CUDB(BG), respectively. These two approaches are compared with the simple cell-based approaches on the conventional and the bounded Gaussian uncertainties, presented in chapter 4. In the simple cell-based approaches, the cell-based algorithms presented in Secs. 4.3 and 4.4 are executed for all the objects in the state set at every timestamp. The simple cell-

based approaches are denoted by the $UDB(CG)$ and the $UDB(BG)$, respectively in the figures.

6.7.1 Datasets

In this work one synthetic and one real datasets are used for experiments. Synthetic dataset, uni-modal Gaussian (UG) is 2-dimensional and is generated using Box Muller method [110]. This method generates pair of independent, standard, normally distributed (zero mean, unit variance) random numbers, given a source of uniformly distributed random numbers. In order to provide the stream behaviour in synthetic dataset, states of the fraction of dataset objects are modified by adding normal random numbers with zero mean and standard deviation σ_{SC} at every time instance. The proposed incremental algorithm is executed on this modified dataset. Unless specified, the synthetic dataset consists of 5,000 tuples.

As for real-world data, we have used three hourly met office weather (MOW) forecast data available at [5]. In the experiments on real data, we have used a two dimensional subset of MOW data, which consists of screen and feels like temperature forecast values for 5,802 weather stations around UK. Consecutive forecasts are used as data stream in these experiments.

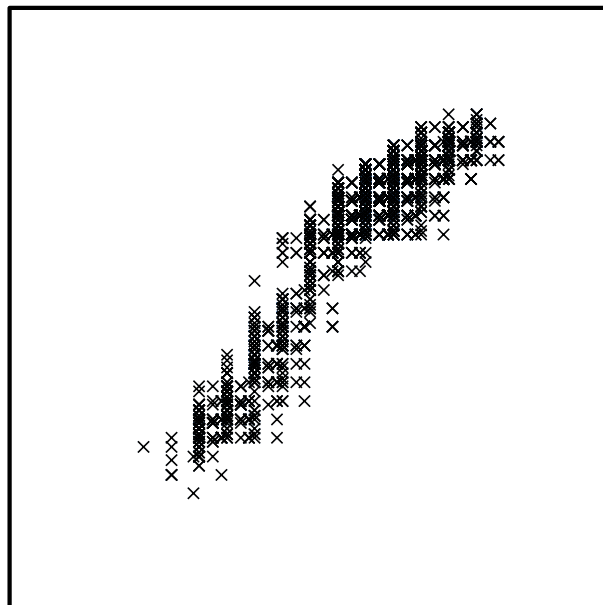


Figure 6.5: Met Office Weather (MOW) dataset [5]

Both the datasets are normalized to have a domain of $[0,1000]$ on every dimension. For each point z in the datasets, an uncertain object o is created, whose uncertainty is given by the Gaussian distribution with mean z and standard deviation σ in both the dimensions. The dataset UG used in the experiments is given in Fig. 4.5 and the dataset MOW is given in Fig. 6.5.

6.7.2 Results

Experiments are conducted to evaluate the efficiency and scalability of the proposed algorithms in this chapter. Since there are no known algorithms for distance-based outlier detection on uncertain data streams, the simple cell-based methods (the UDB(CG) and the UDB(BG)) are used as baseline. We did not perform experiments to evaluate the accuracy in this chapter, because the basic approach used to find the outliers is same as the one presented in chapter 4. Therefore, the accuracy of the experiments is similar to the one presented in Sec. 4.6.2.

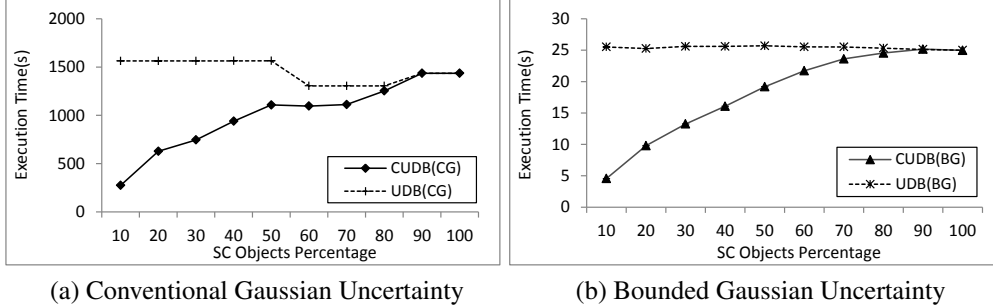


Figure 6.6: Varying SC-object's percentage for UG dataset ($D = 100$, $\sigma = 10$, $t = 2$, $r = t\sigma$, and $p = 0.998$)

Firstly, experiments are performed on the synthetic dataset UG by varying the percentage of SC-objects. Figs. 6.6a and 6.6b show the effect of varying the SC-objects' percentage on the execution time. In the figures, as the percentage of the SC-objects increases, the number of target cells requiring re-evaluation also increases. As a result the execution times of the proposed approaches the CUDB(CG) and the CUDB(BG) also increase. The graphs of the CUDB(CG) and the UDB(CG) in Fig. 6.6a meets when all the objects in the dataset change their states i.e., when the percentage of SC-Objects is 100%. At this point, the CUDB(CG) algorithm becomes similar to that of UDB(CG) algorithm, that is,

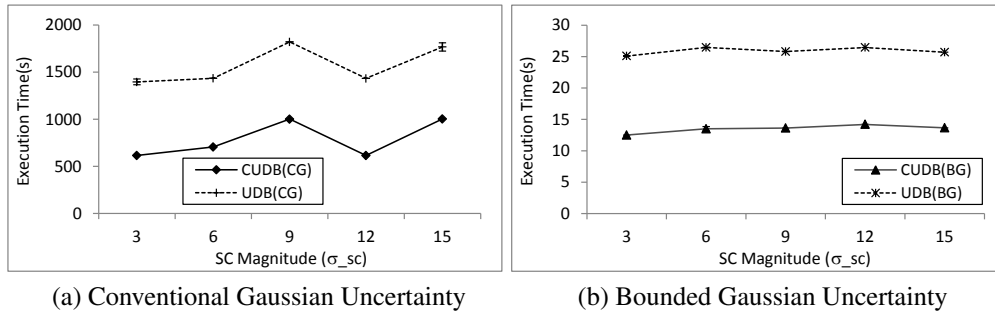


Figure 6.7: Varying SC magnitude for UG dataset ($D = 100$, SC-objects = 30%, $\sigma = 10$, $t = 2$, $r = t\sigma$, and $p = 0.998$)

executing cell-based algorithm for the complete state set rather than only for SC-objects. Similar trends can be observed for the CUDB(BG) and the UDB(BG) in Fig. 6.6b. As discussed in previous sections, the main cost of our algorithms lies in the processing of un-pruned objects. In the bounded Gaussian case, for the computation of $\#D$ -neighbors of un-pruned objects, limited number of objects within certain boundary are considered. However, the conventional Gaussian algorithm needs to consider all the object in the grid for the computation of $\#D$ -neighbors of an un-pruned object. Therefore the approach CUDB(BG) is far less expensive than the CUDB(CG). The variation in the UDB(CG) graph is due to the change in percentage of moving objects, which results in the variation in the state set distribution and consequently in the number of outliers returned by the algorithms.

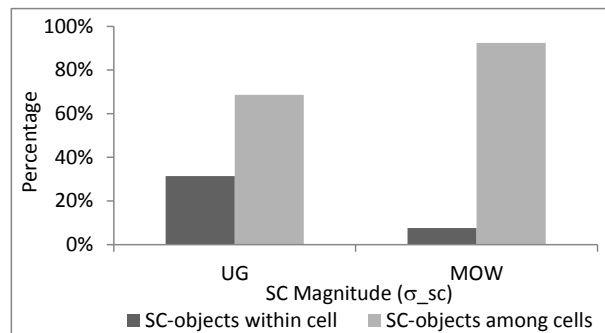


Figure 6.8: SC-objects within and among cells ($D = 100$, $\sigma = 10$, SC-objects = 30%, $t = 2$, $r = t\sigma$, and $p = 0.998$)

Next, experiments are performed by varying the magnitude of movement of SC-objects. The magnitude was varied by varying the σ_{sc} . From Fig. 6.7,

we can observe that as σ_{sc} increases, the execution times of all the algorithms increase. This is due to the fact that the increase in σ_{sc} , results in an increase in the number of objects that move among cells. Hence the number of target cells increases, which is the cause of increase in the execution times. Furthermore, from Fig. 6.7, the proposed incremental approaches are computationally less expensive than executing the cell-based algorithm for all the dataset objects. Moreover, it can be observed from Figs. 6.7a and 6.7b that the approach using the bounded Gaussian uncertainty is far less expensive computationally than the approach using the conventional Gaussian uncertainty. This is due to the fact that the bounded Gaussian approach requires limited number of objects for the computation of un-pruned objects' $\#D$ -neighbors. On the other hand, the conventional Gaussian approach needs to consider all the objects in the dataset for the computation of un-pruned objects' $\#D$ -neighbors.

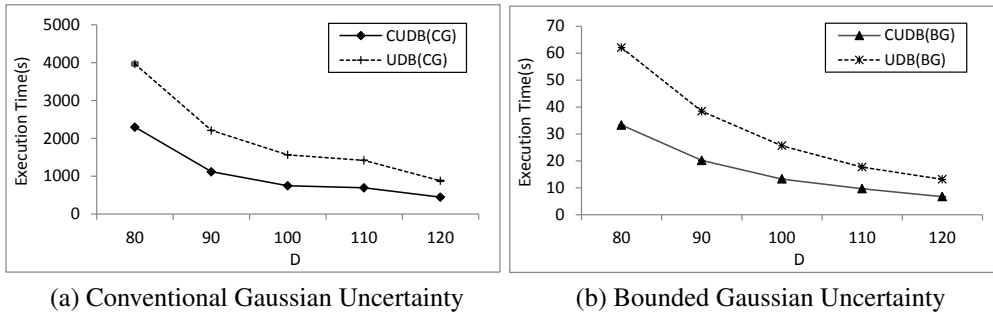


Figure 6.9: Varying parameter D for UG dataset ($\sigma = 10$, SC-objects = 30%, $t = 2$, $r = t\sigma$, and $p = 0.998$)

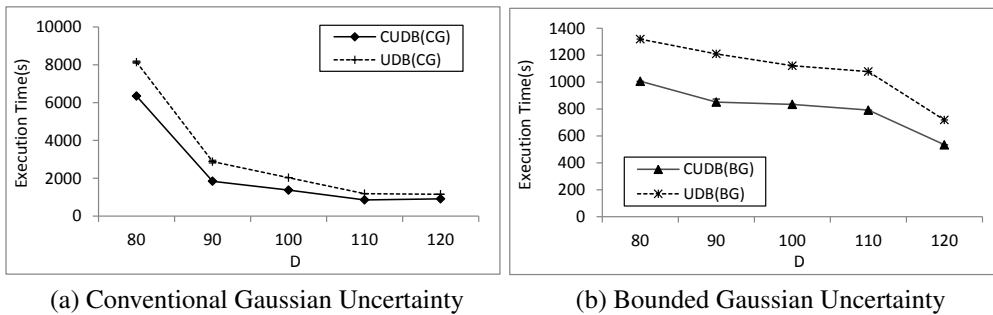


Figure 6.10: Varying parameter D for MOW dataset ($\sigma = 10$, SC-objects = 30%, $t = 2$, $r = t\sigma$, and $p = 0.998$)

Figure 6.8 shows the percentage of objects moving within and among cells

for both the datasets. Percentage of objects that move within and among the cells depend on the magnitude of SC-objects, i.e., the σ_{sc} . Larger the σ_{sc} , bigger the number of cells affected by the SC-objects which results in higher algorithm execution times.

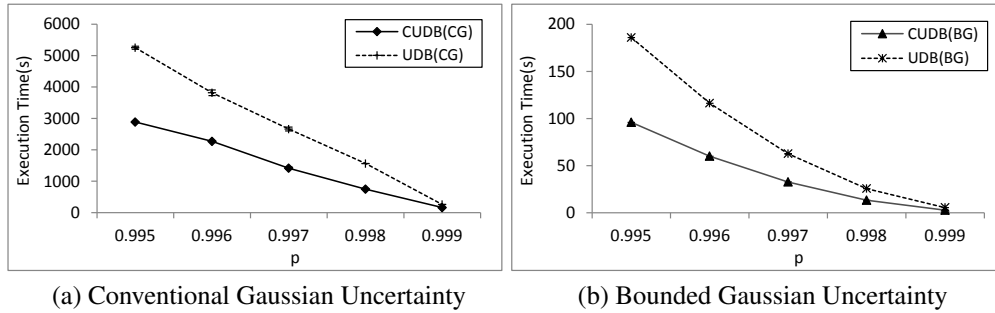


Figure 6.11: Varying parameter p for UG dataset ($D = 100$, SC-objects = 30%, $\sigma = 10$, $t = 2$, and $r = t\sigma$)

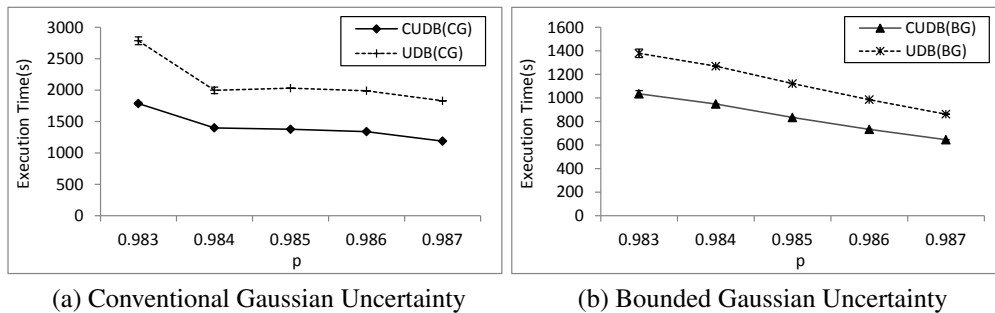


Figure 6.12: Varying parameter p for MOW dataset ($D = 100$, SC-objects = 30%, $\sigma = 10$, $t = 2$, and $r = t\sigma$)

Next, experiments are performed by varying the parameter D . As D increases, number of D -neighbors also increases. As a result, objects are more easily pruned and we obtain fewer outliers. This results in the decline in the execution times for all the algorithms and for both the datasets i.e., UG and MOW, as can be observed from Figs. 6.9 and 6.10. Again, it is quite obvious from the Figs. 6.9 and 6.10 that the CUDB(BG) is far less computationally expensive than the CUDB(CG).

We also performed experiments by varying the parameter p . As p increases, threshold θ and consequently the number of outliers returned by the algorithms decreases, which results in a decline in execution times of all the algorithms

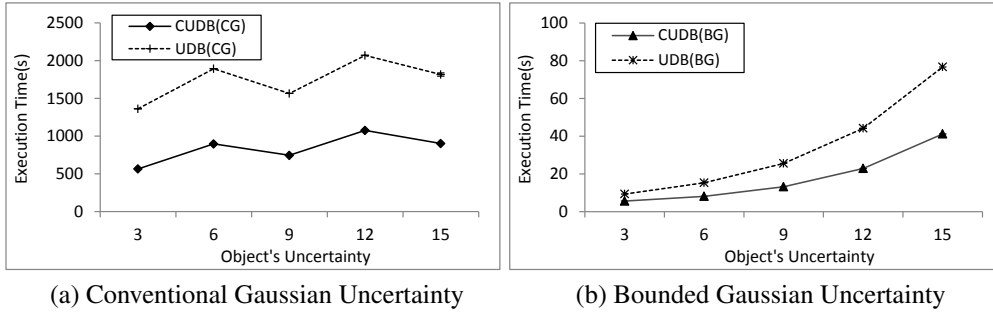


Figure 6.13: Varying object's uncertainty σ for UG dataset ($D = 100$, SC-objects = 30%, $t = 2$, $r = t\sigma$, and $p = 0.998$)

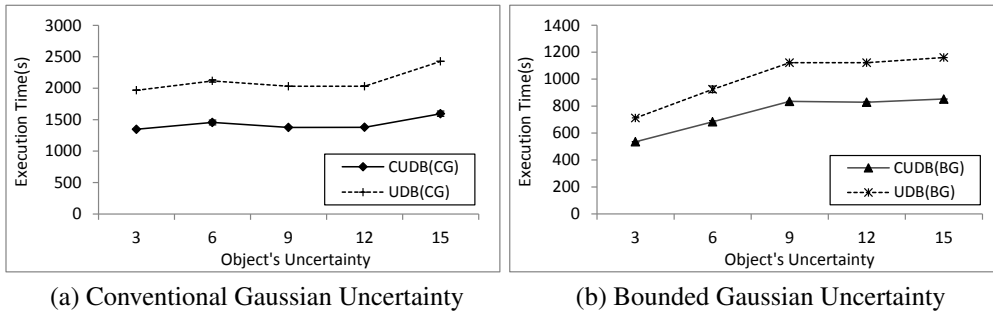


Figure 6.14: Varying object's uncertainty σ for MOW dataset ($D = 100$, SC-objects = 30%, $t = 2$, $r = t\sigma$, and $p = 0.998$)

for both the datasets. Please note that the proposed incremental approaches are faster than using the simple cell-based approach, in case of both uncertainty types, i.e., the conventional Gaussian and the bounded Gaussian. This can be observed from Figs. 6.11 and 6.12.

The effectiveness of the proposed approaches is also measured with the increasing level of objects' uncertainty. Figs. 6.13 and 6.14 show the effect of increasing the dataset objects' uncertainty level. As standard deviation (σ) increases, the uncertainty level of the objects increases which results in the decline in $Pr(o_p, o_q, D)$ values, even if the Euclidean distance between the o_p and the o_q is small. As a result, un-pruned objects are not pruned easily and their processing becomes expensive, which causes increase in the execution times of the algorithms.

Next, experiments are performed by varying the number of dimensions d . As d increases, the execution time of all the approaches increases. Moreover, the difference between the execution times for the incremental and the simple

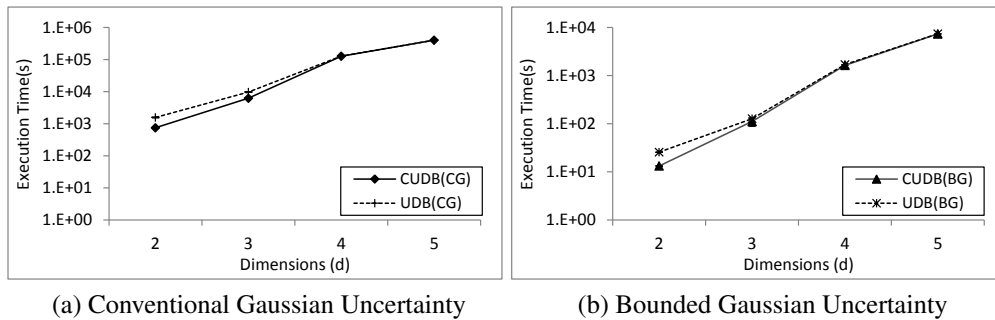


Figure 6.15: Varying dimensions d for UG dataset ($D = 100$, $\sigma = 10$, SC-objects = 30%, $t = 2$, $r = t\sigma$, and $p = 0.9997$)

cell-based approaches also decrease with the increase in d . This is due to the fact that with the increase in d , the number of cells increases; and the percentage of the time taken by the cell-based processing increases. Since in the CUDB(CG) and the CUDB(BG) approaches, the main benefit in the execution time, i.e., the reduction in the execution times, is obtained from the processing of the unpruned objects, however with the increase in d , this benefit reduces due to the shifting of the main cost of the algorithm from the unpruned objects processing to the cell-based processing. As a result, the proposed incremental outlier detection approaches do not remain much effective for the higher dimensional data as can be observed from Fig. 6.15.

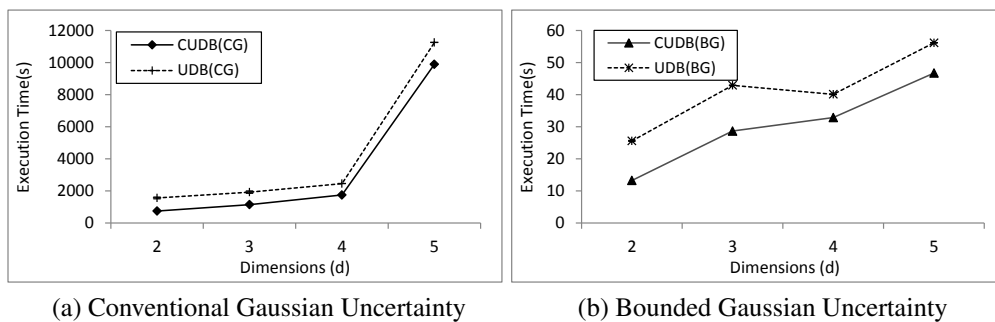


Figure 6.16: Varying dimensions d and parameter D for UG dataset ($\sigma = 10$, SC-objects = 30%, $t = 2$, $r = t\sigma$, and $p = 0.999$)

We also performed experiments by varying the number of dimensions d and the parameter D in parallel. In the experiments shown in Fig. 6.15, parameter D was kept constant, due to which the number of outliers increases dramatically with the increase in d . With the increase in d , objects get sparse and the

parameter D must be increased accordingly to find the limited number of outliers. Hence in Fig. 6.16, we increase the parameter D with the increase in d , in proportion to the average distance between the dataset objects. By doing so, we obtain almost same percentage of outliers from all the dimensions. Therefore, the execution times of all the approaches are far low in Fig. 6.16 as compared to the Fig. 6.15, specially for dimensions 3 and higher.

To summarize, as d increases the execution time of all the approaches increases, and the difference between the execution times for the incremental and the simple cell-based approaches decrease with the increase in d , due to the reasons discussed for the Fig. 6.15. As a result, the proposed incremental outlier detection approaches do not remain much effective for the higher dimensional data.

6.8 Summary

In this chapter, two continuous distance-based outlier detection approaches (an exact and an approximate) are proposed for uncertain time series data streams. The proposed approaches are based on the incremental processing of the state change objects, that is, they process only those objects which are affected by the change in objects' states. We employed a cell-based algorithm for the efficient detection of outliers within a state set in both the incremental algorithms. An extensive empirical study on synthetic and real datasets demonstrates the efficiency and scalability of the proposed approaches. From the experiments we found that the proposed incremental outlier detection techniques (CUDB(CG) and CUDB(BG)) are faster than the UDB(CG) and the UDB(BG). Moreover CUDB(BG) is computationally less expensive than the CUDB(CG), because the SC-Objects in the CUDB(BG) affects very few neighboring cells.

Chapter 7

Conclusions and Future Works

In this chapter, we conclude this dissertation by summarizing its main contributions and outlining some directions for future work.

7.1 Conclusions

In this dissertation, we addressed the problem of distance-based outlier detection on uncertain static data and uncertain data streams, and proposed three approaches. Two of the approaches *UDB Outlier Detection* and *kUDB Outlier Detection* are proposed for uncertain static data while one approach *CUDB Outlier Detection* is proposed for uncertain data streams. In the following subsections, we summarize the contributions of each of the proposed approaches.

7.1.1 UDB Outlier Detection

In chapter 4, we proposed a distance-based outlier technique on uncertain data. In the proposed work, an object's uncertainty is given by the Gaussian distribution. Since the distance computation between uncertain data objects is computationally expensive, we proposed a cell-based approach. The cell-based approach is capable of indexing the dataset objects besides speeding up the outlier detection process. The cell-based approach identifies and prunes the cells containing only inliers based on its bounds on outlier score ($\#D$ -neighbors). Similarly cell-based approach is also used for detecting the cells containing only outliers. Although the cell-based technique is very effective, yet it may leave some cells undecided, i.e., they are neither identified as inlier cells nor as outlier cells. For

the uncertain data objects in such cells the Naive computation follows, which is the use of nested loop to compute un-pruned objects' $\#D$ -neighbors.

The infinite nature of the Gaussian distribution makes it expensive to compute $\#D$ -neighbors of a dataset object, therefore to further reduce the computation cost of outlier detection, an approximate approach using the bounded Gaussian uncertainty is also proposed in this research. The basic idea is that the bounded Gaussian distribution is a good approximation of the Gaussian distribution and can increase the outlier detection efficiency at a small cost of accuracy. Finally, detailed experiments are performed on real and synthetic datasets to show the accuracy, efficiency and scalability of the proposed approaches. From the experiments, we found that the proposed approaches are more accurate than the baseline approach (Knorr et al. [63] of outlier detection). Furthermore, we found that the approximate approach using the bounded Gaussian uncertainty is computationally less expensive than the accurate approach, however it is less accurate as compared to the accurate one.

7.1.2 k UDB Outlier Detection

In chapter 5, a top- k approach of distance-based outlier detection is presented, which returns k objects with lowest outlier scores ($\#D$ -neighbors) or in other words, k strongest outliers along with their ranking. In order to compute the top- k outliers from uncertain datasets efficiently, a populated-cells list (PC-list) is proposed. The PC-list is a sorted list of non-empty cells of a d -dimensional grid, where the grid is used to index data objects. Using PC-list, the top- k outlier detection algorithm needs to consider only a fraction of the dataset objects and hence quickly identifies candidate objects for the top- k outliers. Finally, an exact outlier score ($\#D$ -neighbors) is computed for each candidate object to find the top- k outliers and their ranking.

From experiments we found that, computationally the most expensive part of the k UDB outlier detection approach is the computation of exact outlier scores of the candidate objects. In order to reduce this cost, two approximate k UDB outlier detection approach are also proposed. The first approximate algorithm, approximates only the candidate objects' $\#D$ -neighbors. According to our distance probability function, major contribution in $\#D$ -neighbors computation of an object is made by the nearer objects. Hence the first approximate algorithm,

approximates the candidate objects' $\#D$ -neighbors computation by considering only the nearer objects. The second approximate algorithm makes use of the bounded Gaussian uncertainty to increase the efficiency of the top- k outlier detection algorithm. Finally, detailed experiments on real and synthetic datasets are performed to prove the accuracy, efficiency and scalability of the proposed approaches. From the experiments, we found that the accuracy of outlier detection improves with the consideration of data uncertainty. In addition, we found that both of the approximate algorithms are several times faster than the exact counterpart.

7.1.3 CUDB Outlier Detection

In chapter 6, we proposed a continuous outlier detection technique for uncertain time series data streams. In particular, we presented a continuous distance-based outlier detection approach on a set of uncertain objects' states that are originated synchronously from a group of data sources (e.g., sensors in WSN) at every timestamp. A set of objects' states at a timestamp is called a state set. In this research we assume that the duration between two consecutive timestamps is very short and the state of all the objects may not change much in this duration. Therefore, we proposed an incremental approach of outlier detection, which makes use of results obtained from the previous state set to efficiently detect outliers in the current state set. An approximate incremental outlier detection approach using the bounded Gaussian uncertainty is also presented to further reduce the cost of incremental outlier detection. Finally, an extensive empirical study on synthetic and real datasets is presented. From experiments, we found that the incremental outlier detection algorithm is quite effective than using the cell-based algorithm for the complete dataset at every timestamp, specially when the percentage of state change objects is small.

7.2 Future Works

In the following, we present some possible future research issues and some suggestions to extend and improve the work presented in this dissertation.

7.2.1 Continuous Top- k Outlier Detection

In this dissertation, we proposed three approaches of outlier detection on uncertain datasets. Two of the approaches *UDB Outlier Detection* and *kUDB Outlier Detection* are proposed for uncertain static data while one approach *CUDB Outlier Detection* is proposed for uncertain data streams.

Just like the need of top- k outlier detection on uncertain static data, there is a need to obtain k strongest outliers and their ranking continuously on uncertain data streams. Hence, one of the natural extension of this work is *Continuous top- k outlier detection on uncertain data streams*.

7.2.2 UDB Outlier Detection for Very High Dimensional Data

In the current work, we have presented an outlier detection solution for relatively low-dimensional data. We performed experiments on at most 5-dimensional data. From the experimental evaluations in different chapters, it is very clear that the computation cost of the proposed solutions increase dramatically with the increase in dimensions. This is due to the fact that with the increase in dimensions, number of cells increase exponentially. Moreover, the probability computation $Pr(o_i, o_j, D)$ becomes too expensive for high dimensional data.

Hence, in order to detect outliers from very high dimensional uncertain data, dimensionality reduction and subspace mining techniques are required. In addition, an alternative of cell-based indexing/pruning is required. In our work, subspace outlier detection can be incorporated using the subspace distance-based outlier detection approach presented by Knorr et al. in [64] for deterministic data. With the increase in dimensions, the number of subspaces need to be considered for outlier detection increases exponentially and due to this reason identification of relevant subspaces for outlier detection is a known research problem. Authors in [64] proposed to select subspaces at random and search for outliers in them. Using this approach, if a subspace \mathcal{A} does not contain any outlier, then none of its subspaces $\mathcal{B} \subset \mathcal{A}$ can contain an outlier, in this way a lot of subspaces can be pruned from consideration. Experimentally, authors in [64] found $d = 3$ (d denotes the number of subspace dimensions) as a good starting point for random subspace selection, however further research is needed to identify relevant subspaces.

7.2.3 UDB Outlier Detection for General Uncertainty Models

In this dissertation, objects uncertainty is modelled by the Gaussian distribution. The Gaussian distribution is chosen to represent an object's uncertainty, because in Statistics it is the most important and the most commonly used. However, a general uncertainty model is required so that any distribution can be used to represent an object's uncertainty depending upon the requirements. For such an uncertainty model, the proposed cell-based pruning is not effective and we require a pruning technique which can handle data with arbitrary uncertainty distribution. A pruning technique presented by Tal et al. in [108] for finding nearest-neighbours from multi-dimensional uncertain data could be one option. In addition, an indexing technique is also required to index uncertain data.

Appendix

A: Probability Density Function for the Difference between Two Random Variables following the d -dimensional Gaussian Distributions

Let o be a k -dimensional uncertain object with attributes $\vec{\mathcal{A}} = (x_1, \dots, x_k)$, mean $\vec{\mu} = (\mu_1, \dots, \mu_k)^T$ and a diagonal covariance matrix $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_k^2)$. The probability density function of o can be expressed as follows.

$$f_{\vec{\mathcal{A}}}(x_1, \dots, x_k) = \frac{1}{\sqrt{(2\pi)^k \det \Sigma}} \exp \left\{ -\frac{(\vec{\mathcal{A}} - \vec{\mu})^T \Sigma^{-1} (\vec{\mathcal{A}} - \vec{\mu})}{2} \right\}.$$

Since Σ is diagonal, the distribution functions are independent in coordinates. Hence the k -dimensional normal distribution function is given by the product of k 1-dimensional normal distribution functions.

$$f_{\vec{\mathcal{A}}}(x_1, \dots, x_k) = \prod_{1 \leq i \leq k} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp \left\{ -\frac{(x_i - \mu_i)^2}{2\sigma_i^2} \right\}. \quad (7.1)$$

Let o_i and o_j are two k -dimensional uncertain objects with attributes $\vec{\mathcal{A}}_i = (x_{i,1}, \dots, x_{i,k})^T$ and $\vec{\mathcal{A}}_j = (x_{j,1}, \dots, x_{j,k})^T$, means $\vec{\mu}_i = (\mu_{i,1}, \dots, \mu_{i,k})^T$ and $\vec{\mu}_j = (\mu_{j,1}, \dots, \mu_{j,k})^T$ and diagonal covariance matrices $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,k}^2)$ and $\Sigma_j = \text{diag}(\sigma_{j,1}^2, \dots, \sigma_{j,k}^2)$, respectively. Assuming that $\vec{\mathcal{A}}_i$ and $\vec{\mathcal{A}}_j$ are independent random vectors, then $\vec{\mathcal{A}}_i - \vec{\mathcal{A}}_j = \mathcal{N}(\vec{\mu}_i - \vec{\mu}_j, \Sigma_i + \Sigma_j)$ [114].

Since Σ_i and Σ_j are diagonal matrices, the k -dimensional normal difference distribution function can be given by the product of k 1-dimensional normal distribution functions as follows.

$$f_{\vec{\mathcal{A}}_i - \vec{\mathcal{A}}_j}(x_1, \dots, x_k) = \prod_{1 \leq m \leq k} \frac{1}{\sqrt{2\pi(\sigma_{i,m}^2 + \sigma_{j,m}^2)}} \exp\left\{-\frac{(x_m - (\mu_{i,m} - \mu_{j,m}))^2}{2(\sigma_{i,m}^2 + \sigma_{j,m}^2)}\right\}. \quad (7.2)$$

Since this lemma focus on 2-dimensional $Pr(o_i, o_j, D)$. The normal difference distribution of 2-dimensional uncertain objects o_i and o_j is given by,

$$f_{\vec{\mathcal{A}}_i - \vec{\mathcal{A}}_j}(x_1, x_2) = \frac{1}{2\pi\sqrt{(\sigma_{i,1}^2 + \sigma_{j,1}^2)(\sigma_{i,2}^2 + \sigma_{j,2}^2)}} \times \exp\left\{-\left(\frac{(x_1 - \alpha_1)^2}{2(\sigma_{i,1}^2 + \sigma_{j,1}^2)} + \frac{(x_2 - \alpha_2)^2}{2(\sigma_{i,2}^2 + \sigma_{j,2}^2)}\right)\right\}, \quad (7.3)$$

where $\alpha_1 = \mu_{i,1} - \mu_{j,1}$ and $\alpha_2 = \mu_{i,2} - \mu_{j,2}$ are the differences between the means of objects o_i and o_j respectively. Hence the probability that $o_j \in DN(o_i)$ denoted by $Pr(o_i, o_j, D)$, is given as follows.

$$Pr(o_i, o_j, D) = \frac{1}{2\pi\sqrt{(\sigma_{i,1}^2 + \sigma_{j,1}^2)(\sigma_{i,2}^2 + \sigma_{j,2}^2)}} \times \int_0^D \int_0^{2\pi} \exp\left\{-\left(\frac{(r \cos \theta - \alpha_1)^2}{2(\sigma_{i,1}^2 + \sigma_{j,1}^2)} + \frac{(r \sin \theta - \alpha_2)^2}{2(\sigma_{i,2}^2 + \sigma_{j,2}^2)}\right)\right\} r \, d\theta \, dr \quad \blacksquare \quad (7.4)$$

B: Transformation of a Correlated d -dimensional Gaussian Distribution into an Uncorrelated d -dimensional Gaussian Distribution

This appendix shows that a correlated d -dimensional Gaussian distribution can be transformed into an uncorrelated d -dimensional Gaussian distribution with appropriate coordinate transformation (Principal Component Analysis [100]). Hence resulting in a diagonal, uncorrelated covariance matrix whose variance is uniform in all dimensions, which is consistent with the proposed distance probability function (Eq. 4.3) in this dissertation, for the computation of outlier score ($\#D$ -neighbors).

Assuming that we have two-dimensional Gaussian distribution with attribute

vector $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$, mean vector $\boldsymbol{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$ and covariance matrix $C_{\mathbf{x}} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$.

The covariance matrix $C_{\mathbf{x}}$ is a symmetric, positive-definite and real-valued square matrix. Therefore $C_{\mathbf{x}}$ has real-valued eigenvalues which are non-zero and are orthogonal. Hence real-valued eigenvectors may be defined by $C_{\mathbf{x}}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}$. If any eigenvalues of the covariance matrix are identical, orthogonal eigenvectors may still be obtained, however they are not unique. Choosing the eigenvectors to have unit magnitude, they may be put together into an eigenvector matrix $\Phi = [\phi_1, \dots, \phi_n]$ such that $C_{\mathbf{x}}\Phi = \Phi\Lambda$ where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ and the eigenvector matrix is orthonormal, $\Phi^{-1} = \Phi^t$.

In order to transform a correlated d -dimensional Gaussian Distribution into an Uncorrelated d -dimensional Gaussian Distribution, a series of coordinate transformations is required which are as follows.

1. Shift the mean to the coordinate origin $\mathbf{x}' = \mathbf{x} - \boldsymbol{\mu}$. According to the definition, the mean of the shifted set is zero, $E\{\mathbf{x}'\} = E\{\mathbf{x} - \boldsymbol{\mu}\} = E\{\mathbf{x}\} - \boldsymbol{\mu} = 0$, where $E\{\}$ denotes the expectation operator, thus this transformation of coordinate has no effect on the covariance, $C_{\mathbf{x}'} = E\{(\mathbf{x}')(\mathbf{x}')^t\} = E\{(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^t\} = C_{\mathbf{x}}$.
2. After the shifting of mean, consider the following transformation $\mathbf{x}'' = \Phi^t \mathbf{x}'$. This results in a diagonal covariance matrix which is equal to rotating the coordinate axes: $C_{\mathbf{x}''} = E\{(\mathbf{x}'')(\mathbf{x}'')^t\} = E\{(\Phi^t \mathbf{x}')(\Phi^t \mathbf{x}')^t\} = \Phi^t E\{(\mathbf{x}')(\mathbf{x}')^t\} \Phi$, hence $C_{\mathbf{x}''} = \Phi^t C_{\mathbf{x}} \Phi = \Phi^{-1} C_{\mathbf{x}} \Phi = \Lambda$.
3. Finally, consider the following transformation, $\mathbf{x}''' = S^{-1} \mathbf{x}''$, where $S = \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n})$. This results in a unity covariance matrix and is equal to re-scaling the coordinate axes: $C_{\mathbf{x}'''} = E\{(\mathbf{x}''')(\mathbf{x}''')^t\} = E\{(S^{-1} \mathbf{x}'')(S^{-1} \mathbf{x}'')^t\} = S^{-1} E\{(\mathbf{x}'')(\mathbf{x}'')^t\} S^{-t}$, hence $C_{\mathbf{x}'''} = S^{-1} \Lambda S^{-t} = \mathbf{I}$, i.e.,

$$C_{\mathbf{x}'''} = \begin{bmatrix} \frac{1}{\sqrt{\lambda_1}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\lambda_n}} \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\lambda_1}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\lambda_n}} \end{bmatrix} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$$

With this series of transformations it is always possible to define alternative

coordinates for an uncertain object, with a diagonal, uncorrelated covariance matrix whose variance is uniform in all dimensions [100, 115].■

C: Approximate Values of d -dimensional Probability Density Function for the Difference between Two Random Variables following the d -dimensional Gaussian Distributions

According to the well known three sigma rule or the empirical rule of the Gaussian distribution, approximately 68.27% of the values lie within one standard deviation of the mean. Similarly, approximately 95.45% of the values lie within two standard deviations of the mean. Nearly all (99.73%) of the values lie within three standard deviations of the mean [88]. Since the difference between two Gaussian distributions follows the Gaussian distribution [114], the three sigma rule is applicable to the Gaussian difference distribution and to our proposed probability distribution function which is based on the Gaussian difference distribution, $Pr(o_i, o_j, D)$, as long as $D \geq 3\sigma$, where σ is the standard deviation of the Gaussian difference distribution used in $Pr(o_i, o_j, D)$. However, the above mentioned values of the three sigma rule is only valid for the 1-dimensional Gaussian distribution.

The number of standard deviations s needed to enclose a given percentage of values for a d -dimensional random variable X following the Gaussian distribution can be obtained using the expression $Pr\{d_M(X, \mu) \leq s\} = G_d(s^2)$ [19], where $d_M(X, \mu) = \sqrt{(X - \mu)^T \Sigma^{-1}(X - \mu)}$ is the Mahalanobis distance and $G_d(s^2)$ is the CDF of the chi-squared distribution with d -degrees of freedom. In this dissertation we assume that there exists no correlation between the attributes of uncertain objects. Appendix B shows that the series of transformations is always possible to find alternative coordinates of an object, which eliminates the correlation among an object's coordinates.

Hence if t denotes the value of s , such that $Pr\{d_M(X, \mu) \leq t\}$ covers a large area of the Gaussian distribution (say $> 99\%$), then for $\alpha \leq D - t\sigma'$, $Pr(o_i, o_j, D) \approx 1$ and for $\alpha \geq D + t\sigma'$, $Pr(o_i, o_j, D) \approx 0$.■

Bibliography

- [1] Cisl research data archive. <http://rda.ucar.edu>, 2012. [Online; accessed 06-July-2012].
- [2] International surface pressure databank (ispdv2) 17682010. <http://rda.ucar.edu>, 2012. [Online; accessed 06-July-2012].
- [3] Sloan digital sky survey. <http://www.sdss.org>, 2012. [Online; accessed 06-July-2012].
- [4] Intel berkeley research lab sensor data. <https://www.intel-university-collaboration.net/>, 2013. [Online; accessed 31-October-2013].
- [5] Met office weather data. <http://data.gov.uk/data>, 2013. [Online; accessed 03-September-2013].
- [6] Stevens water monitoring systems, inc. <http://www.stevenswater.com/>, 2013. [Online; accessed 07-March-2013].
- [7] Vaisala corporation. <http://www.vaisala.com/>, 2013. [Online; accessed 07-March-2013].
- [8] Xylem corporation. <http://www.globalw.com/>, 2013. [Online; accessed 07-March-2013].
- [9] Techtarget: Intrusion detection definition. <http://techtarget.com/>, 2014. [Online; accessed 05-January-2014].
- [10] Charu C. Aggarwal. *Managing and Mining Uncertain Data*. Springer Publishing Company, Incorporated, 2009.

-
- [11] Charu C. Aggarwal. *Outlier Analysis*. Springer-Verlag New-York, 2013.
- [12] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 37–46, 2001.
- [13] Charu C. Aggarwal and Philip S. Yu. Outlier detection with uncertain data. In *SIAM International Conference on Data Mining (SDM)*, 2008.
- [14] Noor Alaydie, Farshad Fotouhi, Chandan K. Reddy, and Hamid Soltanian-Zadeh. Noise and outlier filtering in heterogeneous medical data sources. In *DEXA Workshop on Database and Expert Systems Applications*, 2010.
- [15] E. Aleskerov, B. Freisleben, and B. Rao. Cardwatch: A neural network based database mining system for credit card fraud detection. In *In Proceedings of the IEEE Conference on Computational Intelligence for Financial Engineering*, pages 220–226, 1997.
- [16] Fabrizio Angiulli and Fabio Fassetti. Detecting distance-based outliers in streams of data. In *ACM 16th Conference on Information and Knowledge Management (CIKM)*, pages 811–820, 2007.
- [17] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *Principles of Data Mining and Knowledge Discovery*, pages 15–27. 2002.
- [18] F. J. Anscombe and Irwin Guttman. Rejection of outliers. *Technometrics*, 2(2):123147, 1960.
- [19] Peter Bajorski. *Statistics for Imaging, Optics, and Photonics*. Wiley Series in Probability and Statistics, 2012.
- [20] Vic Barnett and Toby Lewis. *Outliers in statistical data*. 1994.
- [21] Sugato Basu, Mikhail Bilenko, and Raymond J. MooneyBasu. A probabilistic framework for semi-supervised clustering. In *In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 5968, 2004.

- [22] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbors meaningful? In *Proceedings of the International Conference on Database Theory*, pages 217–235, 1999.
- [23] Richard J. Bolton and David J. Hand. Unsupervised profiling methods for fraud detection. In *In Proceedings of the Conference on Credit Scoring and Credit Control VII*, 1999.
- [24] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *In Proceedings of the 8th SIAM International Conference on Data Mining*, page 243254, 2008.
- [25] Joel Branch, Boleslaw Szymanski, Chris Giannella, Ran Wolff, and Hillol Kargupta. In-network outlier detection in wireless sensor networks. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2006.
- [26] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. Optics-of: Identifying local outliers. In *In Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, page 262270, 1999.
- [27] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. Lof: Identifying density-based local outliers. In *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, page 93104, 2000.
- [28] M.R. Brito, E.L. Chfivez, A.J. Quiroz, and J.E. Yukich. Connectivity of the mutual k-nearestneighbor graph in clustering and outlier detection. *Statistics and Probability Letters*, 35(1):3342, 1997.
- [29] Vladimir Bychkovski, Seapahn Megerian, Deborah Estrin, and Miodrag Potkonjak. A collaborative approach to in-place sensor calibration. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks*, pages 301–316, 2003.
- [30] Simon Byers and Adrian E. Raftery. Nearest neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, page 577584, 1998.

- [31] Keyan Cao, Donghong Han, Guoren Wang, Yachao Hu, and Ye Yuan. An algorithm for outlier detection on uncertain data stream. In *In Proceedings of the 15th Asia-Pacific Web Conference (APWeb)*, pages 449–460, 2013.
- [32] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1 – 15:58, 2009.
- [33] Varun Chandola, Shyam Boriah, and Vipin Kumar. Understanding categorical similarity measures for outlier detection. *Technical report, University of Minnesota*, 2008.
- [34] V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris. Hierarchical anomaly detection in distributed large-scale sensor networks. In *In Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC)*, page 761767, 2006.
- [35] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 551–562, 2003.
- [36] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. volume 16, pages 523–544. Springer-Verlag New York, Inc., 2007.
- [37] M. J. Desforges, P. J. Jacob, and J. E. Cooper. Applications of probability density estimation to the detection of abnormal conditions in engineering. In *In Proceedings of the Institute of the Mechanical Engineers.*, volume 212, page 687703, 1998.
- [38] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *In Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, 2004.
- [39] Yanlei Diao, Boduo Li, Anna Liu, Liping Peng, Charles Sutton, Thanh Tran, and Michael Zink. Capturing data uncertainty in high-volume

- stream processing. In *In Proceedings of 4th Biennial Conference on Data Systems (CIDR)*, 2009.
- [40] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. 2nd Ed. Wiley-Interscience, 2000.
- [41] Arturo Elias, Alberto Ochoa-Zezzatti, Alejandro Padilla, and Julio Ponce. Outlier analysis for plastic card fraud detection a hybridized and multi-objective approach. In *Hybrid Artificial Intelligent Systems*, pages 1–9. 2011.
- [42] Levent Ertloz, Michael Steinbach, and Vipin Kumar. Finding topics in collections of documents: A shared nearest neighbor approach. In *Clustering and Information Retrieval*, page 83104, 2003.
- [43] Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the 17th International Conference on Machine Learning*, page 255262, 2000.
- [44] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. Geometric framework for unsupervised anomaly detection. In *In Proceedings of the Conference on Applications of Data Mining in Computer Security*, page 78100, 2002.
- [45] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, page 226231, 1996.
- [46] Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. In *In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, 1999.
- [47] Norbert Fuhr and Thomas Rolleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 1997.
- [48] Edgeworth F.Y. Xli. on discordant observations. *Philosophical Magazine Series 5*, 23(143):364–375, 1887.

- [49] Hugo Garces and Daniel Sbarbaro. Outliers detection in environmental monitoring databases. *Engineering Applications of Artificial Intelligence*, 24(2):341 – 349, 2011.
- [50] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information Systems Journal*, 25(5):345–366, 2000.
- [51] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. In *In Proceedings of the 17th International Conference on Pattern Recognition*, page 430433, 2004.
- [52] D.M. Hawkins. Identification of outliers. Monographs on Applied Probability and Statistics, pages 1–12. 1980.
- [53] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):16411650, 2003.
- [54] Irja Helm, Lauri Jalukse, and Ivo Leito. Measurement uncertainty estimation in amperometric sensors, a tutorial review. *Sensors*, 10(5):4430–4455, 2010.
- [55] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.
- [56] Peter Huber and Elvezio Ronchetti. *Robust Statistics*. Wiley, New York, 1974.
- [57] Ihab Ilyas, George Beskales, and Mohamed Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4):11:1–11:58, 2008.
- [58] Kozue Ishida and Hiroyuki Kitagawa. Detecting current outliers: Continuous outlier detection over time-series data streams. In *In proceedings of the 19th International Conference on Database and Expert Systems (DEXA)*, 2008.
- [59] Anil Jain and Richard Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1988.

- [60] Bin Jiang and Jian Pei. Outlier detection on uncertain data: Objects, instances, and inferences. In *IEEE 27th International Conference on Data Engineering (ICDE)*, pages 422–433. IEEE, 2011.
- [61] M.F. Jiang, S.S. Tseng, and C.M. Su. Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, 22(6-7), 2001.
- [62] George H. John. Robust decision trees: Removing outliers from databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, page 174179, 1995.
- [63] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *In Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB)*, 1998.
- [64] Edwin M. Knorr and Raymond T. Ng. Finding intensional knowledge of distance-based outliers. In *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB*, pages 211–222, 1999.
- [65] Edwin M. Knorr, Raymond T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal*, 8(3-4):237–253, 2000.
- [66] Maria Kontaki, Anastasios Gounaris, Apostolos N. Papadopoulos, Kostas Tsichlas, and Yannis Manolopoulos. Continuous monitoring of distance-based outliers over data streams. In *In Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE)*, 2011.
- [67] Yufeng Kou, Chang-Tien Lu, and Dechang Chen. Spatial weighted outlier detection. In *In Proceedings of the SIAM Conference on Data Mining*, 2006.
- [68] Vipin Kumar. Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems Online*, 6(10):1–9, 2005.
- [69] Weining Qian Li Wei, Aoying Zhou, Wen Jin, and Jeffrey X. Yu. Hot: Hypergraph-based outlier test for categorical data. In *In Proceedings of the 7th Pacific-Asia Conference on Knowledge and Data Discovery*, page 399410, 2003.

- [70] Jessica Lin, Eamonn Keogh, Ada Fu, and Helga Van Herle. Approximations to magic: Finding unusual medical time series. In *In Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, pages 329–334, 2005.
- [71] Matthew V. Mahoney and Philip K. Chan. Learning rules for anomaly detection of hostile network traffic. In *In Proceedings of the 3rd IEEE International Conference on Data Mining*, 2003.
- [72] Matthew V. Mahoney and Philip K. Chan. Learning rules for anomaly detection of hostile network traffic. In *In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, 2003.
- [73] Oded Maimon and Lior Rokach. *Data Mining and Knowledge Discovery Handbook, 2nd ed.* Springer, 2010.
- [74] Graeme Manson, Gareth Pierce, and Keith Worden. On the long-term stability of normal condition for damage detection in a composite panel. *Key Engineering Materials*, 204(1):359–369, 2001.
- [75] Markos Markou and Sameer Singh. Novelty detection: A review part 1: Neural network based approaches. *Signal Processing*, 83(12):2481 – 2497, 2003.
- [76] Markos Markou and Sameer Singh. Novelty detection: A review part 2: Neural network based approaches. *Signal Processing*, 83(12):2499 – 2521, 2003.
- [77] Takazumi Matsumoto and Edward Hung. Accelerating outlier detection with uncertain data using graphics processors. In *In the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 169–180, 2012.
- [78] Wang Kay Ngai, Ben Kao, Chun-Kit Chui, M. Chau R. Cheng, and K.Y. Yip. Efficient clustering of uncertain data. In *In Proceedings of the 6th International Conference on Data Mining, (ICDM)*, pages 436–445, 2006.
- [79] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and

- Mani Srivastava. Sensor network data fault types. *ACM Transactions on Sensor Networks*, 5(3):25:1–25:29, 2009.
- [80] Jurg Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1):38–71, 1984.
- [81] Ashrar Omer, Johnson Thomas, and Ling Zhu. Mutual authentication protocols for rfid systems. *International Journal of Automation and Computing*, 5(4):348–365, 2008.
- [82] Gustavo H. Orair, Carlos H. C. Teixeira, Wagner Meira Jr., and Ye Wang and Srinivasan Parthasarathy. Distance-based outlier detection: Consolidation and renewed bearing. *Proceedings of the VLDB Endowment*, 3(1-2):1469–1480, 2010.
- [83] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, 12(2-3):203–228, 2006.
- [84] Themistoklis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Distributed deviation detection in sensor networks. *ACM SIGMOD Records*, 32(4):77–82, 2003.
- [85] Girish Keshav Palshikar. Distance-based outliers in sequences. *Lecture Notes in Computer Science*, 3816:547–552, 2005.
- [86] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. *IEEE 29th International Conference on Data Engineering (ICDE)*, pages 315–326, 2003.
- [87] A.M. Pires and C.M. Santos-Pereira. Using clustering and robust estimators to detect outliers in multivariate data. In *In Proceedings of the International Conference on Robust Statistics*, 2005.
- [88] Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

- [89] Nithya Ramanathan, Laura Balzano, Marci Burt, Deborah Estrin, Tom Harmon, Charlie Harvey, Jenny Jay, Eddie Kohler, Sarah Rothenberg, and Mani Srivastava. Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks. *CENS Tech Report #62*, 2006.
- [90] Nithya Ramanathan, Laura Balzano, Marci Burt, Deborah Estrin, Tom Harmon, Charlie Harvey, Jenny Jay, Eddie Kohler, Sarah Rothenberg, and Mani Srivastava. Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks. *CENS Tech Report #68*, 2006.
- [91] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large datasets. *ACM SIGMOD Record*, 29(2), 2000.
- [92] Peter J. Rousseeuw and Annick M. Leroy. Robust regression and outlier detection. *John Wiley and Sons, Inc.*, 1987.
- [93] Anish Das Sarma, Omar Benjelloun, Alon Halevy, and Jennifer Widom. Working models for uncertain data. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pages 7–7, April 2006.
- [94] Rob Saunders and John S. Gero. The importance of being emergent. In *In Proceedings of the Conference on Artificial Intelligence in Design*, 2000.
- [95] Guttormsson S.E., Marks R.J., El-Sharkawi M.A., and Kerszenbaum I. Elliptical novelty grouping for online short-turn detection of excited running rotors. *IEEE Transactions on Energy Conversion*, 14(1), 1999.
- [96] Salman Ahmed Shaikh and Hiroyuki Kitagawa. Efficient distance-based outlier detection on uncertain datasets of gaussian distribution. *World Wide Web*, pages 1–28, 2013.
- [97] Salman Ahmed Shaikh and Hiroyuki Kitagawa. Top- k outlier detection from uncertain data. *International Journal of Automation and Computing (IJAC)*, 2014.

- [98] Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Transactions on Sensor Networks*, 6(3):23:1–23:39, 2010.
- [99] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *In Proceedings of the 24rd International Conference on Very Large Databases*, page 428439, 1998.
- [100] Lindsay I Smith. A tutorial on principal components analysis. In *Student Tutorials, University of Otago, New Zealand*, 2002.
- [101] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering approaches for anomaly-based intrusion detection. In *In Proceedings of the Intelligent Engineering Systems through Artificial Neural Networks*, page 579584, 2002.
- [102] Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. Top-k query processing in uncertain databases. In *In proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 345–360, 2007.
- [103] Clay Spence, Lucas Parra, and Paul Sajda. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *In Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis. IEEE Computer Society*, pages 3–10, 2001.
- [104] A. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *In Proceedings of the IEEE Aerospace Conference*, page 38533862, 2005.
- [105] Robert Szewczyk, Joseph Polastre, Alan M. Mainwaring, and David E. Culler. Lessons from a sensor network expedition. In *Wireless Sensor Networks*, pages 307–322, 2004.
- [106] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

- [107] Jian Tang, Zhixiang Chen, Ada Wai chee Fu, and David W. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, page 535548, 2002.
- [108] Yufei Tao, Xiaokui Xiao, and Reynold Cheng. Range search on multidimensional uncertain data. *ACM Transactions on Database Systems*, 32(3), 2007.
- [109] Henry S. Teng, Kaihu Chen, and Stephen C. Y. Lu. Adaptive real-time anomaly detection using inductively generated sequential patterns. In *IEEE Symposium on Security and Privacy*, pages 278–284, 1990.
- [110] William Thistleton, John A. Marsh, Kenric Nelson, and Constantino Tsallis. Generalized box-muller method for generating q-gaussian random deviates. *IEEE Transactions on Information Theory*, 53(12):4805–4810, 2007.
- [111] Phoha Vir. *The Springer Internet Security Dictionary*. Springer-Verlag, 2002.
- [112] Bin Wang, Gang Xiao, Hao Yu, and Xiaochun Yang. Distance-based outlier detection on uncertain data. In *Proceedings of the 9th IEEE International Conference on Computer and Information Technology (CIT)*, 2009.
- [113] Bin Wang, Xiao-Chun Yang, Guo-Ren Wang, and Ge Yu. Outlier detection over sliding windows for probabilistic data streams. *Journal of Computer Science and Technology*, 25(3):389–400, 2010.
- [114] Eric W. Weisstein. Normal difference distribution. from mathworld—a wolfram web resource. <http://mathworld.wolfram.com/>, 2013. [Online; accessed 03-September-2013].
- [115] Eric W. Weisstein. Matrix diagonalization. from mathworld—a wolfram web resource. <http://mathworld.wolfram.com/>, 2014. [Online; accessed 01-January-2014].

-
- [116] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *In Proceedings of the 20th International Conference on Machine Learning. AAAI*, page 808815, 2003.
- [117] Dantong Yu, Gholamhosein Sheikholeslamiy, and Aidong Zhang. Find-out: Finding outliers in very large datasets. *Knowledge and Information Systems*, 4(4):387412, 2002.
- [118] Ji Zhang, Meng Lou, and Hai H. Wang Tok Wang Lin and. Hos-miner: A system for detecting outlying subspaces of digh-dimensional data. In *In Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, page 12651268, 2004.
- [119] Ji Zhang and Hai Wang. Detecting outlying subspaces for high-dimensional data: The new task, algorithms, and performance. *Knowledge and Information Systems*, 10(3):333–355, 2006.
- [120] Cui Zhu, Hiroyuki Kitagawa, Spiros Papadimitriou, and Christos Faloutsos. Outlier detection by example. *Journal of Intelligent Information Systems*, 36(2):217–247, 2011.
- [121] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. 5(5):363–387, 2012.

List of Publications

Refereed Journal Papers

1. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Top- k Outlier Detection from Uncertain Data”. *International Journal of Automation and Computing (IJAC)*, Springer. (to appear)
2. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Efficient distance-based outlier detection on uncertain datasets of Gaussian distribution”. *World Wide Web (WWW)*, Springer, April 2013. (published online)

Refereed Conference Papers

1. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Continuous Outlier Detection on Uncertain Data Streams”. In *IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. Singapore, April 21-24, 2014. (to appear)
2. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Fast Top- k Distance-based Outlier Detection on Uncertain Data”. In *14th International Conference on Web-Age Information Management (WAIM)*, pages 301-313. Beidaihe, China, June 14-16, 2013.
3. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Distance-based Outlier Detection on Uncertain Data of Gaussian Distribution”. In *14th Asia-Pacific Web Conference (APWeb)*, pages 109-121. Kunming, China, April 11-13, 2012.

Domestic Conference Papers

1. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Differential Outlier Detection on Uncertain Streams of the Gaussian Distribution”. In *5th International Workshop with Mentors on Databases, Web and Information Management for Young Researchers (iDB2013)*. Sapporo, Japan, July 21-23, 2013.
2. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Top- k Distance-based Outlier Detection on Uncertain Dataset”. In *5th Forum on Data Engineering and Information Management (DEIM 2013)*. Koriyama, Japan, March 3-5, 2013.
3. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Distance-based Outlier Detection on Uncertain Data of Bounded Gaussian Distribution”. In *4th International Workshop with Mentors on Databases, Web and Information Management for Young Researchers (iDB2012)*. Nagoya, Japan, July 31-August 1, 2012.
4. Salman Ahmed Shaikh and Hiroyuki Kitagawa. “Outlier Detection on Uncertain Data of Gaussian Distribution”. In *4th Forum on Data Engineering and Information Management (DEIM 2012)*. Kobe, Japan, March 3-5, 2012.