MULTI-LEVEL RELATIONSHIP OUTLIER DETECTION

by

Qiang Jiang

B.Eng., East China Normal University, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2012

APPROVAL

Name: Qiang Jiang

Degree: Master of Science

Title of Thesis: Multi-Level Relationship Outlier Detection

Examining Committee: Dr. Binay Bhattacharya
Chair

Dr. Jian Pei, Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Wo-Shun Luk, Professor, Computing Science
Simon Fraser University
Supervisor

Dr. Jiangchuan Liu, Professor, Computing Science
Simon Fraser University
SFU Examiner

Date Approved: 24 July 2012

Abstract

Relationship management is critical in business. Particularly, it is important to detect abnormal relationships, such as fraudulent relationships between service providers and consumers. Surprisingly, in the literature there is no systematic study on detecting relationship outliers. Particularly, no existing methods can detect and handle relationship outliers between groups and individuals in groups. In this thesis, we tackle this important problem by developing a simple yet effective model. We identify two types of outliers and devise efficient detection algorithms. Our experiments on both real data sets and synthetic ones confirm the effectiveness and efficiency of our approach.

To my parents.

"Learn from yesterday, live for today, hope for tomorrow."

— Albert Einstein *(1879 - 1955)*

Acknowledgments

I would like to express my deepest gratitude to my senior supervisor, Dr. Jian Pei, who provides creative ideas for my research and warm encouragement for my life. Throughout my master study, he shared with me not only valuable knowledge and research skills but also the wisdom of life.

My gratitude also goes to my supervisor, Dr. Wo-Shun Luk, for reviewing my work and helpful suggestions that helped me to improve my thesis. I am grateful to thank Dr. Jiangchuan Liu and Dr. Binay Bhattacharya, for serving in my examing committee.

Many thanks to my friends, Guanting Tang, Xiao Meng, Da Huang, Guangtong Zhou and Hossein Maserrat, for their kind help during my study at SFU.

My sincerest gratitude goes to my parents. Their endless love supports me to overcome all the difficulties in my study and life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Relationship management is among the most important aspects in business today. Many business intelligence tools have been developed to automate and improve relationship management [3] [34] [11], such as relationships among business customers, employees, service providers, and channel partners. Some example applications are Customer Relationship Management (CRM), Supply Chain Management (SCM), Human Resource Management (HRM), and Enterprise Resource Planning (ERP).

Data analysis for relationship management has been of great interest in research and development. For example, clustering analysis on relationships, which has been extensively studied [10] [39] [25], identifies and groups similar relationships between parties. Such similar relationships can be used in, for instance, marketing campaigns. In recommender systems, if the relationship between a customer c and a company is similar to those between a group of customers C and the same company, then some popular products or services C purchased may be recommended to c.

In this thesis, we focus on an interesting task of relationship analysis, that is, *relationship outlier detection*. The task is to identify abnormal relationships between parties.

Relationship outliers are interesting and useful in many applications. For example, in a medical insurance company, the relationship between a patient and a medical service provider is interesting to the business fraud detection unit if the patient visits the service provider much more frequently than normal, or the provider often charges the patient at a much higher rate than normal. Detecting such outlying relationships is particularly valuable in business.

Although outlier detection has been studied extensively [23] [31] [6], most of the existing work focuses on mining outliers from a set of objects. Surprisingly, to the best of our knowledge, there does not exist any systematic study on mining outliers about relationships. One may wonder whether we can straightforwardly model a relationship as an object and thus simply apply the existing outlier detection techniques to tackle the relationship outlier detection problem. One critical challenge here is that relationships often form hierarchies. Therefore, relationship outlier detection has to be conducted correspondingly in a multi-level, hierarchical manner.

Example 1 (Motivation)**.** *As a concrete motivation example, consider the scenario in the medical insurance company example we just discussed. We are interested in the relationships between patients and medical service providers. The patients may form groups and a hierarchy according to their genders, locations, and ethnic groups. The medical service providers may also form groups according to their qualifications, experience, and operation locations. The relationships are not only between individual patients and service providers, but also can exist between patient groups and provider groups at different levels. How to detect outliers at different levels and analyze them is a sophisticated problem.*

For example, suppose on average each patient visits a therapist 4 *times a month, which is regarded as a normal relationship. A patient visiting a therapist* 8 *times a month may not look too abnormal a relationship, since some patients do need more frequent treatments. However, a group of patients living in the same address each visiting a group of therapists operating in the same location* 8 *times a month may look suspicious, since it is unlikely everyone in a family needs the same pattern of treatment. Some service provider at that location may be involved in some fraudulent transactions. Detecting such relationship outliers at multiple-levels is very interesting and practically useful, but, at the same time, technically far from trivial.* ■

Detecting relationship outliers is a general problem with many business applications. For example, finance trading between an agent and a customer can be modeled as a relationship between the two parties. Detecting relationship outliers from such relationship data can help to identify possible frauds in trading and discover customers who need special attention. More generally, the relationship can encompass more than two parties. For example, customers credit card purchases can be modeled as a relationship crossing three parties: customers, banks, and vendors. Detecting outliers from such relationships in a multi-level and hierarchical way may help to detect frauds and organized crimes.

In this thesis, we tackle the problem of relationship outlier detection, and make the following contributions.

- We deal with the multi-level relationships where attributes form hierarchies.
- We develop a simple yet effective model, and identify two types of outliers.
- Based on the well adopted statistical model-based outlier detection theme, we devise efficient algorithms.
- We conduct extensive experiments on both real data sets and synthetic data sets. Our results verify the effectiveness and the efficiency of our approach.

The rest of the thesis is organized as follows. In Section 2, we review the related work on outlier detection and data warehouse. In Section 3, we present our model on multi-level relationship outliers. We develop the detection methods in Section 4, and report our experimental results in Section 5. We conclude the thesis in Section 6.

Chapter 2

Related Work

This thesis is related to existing technical work on outlier detection and data warehouse, especially, data cube computation.

2.1 Outlier Detection

"An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [21]. Although there is seemingly no universally accepted definition, the widely cited characterization of outliers is based on a statistical intuition that objects are generated by a stochastic process (a generative model), as such, objects that fall in the regions of high probability for the stochastic model are normal whereas those in the regions of low probability are outliers [20]. Outliers can be classified into three categories: global, contextual, and collective outliers.

There are a few excellent surveys on outlier detection techniques [1] [2] [7] [24]. Here, we review the major methods briefly.

2.1.1 Detection Methods

Outlier detection is the process of finding objects that exhibit behaviors that are very different from the expectation. There are many outlier detection methods in the literature and in practice. Outlier detection methods can be categorized according to whether or not prior knowledge is available to model both normality and abnormality. Prior knowledge typically consists of samples that are tagged as normal or abnormal by subject matter experts. If the prior knowledge is available, detection approach is analogous to supervised classification. If, on the other hand, there is no prior knowledge, the detection approach is effectively a learning scheme analogous to unsupervised clustering [23].

The Knowledge Based Approach

If samples labeled by domain experts can be obtained, they can be used to build outlier detection models. The methods using domain knowledge and labeled samples are called the knowledge based approaches and can be divided into *supervised* and *semi-supervised methods*.

Supervised methods model data normality and abnormality based on the samples labeled by domain experts. Outlier detection can then be modeled as a classification problem to learn a classifier that can recognize outliers. Classification algorithms require a decent spread of both normal and abnormal data; however, in reality, the population of outliers is much smaller than that of normal objects. Thus, additional considerations for handling imbalanced data must be taken into account and techniques, such as oversampling or artificial outliers have been devised [28] [29] [8] [35] [41] [40].

Semi-supervised outlier detection methods [18] [12] can be regarded as applications of a semi-supervised learning approach where the normal class is taught but the algorithm learns to recognize abnormality. For example, when only a small number of normal samples are available, unlabeled objects that are similar to normal objects can be added to supplement and train a model for normal objects. Or, the model can be incrementally tuned as more labeled normal objects become available. The model of normal objects can then be used to detect outliers. It is suitable for static or dynamic data as it only learns one class (*i.e.* the normal class) that provides the model of normality. As such, it aims to define a boundary of normality.

The Assumption Based Approach

In this category, the outlier detection methods make certain assumptions about outliers against the rest of the data. According to the assumptions made, the methods can be categorized into three types: *statistical methods*, *proximity-based methods*, and *clustering-based methods*.

- The general idea behind statistical methods [5] [33] for outlier detection is to learn a generative model fitting the given data set, and then identify those objects in low-probability regions of the model as outliers. The effectiveness of statistical methods depends highly on whether or not the assumptions made for the statistical model hold true for the given data. Statistics based methods typically operate in two phases: the training phase in which the model (*i.e.* distribution parameters) is estimated, and the testing phase where a test instance is compared to the model to determine if it is an outlier or not.

The statistical methods can be divided into two categories: *parametric* and *nonparametric* methods, according to how the models are specified and learned. Parametric methods assume that the normal data objects are generated by a parametric distribution with parameter, Θ. The probability density function of the parametric distribution, $f(x, \Theta)$, gives the probability that object x is generated by the distribution. The smaller the value, the more likely x is an outlier. Nonparametric methods do not assume any knowledge of data distribution; instead, they attempt to determine the distributions from the input data. One of the most widely used techniques is histogram analysis [17] [14] [13] where model estimation involves counting the frequency of occurrences of data instances; thereby estimating the probability of occurrence of a data instance.

- Proximity-based methods assume that an object is an outlier if the nearest neighbors of the object are far away in a feature space. In other words, if the proximity of an object to its neighbors significantly deviates from the proximity of most of the other objects to their neighbors in the same data set, then object is an outlier. The effectiveness of proximity-based methods relies heavily on the proximity (or distance) measure used. However, in some applications, such measures cannot be easily obtained. Furthermore, proximity-based methods often have difficulty in detecting a group of outliers if the outliers are close to one another. There are two types of proximity-based outlier detection methods; distance based methods [30] and density based methods [16].

- The basic premise of unsupervised outlier detection methods is that the normal objects probably are clustered and follow a pattern far more often than outliers. The idea is to find clusters first and objects that do not belong to any cluster are flagged as outliers. Clustering-based methods assume that the normal data objects belong to large and dense clusters while outliers belong to small or sparse clusters or do not belong to any cluster at all. However, objects that do not belong to clusters may be simply noise instead of outliers. Further, it is often costly to find clusters. Since outliers are assumed to occur far less frequently than normal objects, it does not make economic sense to undertake a laborious process of developing clusters for normal objects only to find a few outliers in the end. Recent unsupervised outlier detection methods [22] [15] [44] incorporate ideas to handle outliers without explicitly and completely finding clusters of normal objects.

Although there are extensive studies on outlier detection, to the best of our knowledge, there is no systematic exploration on outlying relationship detection, which is the topic of this thesis.

2.2 Data Warehouse

"A data warehouse is a subject-oriented, integrated, time-variant, and non-volatile collection of data organized in support of management decision making process" [26]. A data warehouse consolidates and generalizes data in a multidimensional space. The construction of a data warehouse involves data cleansing, integration, and transformation, hence, can be regarded as an important preprocessing step for data mining. Multidimensional data mining, also known as exploratory multidimensional data mining, or online analytical mining, integrates Online Analytical Processing (OLAP) with data mining to uncover knowledge in multidimensional databases.

2.2.1 Data Cubes

A data cube [19] allows data to be modeled and viewed in multiple dimensions; thus, is the fundamental constituent of multidimensional model for data warehouses. Each data cube consists of facts and dimensions. OLAP is an approach to swiftly answer multidimensional analytical queries run against data cubes. The primary means for ensuring optimal OLAP performance is through aggregations.

In data warehouse research literature, an n-dimensional data cube is often referred to as a lattice of cuboids. Given a set of dimensions, a cuboid for each possible subset of given dimensions can be generated, showing data at a different level of summarization, or SQL group-by (*i.e.* each can be represented by a cuboid). A collection of cuboids for all possible

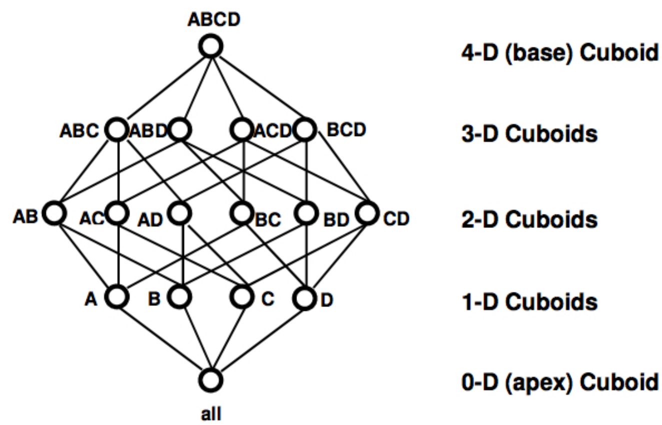


Figure 2.1: Lattice of Cuboids

dimension subsets form a lattice of cuboids. Figure 2.1 illustrates the lattice of cuboids, constituting a 4-D data cube for dimensions: A, B, C, and D.

The lattice that holds the lowest level of summarization (i.e. the most specific one) is referred to as the base cuboid while the lattice that holds the highest level of summarization (*i.e.* the least specific one) is referred to as the apex cuboid and it is typically denoted by "all". It is the grand total summary over all 4 dimensions and refers to the case where group-by is empty.

2.2.2 Cubing Methods

If the whole data cube is pre-computed (*i.e.* all cuboids are computed), queries run against the cube will return results very fast; however, the size of a data cube for n dimensions, $D_1, ..., D_n$ with cardinalities $|D_1|, ..., |D_n|$ is $\Pi_{i=1}^{n}(|D_i| + 1)$. The size increases exponentially with the number of dimensions, and the space requirement will quickly become unattainable. This problem is known as the **curse of dimensionality**. The goal of OLAP is to retrieve information in the most efficient manner possible. To this end, there are 3 options; no materialization (no pre-computation at all), full materialization (pre-compute everything), and partial materialization (somewhere between the two extremes).

Partial materialization represents an interesting trade-off between the response time and the storage space. Among the widely considered strategy is the **Iceberg Cubes** which contain only the cells that store aggregate values that are above some minimum support threshold, referred to as *min_sup*. Two major approaches, top-down and bottom-up, have been developed for efficient cube computation.

The Top-Down Approach

The Top-Down approach [45] [36] [42], represented by Multi-Way Array Cube, or *Multi-Way*, aggregates simultaneously on multiple dimensions. It uses a compressed sparse array structure to load the base cuboid and compute the cube. For efficient use of main memory, the array structure is partitioned into chunks, and by ordering the chunks and loading only the necessary ones into memory, it computes multiple cuboids simultaneously in one pass. The Top-Down approach cannot take advantage of the **Apriori** pruning because the iceberg condition can only be used after the whole cube is computed. This is due to the fact successive computation does not have the anti-monotonic property.

The Bottom-Up Approach

Bottom-Up Computation (BUC) [4] begins with apex cuboid toward the base cuboid. BUC is an algorithm to compute sparse and iceberg cubes. It counts the frequency of values of the first single dimension and then partitions the table based on the frequent value of the first dimension such that only those tuples that meet the minimum support are examined further. Unlike the Top-Down approach, the Bottom-Up approach can exploit Apriori pruning when computing Iceberg Cubes. The performace of BUC is sensitive to skew in the data and to the order of the dimensions.

Chapter 3

Multi-level Relationship Outliers

In this section, we formulate the problem of detecting multi-level relationship outliers. We first model multi-level hierarchical relationships. Then, we model and categorize relationship outliers.

3.1 Multi-level Relationships

Let us start with a simple example.

Example 2 (Fact table)**.** *Consider a table*

$$FACT = (\texttt{patient-id}, \texttt{gender}, \texttt{age-group}, \texttt{doctor-id}, \texttt{location}, \texttt{qualification}, \texttt{avg-charge}, \texttt{count}).$$

The table records the relationships between two parties: patients and medical doctors. The attributes in the table can be divided into three subsets.

- *The information about patients, including* `patient-id`*,* `gender`*, and* `age-group`*. While each patient, identified by her/his* `patient-id`*, is at the lowest granularity level, patients form groups and a hierarchy using attributes* `gender` *and* `age-group`*. The patient groups form a lattice, called the patient lattice, as shown in Figure 3.1.*

  In the figure, we use a symbol $*$ to indicate that an attribute is generalized. For example, $(gender, *)$ means the two groups formed by gender, that is, the group of female patients and the one of male patients.

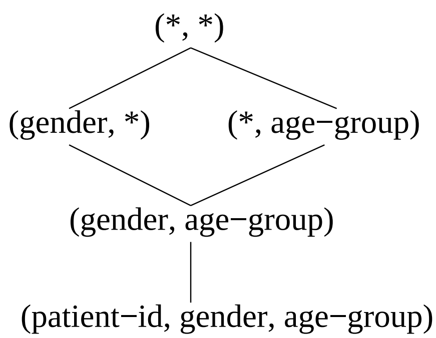


Figure 3.1: The patient lattice in Example 2.

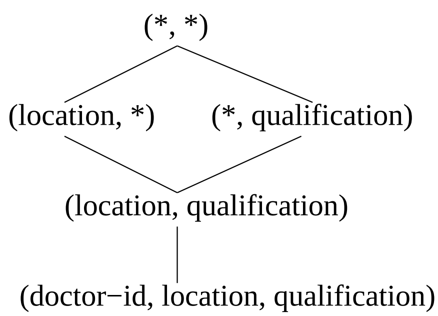


Figure 3.2: The doctor lattice in Example 2.

- *The information about doctors, including* `doctor-id`*,* `location`*, and* `qualification`*. Similar to the case of patients, individual doctors, identified by their* `doctor-ids`*, are at the lowest granularity level. Moreover, doctors form groups in a lattice hierarchy using attributes* `location` *and* `qualification`*, called the doctor lattice, as shown in Figure 3.2.*

- *The information about the relationship between a patient and a doctor, including* `avg-charge` *and* `count`*. This information can be aggregated from the corresponding transactions where patients and doctors are involved. The aggregate information reflects the nature of the relationships.*

We are interested in the relationships between patients and doctors. A relationship may involve a patient and a doctor, a group of patients and a doctor, a patient and a group of doctors, and a group of patients and a group of doctors. In general, a relationship involves an entry in the patient lattice and an entry in the doctor lattice. ∎

Let us formalize the ideas discussed in Example 2.

Definition 1 (Fact table). *In this thesis, we consider a* ***fact table*** F *about* k *parties, whose attributes can be partitioned into* $k+1$ *subsets* F_i $(1 \leq i \leq k+1)$. *That is,* $F = \cup_{i=1}^{k+1} F_i$.

- *The subset* $F_i (1 \leq i \leq k)$ *contains the information about the* i*-th party. We assume that a tuple of unique values on* F_i *corresponds to an entity of the* i*-th part in reality. In the rest of the thesis, we also call* F_i *the* ***party*** i.
- *The subset* F_{k+1} *contains the information about the relationship among the* k *instances from the* k *parties in a tuple, one from each. We call* F_{k+1} *the* ***measurement***.
- *We assume that* $\cup_{i=1}^{k} F_i$ *is a key of the table* F. *That is, no two tuples have the same values on all the attributes in* $\cup_{i=1}^{k} F_i$. *In other words, for a unique set of* k *instances from the* k *parties, one from each, there is only one tuple describing it.* ■

For the easy of presentation and understanding, in the rest of the thesis, we often confine our discussion to only 2 parties. However, our techniques can be straightforwardly extended to arbitrarily multiple parties.

We introduce the notion of groups as follows.

Definition 2 (Groups). *For the domain of each attribute in* $\cup_{i=1}^{k} F_i$, *we introduce a meta symbol* $*$, *which indicates that the attribute is generalized.*

Consider a tuple t *that takes values on the attributes in* F_i. *We say* t *to represent a* ***base level group*** *of party* i *if* t *takes a non-*$*$ *value on every attribute in* F_i. *Otherwise,* t *is called an* ***aggregate group*** *of party* i.

For groups t_1 *and* t_2 *such that* $t_1 \neq t_2$, t_1 *is an* ***ancestor*** *of* t_2 *and* t_2 *a* ***descendent*** *of* t_1, *denoted by* $t_1 \prec t_2$, *if for every attribute in* $\cup_{i=1}^{k} F_i$ *where* t_1 *takes a non-*$*$ *value,* t_2 *takes the same value as* t_1. ■

Example 3 (Groups). *Using the schema in Example 2, a patient (p12, male, 30-40) is a base level group. A group (*, *, 30-40) is an aggregate patient group. (*, *, 30-40)* $\prec$ *(p12, male, 30-40).* ■

We can show the following immediately.

Lemma 1. *For a party* F_i *where the domain of every attribute is finite, all groups including the base level groups and the aggregate groups form a lattice under the relation* $\prec$.

Proof. The lemma follows with the definition of lattice immediately. ■

In theory, we can loosen the requirement in Lemma 1. As long as the domain of every attribute is either finite or countable, the lemma still holds. We notice that in practice a fact table is always finite and thus the domains of the attributes can be regarded finite in the analysis.

From the relational database point of view, groups can be regarded as a group-by tuples on subsets of attributes in a party.

Example 4 (Group as group-by)**.** *Consider the fact table* `FACT` *in Example 2. The group of patients according to gender are group-by tuples on attribute* `gender`*. Moreover, using group-by on attributes* `gender` *and* `age-group`*, we can form patient groups according to the combinations of those two attributes.* ■

Now, we can generalize our notion of relationships.

Definition 3 (Relationship)**.** *Given a fact table F of k parties, we extend the domain of each attribute in $\cup_{i=1}^{k} F_i$ such that meta-symbol $*$ is included as a special value. A* ***relationship*** *is a group-by tuple $t \in F$. That is, for every attribute in $\cup_{i=1}^{k} F_i$, t takes either a value in the domain of the attribute, or meta-symbol $*$.* ■

Apparently, if all groups are base level ones, a relationship is simply a tuple in the fact table. Recall that the measurement attributes F_{k+1} describe the relationship. When a relationship contains some aggregate groups, how can we describe the relationship using the measurement attributes? We can seek help from an aggregate function.

Definition 4 (Measurement of a relationship)**.** *Consider a fact table F of k parties. Let*

$$aggr : 2^{F_{k+1}} \rightarrow F_{k+1}$$

be an aggregate function. For any aggregate relation t, the ***measurement*** *of t is defined as*

$$t.F_{k+1} = aggr(\{s.F_{k+1} | s \in F, s \textit{ is a descendant of } t\}).$$

In other words, the measure of a relationship is the aggregate of the measurements of all base level relationships that are descendant of the target relationship. ■

Example 5 (Measurement of relationship)**.** *Consider the fact table* `FACT` *in Example 2 again. The measurement of the relationship between aggregate groups "female patients" and "doctors in BC" can be computed by aggregating the average from all relationships between every female patient and every doctor in BC that appear in the fact table. Here, the aggregate function is*

$$aggr(\{(avg_i, count_i)\}) = (\frac{\sum_i avg_i \times count_i}{\sum_i count_i}, \sum_i count_i).$$

■

In the rest of the thesis, we use average as the aggregate function due to its popular applications in practice. Moreover, for the sake of simplicity, we overload the term relationship to include both the groups and the measurement.

We have the following result about the relationships.

Theorem 1 (Relationship lattice)**.** *Given a fact table F of k parties, if the domain of every attribute in $\cup_{i=1}^{k} F_i$ is finite, then all relationships form a lattice $L = \prod_{i=1}^{k} L_{F_i}$, where L_{F_i} is the lattice of party F_i. Moreover, $|L| = |\prod_{i=1}^{k} L_{F_i}| = \prod_{A \in \cup_{i=1}^{k} F_i}(|A| + 1)$.*

Proof. Suppose that there are k parties. Let g_i and g_i' be two groups in party F_i $(1 \leq i \leq k)$ such that $g_i \preceq g_i'$, that is, either $g_i \prec g_i'$ or $g_i = g_i'$. Then, for relationships $(g_1, \ldots, g_k)$ and $(g_1', \ldots, g_k')$ on $\cup_{i=1}^{k} F_i$, we define $(g_1, \ldots, g_k) \preceq (g_1', \ldots, g_k')$. The claim that $\cup_{i=1}^{k} F_i$ is a lattice based on $\prec$ follows the definition of lattice immediately. Due to the construction of the relation $\prec$, the theorem holds. ■

Based on the theorem, the size of the space of relationships is exponential to the number of parties.

3.2 Relationship Outliers and Categorization

Now, let us model outliers in relationships. Outliers can be modeled in many possible ways. In this thesis, we follow the statistical outlier model that is simple and popularly used in business, industry and many applications.

Given a set of samples where each sample carries a numeric measurement, we can calculate the average m and the standard deviation δ. The well known Chebyshev inequality [9] tells the following.

Theorem 2 (Chebyshev inequality [9]). *Let X be a random variable with finite expected value m and non-zero variance δ. For any real number $l > 0$,*

$$Pr(|X - m| \geq l\delta) \leq \frac{1}{l^2}.$$

■

Therefore, we can experimentally use the l as outlier threshold. The samples that are more than $l\delta$ away from m are considered outliers.

Definition 5 (Outliers). *Given a fact table F and an outlier threshold l, where F_{k+1} contains only one attribute, let $\vec{m}$ and δ be the mean and the standard deviation, respectively, of F_{k+1} of all base level relationships. A relationship t is an outlier if $|t.F_{k+1} - \vec{m}| > l\delta$.* ■

Please note that we can easily extend Definition 5 to fact tables containing multiple measurement attributes.

We may find multiple outlier relationships. Are there any redundancy among relationship outliers? We have the following observation.

Theorem 3 (Weak Monotonicity). *Consider a fact table F of k parties and average is used as the aggregate function. Let t be an aggregate relationship and $A \in \cup_{i=1}^{k} F_i$ be an attribute where t takes value $*$. If t is an outlier, then there exists at least one relationship outlier t' such that (1) $t' = t$ on all attributes in $\cup_{i=1}^{k} F_i - \{A\}$, and (2) $t'.A \neq *$.*

Proof. We prove the theorem by contradiction. It is clear to know that if the relationship t' satisfies both conditions, then t' must be a child of the relationship t. Let us assume all children are not outliers. Then t' is not an outlier relationship. From Definition 5, if t is an outlier relationship then $|t.F_{k+1} - \vec{m}| > l\delta$. Suppose the children of t is t_i ($1 \leq i \leq n$), then $t' \in t_i$. From Definition 4 we can get

$$t.F_{k+1} = \frac{\sum_{i=1}^{n} t_i.F_{k+1}}{n}.$$

In our assumption all children are non-outlier relationships. Then we can get

$$|t_i.F_{k+1} - \vec{m}| \leq l\delta$$

Then

$$\vec{m} - l\delta \leq t_i.F_{k+1} \leq \vec{m} + l\delta$$

$$n(\vec{m} - l\delta) \leq \sum_{i=1}^{n} t_i.F_{k+1} \leq n(\vec{m} + l\delta)$$

$$\vec{m} - l\delta \leq \frac{\sum_{i=1}^{n} t_i.F_{k+1}}{n} \leq \vec{m} + l\delta$$

$$|\frac{\sum_{i=1}^{n} t_i.F_{k+1}}{n} - \vec{m}| \leq l\delta$$

Then, we can get

$$|t.F_{k+1} - \vec{m}| \leq l\delta$$

A contradiction. ∎

According to Theorem 3, if an aggregate relationship t is an outlier, then some descendant relationships of t must also be outliers. Then whether an aggregate relationship as an outlier provides significant extra information than its descendant outliers depends on whether it can summarizes those descendant outliers. Based on this observation, we categorize relationship outliers into two types.

- An aggregate relationship outlier t is a **type-I** outlier if most base level relationships in t are not outliers. In other words, t being an outlier is caused by a small number of descendants being outliers. Consequently, t itself may not be very interesting in terms of being an outlier. Instead, some outlying descendants are more interesting and should be checked.

- An aggregate relationship outlier t is a **type-II** outlier if many base level relationships in t are outliers. In other words, t is a good summary of a set of outlying descendants. Therefore, t is interesting in outlier detection.

To quantitatively define the two types of outlier, we use Kullback-Leibler divergence (KL-divergence for short) [32].

KL-divergence is a measure of the difference between two distributions in information theory [32]. The smaller the KL-divergence, the more similar the two distributions. It shows effectiveness to measure the similarity between uncertain objects in the paper [27] because the distribution difference cannot be captured by geometric distances directly. KL-divergence is a non-symmetric measurement, which means the difference from distribution P to Q is generally not the same as that from Q to P.

Definition 6 (Kullback-Leibler divergence [32])**.** *In the discrete case, for probability distributions P and Q of a discrete random variable their KL-divergence is defined to be*

$$KL(P|Q) = \sum_{x \in D} P(x) \ln \frac{P(x)}{Q(x)}$$

In the continuous case, for distributions P and Q of a continuous random variable, KL divergence is defined to be

$$KL(P|Q) = \int_D p(x) \ln \frac{p(x)}{q(x)} \, \mathrm{d}x,$$

where p and q denote the densities of P and Q, respectively.

In both discrete and continuous cases, KL divergence is only defined if P and Q both sum to 1 and if $Q(x) > 0$ for any i such that $P(x) > 0$. If the quantity $0 \ln 0$ appears in the formula, it is interpreted as zero, that is $0 \ln 0 = 0$.

The smaller the KL-divergence, the more similar the two distributions. KL-divergence is non-negative, but not symmetric, that is, in generally $KL(P|Q) \neq KL(Q|P)$.

The KL-divergence is always non-negative, which is a result know as Gibb's inequality. That is, $KL(P|Q) \geq 0$ with equality if and only if $P = Q$ almost everywhere.

Definition 7 (Types of outlier)**.** *Let t be a relationship outlier. Let S be the set of base level relationships that are descendants of t. S can be divided into two exclusive groups: S_0, the subset of normal relationships, and S_1, the subset of relationship outliers. $S = S_0 \cup S_1$ and $S_0 \cap S_1 = \emptyset$.*

*Relationship t is a **type-I outlier** if $KL(S|S_1) \geq KL(S|S_0)$; otherwise, t is a **type-II outlier**.* ■

Now, the problem is to find all outlier relationships and their types.

Chapter 4

Detection Methods

In this section, we develop outlier detection methods to find all relationship outliers and their types. We present the framework of the detection methods in Section 4.1. Two detection methods to find multi-level relationship outliers are shown in Section 4.2 and Section 4.3. And then we improve our methods in Section 4.4. In Section 4.5 we introduce using Kullback-Leibler divergence as similarity to determine which type the outlier relationship is in.

4.1 Framework

The framework of our method is presented in pseudo-code in Algorithm 1, which consists of two steps:

- Choose from 3 possible cubing approaches to detect multi-level relationship outliers: the group generalization approach (top-down cubing, TDC [45] for short), the group specification approach (bottom-up cubing, BUC [4] for short) and extended BUC (eBUC [43]) approach. Details of the first two cubing algorithms are shown in Section 4.2 and Section 4.3. We can get a list of relationship outliers after the cubing. Later in Section 4.4, we improve the BUC algorithm to eBUC [43], which is more efficient for iceberg cube computation.

- For those aggregate outliers (which means there is at least one star in $\cup_{i=1}^{k} F_i$), we use KL divergence to determine their types. The definition and implementation of KL divergence are in Section 4.5.

Algorithm 1: RELATIONSHIP-OUTLIER-DETECTION(*input*)

Input : *input*: The relationships to detect.
Output: *outlierRec*: Stores the detected relationship outliers.
outlierType: The outlier type of the relationships.

RELATIONSHIP-OUTLIER-DETECTION(*input*)
$CubingAlgorithm \leftarrow \{TDC, BUC, eBUC\}$;
for $CubingAlgorithm$ **do**
    if $isOutlier$ **then**
        $outlierRec.add()$;
        if $isAggregateOutlier$ **then**
            compute KL-divergence;
            write $outlierType$;
        end
    end
end

4.2 The Top-down Cubing (TDC) Approach

4.2.1 Top-down Cubing

The top-down cubing (TDC) [45] algorithm starts from the base level groups, groups that do not have any descendant, to the more general levels.

Top-down cubing, represented by Multiway Array Cube, is to aggregate simultaneously on multiple dimensions. It uses a multidimensional array as its basic data structure. The cubing process of TDC on a 4-D data cube is presented in Figure 4.1. We present how to detect outliers using TDC in Section 4.2.2.

4.2.2 Outlier Detection Using TDC

By Definition 5, we notice that given a fact table F and a threshold parameter l, a relationship is defined as an outlier if $|t.F_{k+1} - \vec{m}| > l\delta$, where m is the average and δ is the standard deviation of the samples. We use two cubing methods to process the cube lattice in top-down and bottom-up manners. In our example, we use the average charge amount as the aggregate measurement.

$$t.F_{k+1} = aggr(\{s.F_{k+1} | s \in F, s \text{ is a descendant of t}\}).$$

$$aggr(\{(avg_i, count_i)\}) = (\frac{\sum_i avg_i \times count_i}{\sum_i count_i}, \sum_i count_i).$$

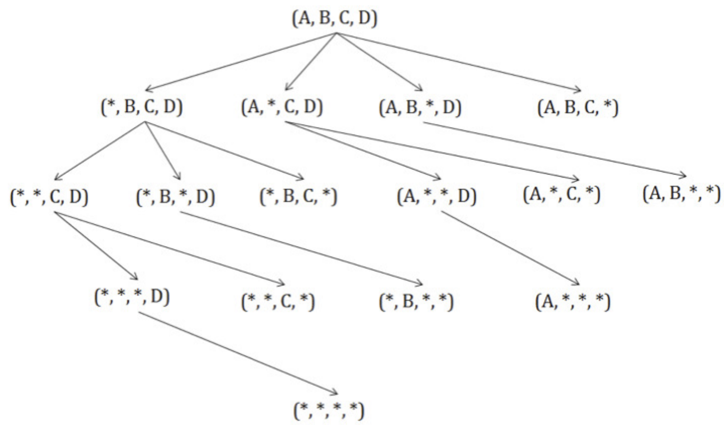


Figure 4.1: TDC cubing

Details of the TDC algorithm are provided in Algorithm 2. In the beginning, *dimensionRec* contains full dimensions, for example, if the relational input table has 4 dimensions A, B, C and D, then *dimensionRec* $= \{A, B, C, D\}$. Then we scan each combination of the dimensions in *dimensionRec* and get the average value. If the average value is in the defined outlier range, then we add the relationship into the *outlierRec*. If the relationship is a base level group, we add it into the *baseGroup* list and set the Boolean value *isOutlier*, which is to indicate whether this base level relationship is an outlier or not. If the outlier relationship is an aggregate group, we add it into the *aggregateOutlier* list. For the aggregate groups, we continue to compute the KL-divergence value using the function *computeKL-divergence()*. The concept of KL-divergence and details of how this function works are presented in section 4.5. Then we set the current dimension in *dimensionRec* to ALL (for example, *dimensionRec*$= \{*, B, C, D\}$) and recursively call itself with the start dimension incremented by 1.

4.2.3 Example of TDC

In this section, we give a running example using the top-down cubing.

Suppose the input table $F = (A, B, C, D, M)$, where A, B, C and D are dimensions and M is the measurement value. The table has 4 dimensions, namely $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, $C = \{c_1, c_2\}$ and $D = \{d_1, d_2\}$. The original table is listed in Table 4.1, and the cubing process is in Figure 4.2.

Algorithm 2: TDC($input, startDim, l$)

Input : $input$: The relationships to aggregate.
$startDim$: The starting dimension for this iteration.
l: The threshold parameter.

Global : constant $numDims$: The total number of dimensions.
constant m: The average value of the sample relationships.
constant δ: The standard deviation of the sample relationships.

Output: $outlierRec$: Stores the detected relationship outliers.

```
TDC(input, startDim, l)
for i ← startDim to numDims do
    currentGroup ← dimensionRec                    ▷ Current combination of dimensions
    avg ← aggregate(currentGroup)
    if |avg − m| > lδ then
        if currentGroup is base then
            baseGroup.add(currentGroup);
            baseGroup.isOutlier = true;
        else
            aggregateOutlier.add(currentGroup);
            computeKL − divergence(currentGroup);  ▷ Get outlier type
        end
        outlierRec.add(currentGroup);
    else
        if currentGroup is base then
            baseGroup.add(currentGroup);
            baseGroup.isOutlier = false;
        end
    end
    dimensionRec[d] = ALL;
    TDC(input, d + 1, l);
    dimensionRec[d] = d;
end
```

Id	A	B	C	D	M
1	a_1	b_1	c_1	d_1	m_1
2	a_1	b_1	c_2	d_1	m_2
3	a_1	b_2	c_2	d_2	m_3
4	a_2	b_1	c_1	d_1	m_4
5	a_2	b_2	c_1	d_1	m_5
6	a_2	b_2	c_2	d_2	m_6

Table 4.1: TDC algorithm example: original table

For the first recursive call (Figure 4.2(a)), the TDC algorithm scan the whole table because it starts from the most generalized groups. So the first recursion starts from the relationship (A, B, C, D).

The cubing process in Figure 4.1 shows that the second recursion of TDC cubing is in the relationship $(*, B, C, D)$. We list the cubing example of the second, third, fourth and fifth recursions in Figure 4.2(b), Figure 4.2(c), Figure 4.2(d) and Figure 4.2(e) respectively.

Id	Relationship	Measure
1	(a_1, b_1, c_1, d_1)	m_1
2	(a_1, b_1, c_2, d_1)	m_2
3	(a_1, b_2, c_2, d_2)	m_3
4	(a_2, b_1, c_1, d_1)	m_4
5	(a_2, b_2, c_1, d_1)	m_5
6	(a_2, b_2, c_2, d_2)	m_6

(a) first recursion (A, B, C, D)

Id	Relationship	Measure
1	$(*, b_1, c_1, d_1)$	$\frac{m_1+m_4}{2}$
2	$(*, b_1, c_2, d_1)$	m_2
3	$(*, b_2, c_1, d_1)$	m_5
4	$(*, b_2, c_2, d_2)$	$\frac{m_3+m_6}{2}$

(b) second recursion $(*, B, C, D)$

Id	Relationship	Measure
1	$(*, *, c_1, d_1)$	$\frac{m_1+m_4+m_5}{3}$
2	$(*, *, c_2, d_1)$	m_2
3	$(*, *, c_2, d_2)$	$\frac{m_3+m_6}{2}$

(c) third recursion $(*, *, C, D)$

Id	Relationship	Measure
1	$(*, *, *, d_1)$	$\frac{m_1+m_2+m_4+m_5}{4}$
2	$(*, *, *, d_2)$	$\frac{m_3+m_6}{2}$

(d) fourth recursion $(*, *, *, D)$

Id	Relationship	Measure
1	$(*, *, *, *)$	$\frac{m_1+m_2+m_3+m_4+m_5+m_6}{6}$

(e) fifth recursion $(*, *, *, *)$

Figure 4.2: TDC algorithm example

The algorithm will continue after the five recursions according to the cubing process in

Figure 4.1. When the algorithm finishes the top-down cubing, we can get a list of outliers and the types of aggregate outliers.

4.3 The Bottom-up Cubing (BUC) Approach

4.3.1 Bottom-up Cubing

The bottom-up cubing algorithm [4] is to build the cube by starting from a group-by on a single attribute, then a group-by on a pair of attributes, then a group-by on three attributes, and so on. It processes the cube lattice from the most general group to the base level groups. The cubing process of BUC on a 4-D data cube is presented in Figure 4.3. BUC begins with apex cuboid and counts the frequency of the first single dimension and then partitions the table based on the frequent values. And then it recursively counts the value combinations for the next dimension and partitions the table in the same manner. BUC can take advantage of Apriori pruning because if a given cell does not satisfy minimum support, then neither will any of its descendants. We will present how to detect outliers for our problem using BUC in Section 4.3.2.

4.3.2 Outlier Detection Using BUC

Details of the algorithm are provided in Algorithm 3. We can use BUC to compute all aggregate cells, and use the outlier determination criteria $|t.F_{k+1} - \vec{m}| > l\delta$ as a post-processing step to capture all outliers, where $\vec{m}$ and δ are the average and the standard deviation of F_{k+1} of all relationships among the base level groups.

The first step of the BUC algorithm is to aggregate the entire input and write the result. To get aggregate value we need to scan *input*. When scanning the whole table, we can get results of *baseGroup* list as a byproduct. The for loop in line 14 controls the current dimension, and partitions the current input. On return from $Partition()$, $dataCount$ contains the number of records for each distinct value of the d-th dimension while $dataAvg$ contains the average value of all the samples in the d-th dimension. If the average value is in the defined outlier range, we add the current dimension and measurement information into the final *outlierRec* list. Then the partition becomes the input relation in the next recursive call with the start dimension incremented by 1. We store the detected outliers into *outlierRec* and all the aggregate ones into *aggregateOutlier* list. For an aggregate outlier

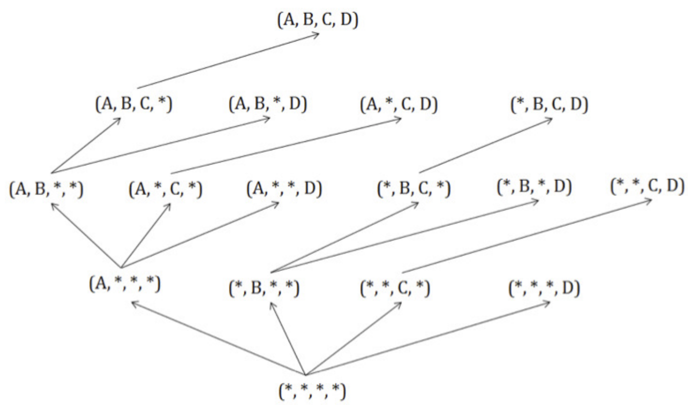


Figure 4.3: BUC cubing

group, we compute the KL-divergence of its descendants to get the outlier type. Details of the function *computeKL-divergence()* is described in section 4.5.

One tricky here is that the BUC relies on the monotonicity of iceberg conditions. However, the iceberg condition in our problem is not monotonic. To tackle with the problem in this thesis, BUC cannot take advantage of Apriori pruning because the measure here is not the count value. We use the average as measure, which does not hold the property that if a given cell is not an outlier, then neither will any of its descendant.

Fortunately, Theorem 3 identifies a weak monotonic property of the iceberg condition in our problem. We thus can adopt some special iceberg cube computation methods for weak monotonic conditions. Specifically, we use eBUC [43], which “looks ahead” to check whether an aggregate cell is an ancestor of some outliers. Details of eBUC are presented in Section 4.4

4.3.3 Example of BUC

In this section, we give a running example using the bottom-up cubing.

Suppose the input table $F = (A, B, C, D, M)$, where A, B, C and D are dimensions and M is the measurement value. The table has 4 dimensions, namely $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, $C = \{c_1, c_2\}$ and $D = \{d_1, d_2\}$. The original table is listed in Table 4.2, and the cubing process is in Figure 4.4.

For the first recursive call (Figure 4.4(a)), the BUC algorithm aggregates the whole

Algorithm 3: BUC($input, startDim, l$)

Input : $input$: The relationships to aggregate.
$startDim$: The starting dimension for this iteration.
l: The threshold parameter.

Global : constant $numDims$: The total number of dimensions.
constant m: The average value of the sample relationships.
constant δ: The standard deviation of the sample relationships.
$dataAvg[numDims]$: Stores the average value of each partition.
$dataCount[numDims]$: Stores the size of each partition.

Output: $outlierRec$: Stores the detected outlier relationships.

```
BUC(input, startDim, l)
for currentGroup ← tuple do                          ▷ To Agrrgate(input) and get baseGroup
    avg = currentGroup.measure;
    if |avg − m| > lδ then
        baseGroup.isOutlier = true;
        outlierRec.add(currentGroup);
    else
        baseGroup.isOutlier = false;
    end
end
if input.count()==1 then
    WriteAncestors(input[0], startDim) ;
    return;
end
for d ← startDim to numDims do
    Let C = cardinality[d];
    Partition(input, d, C, dataCount[d], dataAvg[d]);
    Let k ← 0;
    for i ← 0 to C do
        Let c = dataCount[d][i], a = dataAvg[d][i];
        currentGroup ← dimensionRec                  ▷ Current combination of dimensions
        if |avg − m| > lδ then
            if currentGroup is aggregate then
                aggregateOutlier.add(currentGroup);
                computeKL − divergence(currentGroup);   ▷ Get outlier type
            end
            outlierRec.add(currentGroup);
        end
        BUC(input[k...k + c], d + 1, l);
        dimensionRec[d] = d;
        k+ = c;
    end
    outputRec.dim[d] = ALL;
end
```

Id	A	B	C	D	M
1	a_1	b_1	c_1	d_1	m_1
2	a_1	b_1	c_2	d_1	m_2
3	a_1	b_2	c_2	d_2	m_3
4	a_2	b_1	c_1	d_1	m_4
5	a_2	b_2	c_1	d_1	m_5
6	a_2	b_1	c_1	d_2	m_6

Table 4.2: BUC algorithm example: original table

table. So the first recursion starts from the relationship $(*, *, *, *)$.

The cubing process in Figure 4.3 shows that the second recursion of BUC cubing is in the relationship $(A, *, *, *)$. We list the cubing example of the second, third, fourth and fifth recursions in Figure 4.4(b), Figure 4.4(c), Figure 4.4(d) and Figure 4.4(e) respectively.

The algorithm will continue after the five recursions according to the cubing process in Figure 4.3. When the algorithm finishes the bottom-up cubing, we can get a list of outliers and the types of aggregate outliers.

4.4 The Extended BUC (eBUC) approach

Although BUC is efficient to compute the complete data cube, it cannot take advantage of the Apriori property to prune in our problem. This is because our problem here does not satisfy the monotonic condition that if an aggregate relationship is not an outlier, then any descendent of it must also not be an outlier. If we use BUC directly, we need to compute the whole data cube without any pruning. To make the BUC algorithm more efficient for our problem, we use the extended BUC approach [43].

4.4.1 Outlier Detection Using eBUC

The key idea of the eBUC takes advantage of the week monotonic in Theorem 3. If an aggregate relationship t is an outlier, then there exists at least one descendent of t that is also an outlier.

Details of the eBUC algorithm is in Algorithm 4. The eBUC algorithm follows the structure of the BUC algorithm using depth-first search. It first aggregates the whole table and computes the base level relationships to get a list of base-level relationship outliers as a byproduct. The difference between eBUC and BUC is that, when eBUC encounters an

Id	Relationship	Measure
1	$(*, *, *, *)$	$\frac{m_1+m_2+m_3+m_4+m_5+m_6}{6}$

(a) first recursion $(*, *, *, *)$

Id	Relationship	Measure
1	$(a_1, *, *, *)$	$\frac{m_1+m_2+m_3}{3}$
2	$(a_2, *, *, *)$	$\frac{m_4+m_5+m_6}{3}$

(b) second recursion $(A, *, *, *)$

Id	Relationship	Measure
1	$(a_1, b_1, *, *)$	$\frac{m_1+m_2}{2}$
2	$(a_1, b_2, *, *)$	m_3
3	$(a_2, b_1, *, *)$	$\frac{m_4+m_6}{2}$
4	$(a_2, b_2, *, *)$	m_5

(c) third recursion $(A, B, *, *)$

Id	Relationship	Measure
1	$(a_1, b_1, c_1, *)$	m_1
2	$(a_1, b_1, c_2, *)$	m_2
3	$(a_1, b_2, c_2, *)$	m_3
4	$(a_2, b_1, c_1, *)$	$\frac{m_4+m_6}{2}$
5	$(a_2, b_2, c_1, *)$	m_5

(d) fourth recursion $(A, B, C, *)$

Id	Relationship	Measure
1	(a_1, b_1, c_1, d_1)	m_1
2	(a_1, b_1, c_2, d_1)	m_2
3	(a_1, b_2, c_2, d_2)	m_3
4	(a_2, b_1, c_1, d_1)	m_4
5	(a_2, b_2, c_1, d_1)	m_5
6	(a_2, b_1, c_1, d_2)	m_6

(e) fifth recursion (A, B, C, D)

Figure 4.4: BUC algorithm example

aggregate relationship t, it "looks ahead". That is, eBUC checks whether t is an ancestor of some base-level ourlier relationships. If not, then following Theorem 3, t cannot be an outlier relationship. Moreover, any descendent of t is not an ancestor of a base-level outlier relationship. So we do not need to search further to any descendent of t.

4.4.2 Example of eBUC

In this section, we show a running example using eBUC approach. The input table $F = (A, B, C, M)$, where A, B and C are dimensions and M is the measurement value. The table has 3 dimensions, namely $A = \{a_1, a_2\}$, $B = \{b_1, b_2, b_3\}$ and $C = \{c_1, c_2\}$. The original table is listed in Table 4.3. The average value of the whole table is 17.25 and the standard deviation is 31.12. Follow the outlier condition that $|t.F_{k+1} - \vec{m}| > l\delta$, we can first get the base table class (Table 4.4) when eBUC runs ($l = 1$). In this example, we only have one base-level outlier relationship (a_2, b_3, c_2). Any aggregate relationship that is not an ancestor of it will be pruned. In Figure 4.5(b), the relationship $(a_1, *, *)$ is pruned because it is not an ancestor of (a_2, b_3, c_2). Therefore, any descendent of $(a_1, *, *)$ will also be pruned. In Figure 4.5(c), we only check the descendents of $(a_2, *, *)$.

Id	A	B	C	M
1	a_1	b_1	c_1	7
2	a_1	b_1	c_2	9
3	a_1	b_2	c_1	4
4	a_1	b_2	c_2	7
5	a_1	b_3	c_1	10
6	a_1	b_3	c_2	8
7	a_2	b_1	c_1	5
8	a_2	b_1	c_2	11
9	a_2	b_2	c_1	7
10	a_2	b_2	c_2	15
11	a_2	b_3	c_1	4
12	a_2	b_3	c_2	120

Table 4.3: eBUC algorithm example: original table

Algorithm 4: eBUC($input, startDim, l$)

Input : $input$: The relationships to aggregate.
$startDim$: The starting dimension for this iteration.
l: The threshold parameter.
Global : constant $numDims$: The total number of dimensions.
constant m: The average value of the sample relationships.
constant δ: The standard deviation of the sample relationships.
$dataAvg[numDims]$: Stores the average value of each partition.
$dataCount[numDims]$: Stores the size of each partition.
Output: $outlierRec$: Stores the detected relationship outliers.

```
eBUC(input, startDim, l)
for currentGroup ← tuple do                          ▷ To Agrrgate(input) and get baseGroup
    avg = currentGroup.measure;
    if |avg − m| > lδ then
        baseGroup.isOutlier = true;
        outlierRec.add(currentGroup);
    else
        baseGroup.isOutlier = false;
    end
end
if input.count()==1 then
    WriteAncestors(input[0], startDim) ;
    return;
end
for d ← startDim to numDims do
    Let C = cardinality[d];
    Partition(input, d, C, dataCount[d], dataAvg[d]);
    Let k ← 0;
    for i ← 0 to C do
        Let c = dataCount[d][i], a = dataAvg[d][i];
        currentGroup ← dimensionRec                  ▷ Current combination of dimensions
        if currentGroup is an ancestor of some base-level outlier then
            if |avg − m| > lδ then
                if currentGroup is aggregate then
                    aggregateOutlier.add(currentGroup);
                    computeKL − divergence(currentGroup);          ▷ Get outlier type
                end
                outlierRec.add(currentGroup);
            end
            eBUC(input[k...k + c], d + 1, l);
            dimensionRec[d] = d;
        end
        k+ = c;
    end
    outputRec.dim[d] = ALL;
end
```

Id	A	B	C	M	Class
1	a_1	b_1	c_1	7	Normal
2	a_1	b_1	c_2	9	Normal
3	a_1	b_2	c_1	4	Normal
4	a_1	b_2	c_2	7	Normal
5	a_1	b_3	c_1	10	Normal
6	a_1	b_3	c_2	8	Normal
7	a_2	b_1	c_1	5	Normal
8	a_2	b_1	c_2	11	Normal
9	a_2	b_2	c_1	7	Normal
10	a_2	b_2	c_2	15	Normal
11	a_2	b_3	c_1	4	Normal
12	a_2	b_3	c_2	120	Outlier

Table 4.4: eBUC algorithm example: base table

Id	Relationship	Pruned
1	$(*, *, *)$	No

(a) first recursion $(*, *, *, *)$

Id	Relationship	Pruned
1	$(a_1, *, *)$	Yes
2	$(a_2, *, *)$	No

(b) second recursion $(A, *, *, *)$

Id	Relationship	Pruned
1	$(a_2, b_1, *)$	Yes
2	$(a_2, b_2, *)$	Yes
3	$(a_2, b_3, *)$	No

(c) third recursion $(A, B, *, *)$

Figure 4.5: eBUC algorithm example

4.5 Outlier Type Determination

In this section, we present how to determine the outlier type of aggregate relationships. First we show the framework of the outlier type determination in Section 4.5.1.

4.5.1 Framework

We have two types of outliers: **type-I** outlier means that most relationships among its base level groups are not outliers; and **type-II** outliers means many relationships among its base level groups are outliers. In this thesis, we propose to use KL-divergence as a similarity measure to determine whether an aggregate outlier relationship is more similar to its non-outlier or outlier base level groups.

We organize the type determination section in 4 steps:

- Firstly, we give an introduction and definition of KL-divergence in Section 4.5.2.
- Secondly, to use KL-divergence as a similarity measure for continuous domain in our problem, we need to estimate the probability density function by kernel density estimation (Section 4.5.3).
- Thirdly, we present how to use KL-divergence as similarity in Section 4.5.4.
- At last, we present our function to compute the KL-divergence and to determine the outlier type in Section 4.5.5.

4.5.2 KL-divergence

The concept of Kullback-Leibler divergence was originally introduced by Solomon Kullback and Richard Leibler in 1951 [32]. In probability theory and information theory, Kullback-Leibler divergence is used to measure the difference between two probability distributions. The smaller the KL-divergence, the more similar the two distributions. It should be noted that the KL-divergence is a non-symmetric measure. The difference from distribution P to distribution Q is generally not the same as that from Q to P.

Recall the definition of KL-divergence in Chapter 3.

Definition 8 (Kullback-Leibler divergence [32]). *In the discrete case, for probability distributions P and Q of a discrete random variable their KL divergence is defined to be*

$$KL(P|Q) = \sum_{x \in D} P(i) \ln \frac{P(i)}{Q(i)}$$

In the continuous case, for distributions P and Q of a continuous random variable, KL divergence is defined to be

$$KL(P|Q) = \int_D p(x) \ln \frac{p(x)}{q(x)} \, \mathrm{d}x,$$

where p and q denote the densities of P and Q.

In both discrete and continuous cases, KL divergence is only defined if P and Q both sum to 1 and if $Q(i) > 0$ for any i such that $P(i) > 0$. If the quantity $0 \ln 0$ appears in the formula, it is interpreted as zero, that is $0 \ln 0 = 0$.

4.5.3 Kernel Density Estimation

In our fact table $F = \cup_{i=1}^{k+1} F_i$, $F_i (1 \leq i \leq k)$ contains the information about the i-th party while F_{k+1} contains the information about the relationship among k instances from the k parties, which is also called the measurement. If the measurement is continuous, we use kernel density estimation to estimate the probability density function.

Kernel density estimation [37] [38] is a non-parametric way to estimate the probability density function of a random variable. Let $(x_1, x_2, x_3, ..., x_n)$ be an independent and identically distributed (*iid* for short) sample drawn from some distribution with an unknown density f, the kernel density estimator is

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

Where $K(\cdot)$ is the kernel and $h > 0$ is the smoothing parameter called the bandwidth, h is to control the level of smoothing. A range of kernel functions are commonly used, in this thesis, we use the popular adopted Gaussian kernels and the Gaussian approximation [38], and set $h = 1.06 \times \delta |P|^{-\frac{1}{5}}$ as suggested by [38], where δ is the standard deviation of the samples in P. In 1-dimensional case, the density estimator is

$$P(x) = \frac{1}{|P|\sqrt{2\pi}h} \sum_{p \in P} e^{-\frac{(x-p)^2}{2h^2}}$$

4.5.4 Using KL Divergence as Similarity

This section presents how KL divergence is used as similarity to measure two distributions. We borrow the idea from [27]. We apply the continuous case in our data set since the measurement in this thesis is continuous.

Given two distributions P and Q, $KL(P|Q)$ returns the difference of distribution of Q given the distribution of P. Therefore, the larger the KL divergence, the more different between the two distributions.

Given the samples of P and Q, by the law of large numbers we have

$$\lim_{x \to \infty} \frac{1}{m} \sum_{i=1}^{m} \ln \frac{P(p_i)}{Q(p_i)} = KL(P|Q),$$

where $P = \{p_1, p_2, ..., p_m\}$ is the set of samples. Furthermore, the KL divergence can be estimated as

$$\hat{KL}(P|Q) = \frac{1}{m} \sum_{i=1}^{m} \ln \frac{P(p_i)}{Q(p_i)}$$

In this thesis, we compare the KL divergence of a detected aggregate outlier relationship with its non-outlier and outlier relationship descendants. According to Definition 7 in Chapter 3, let t be an aggregate outlier relationship and S be the set of relationships that are descendants of t and involve only base level groups. S_0 and S_1 are exclusive subsets of S, where S_0 denotes the non-outlier relationships and S_1 denotes the outlier relationships. We have $S = S_0 \cup S_1$ and $S_0 \cap S_1 = \emptyset$. If $KL(S|S_1) \geq KL(S|S_0)$, t is a type-I outlier (most base level groups are not outliers); otherwise, it is a type-II outlier (many base level relationships are outliers). So, for an aggregate relationship outlier t, we compare

$$KL(S|S_0) = \frac{1}{m}\sum_{i=1}^{m} \ln \frac{S(s_i)}{S_0(s_i)}$$

and

$$KL(S|S_1) = \frac{1}{m}\sum_{i=1}^{m} \ln \frac{S(s_i)}{S_1(s_i)}$$

where $S = \{s_1, s_2, ..., s_m\}$. The domain of S is always bounded by the lowest and largest values in S.

4.5.5 Computing KL-divergence

There is a function *computeKL-divergence()* in Algorithm 2 and Algorithm 3. This subsection will show how this function works to get the outlier type. Details of the function is presented in Algorithm 5.

For the TDC algorithm, we cube the input from the most generalized level. So we can get the *baseGroup* list containing all the base level relationships from the first recursive call. For the BUC and eBUC algorithms, we can get the same list from the $Aggregate(input)$ process. The *baseGroup* list records both the dimension information of the relationship and a flag *isOutlier* to indicate whether this base group is an outlier or not.

When the cubing algorithms encounter an aggregate group which is detected as an outlier, it continues to call the function *computeKL-divergence*. Input of this function is the aggregate relationship outlier, and the output is the outlier type of this relationship. First, we scan the *baseGroup* list to get the set S_0 (non-outlier descendant) and S_1 (outlier descendant). Then compute the KL-divergence value using the formula in Section 4.5.4. If $KL(S|S_1) \geq KL(S|S_0)$, then this is a type-I outlier; otherwise, it is a type-II outlier.

Algorithm 5: COMPUTKL-DIVERGENCE(*aggregateOutlier*)

Input : *aggregateOutlier*: The detected outlier aggregate group.
Global : *baseGroup*: The base level groups
Output: *outlierType*: The outlier type of the input.

COMPUTEKL-DIVERGENCE(*aggregateGroup*)
for *currentGroup* $\leftarrow$ *baseGroup* **do**
    $S_0 \leftarrow currentGroup.isOutlier = false$;
    $S_1 \leftarrow currentGroup.isOutlier = true$;
end
compute $KL(S|S_0)$, $KL(S|S_1)$;
if $KL(S|S_1) \geq KL(S|S_0)$ **then**
    $outlierType$ = Type-I;
else
    $outlierType$ = Type-II;
end

Chapter 5

Experiment Results

We conducted extensive experiments on both real and synthetic data sets to evaluate our outlier detection methods.

Our programs were implemented in C++ using Microsoft Visual Studio 2010. All experiments were conducted on a PC computer with an Intel Core Duo E8400 3.0 GHz CPU and 4 GB main memory running the Microsoft Windows 7 operating system.

5.1 Case Studies of Outliers

To test the effectiveness of our methods, we used a real data set related to service business. For the sake of privacy protection, the data is anonymized and the location and service information is coded in a way that it cannot be mapped back to the original data meaningfully.

The data set contains two parties: the consumers and the service providers. The customers are described by two attributes: `gender` and `age-group`. The service providers are described by two attributes: `location` and `service`. We used the attribute `amount` as the measurement. The average amount, as calculated in Example 5, is used for each aggregate relationship. The fact table contains 5,895 base level relationships. The average amount at the base level is 63.50 and the standard deviation is 83.51.

Table 5.1 shows two relationship outliers as examples detected by our methods using threshold $l = 2$. To fully understand those two outliers, we list all base-level relationships of them in Tables 5.2 and 5.3, respectively.

As shown in Table 5.2, R_1 has 8 base level relationships. Among them, 6 are normal

Id	Outlier Relationship (gender, age, location, code)	Type
R_1	$(Female, *, A101, S91)$	I
R_2	$(Female, *, A101, S31)$	II

Table 5.1: Examples of relationship outliers

Id	Base-level Relationships (gender, age, location, code)	Amount	Normal or outlier
1	(Female, 21-40, A101, S91)	125.00	Normal
2	(Female, 41-60, A101, S91)	0.00	Normal
3	(Female, 41-60, A101, S91)	230.00	Normal
4	(Female, 41-60, A101, S91)	222.50	Normal
5	(Female, 41-60, A101, S91)	200.00	Normal
6	(Female, 41-60, A101, S91)	160.00	Normal
7	(Female, 41-60, A101, S91)	1106.25	Outlier
8	(Female, 41-60, A101, S91)	1900.00	Outlier

Table 5.2: Base-level relationships of R_1

cases and 2 are outliers. For R_1, $KL(S|S_0) = 2.39$ and $KL(S|S_1) = 63.74$. $KL(S|S_0) < KL(S|S_1)$. The distribution of R_1 is more similar to the distribution of its normal base level descendant relationships. Thus, R_1 is of type I. R_1 being a type-I outlier indicates that R_1 contains some base level strong outliers (those of amount 1000.00 or more), but the majority in R_1 is still normal. To this extent, R_1 as an outlier may not be very interesting. Instead, those outlier descendants of R_1 should be checked.

As shown in Table 5.3, R_2 has 10 base-level relationships. Among them, 2 are normal cases and 8 are outliers. $KL(S|S_0) = 5.01182$ and $KL(S|S_1) = 0.454034$. $KL(S|S_0) > KL(S|S_1)$. The distribution of R_1 is more similar to the distribution of its outlier base level descendant relationships. R_2 is of type II. R_2 being a type-II outlier indicates that most of the base level descendant relationships of R_2 are outliers, and can be summarized using R_2.

5.2 Efficiency and Scalability

We tested the efficiency of the outlier detection methods on both real data set and the synthetic ones. We compare three cubing methods: TDC [45], BUC [4] and eBUC [43]. TDC and BUC compute the whole cube while eBUC uses the outlier detection criterion directly to compute an iceberg cube.

We used a larger real data set to test the efficiency, and use random samples of various

Id	Base-level Relationships (gender, age, location, code)	Amt	Normal or outlier
1	(Female, 41-60, $A101$, $S31$)	135.00	Normal
2	(Female, 41-60, $A101$, $S31$)	130.03	Normal
3	(Female, 41-60, $A101$, $S31$)	694.00	Outlier
4	(Female, 41-60, $A101$, $S31$)	694.00	Outlier
5	(Female, 41-60, $A101$, $S31$)	694.00	Outlier
6	(Female, 41-60, $A101$, $S31$)	555.20	Outlier
7	(Female, 41-60, $A101$, $S31$)	402.67	Outlier
8	(Female, 41-60, $A101$, $S31$)	624.60	Outlier
9	(Female, 41-60, $A101$, $S31$)	555.20	Outlier
10	(Female, 41-60, $A101$, $S31$)	555.20	Outlier

Table 5.3: Base-level relationships of R_2

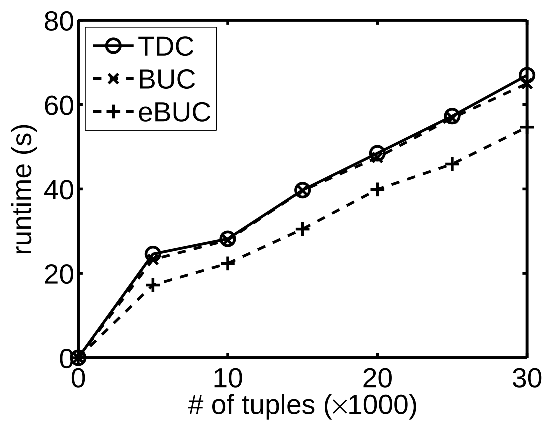
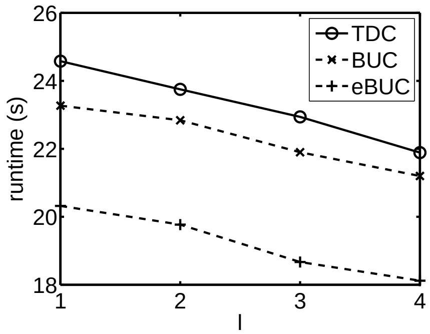


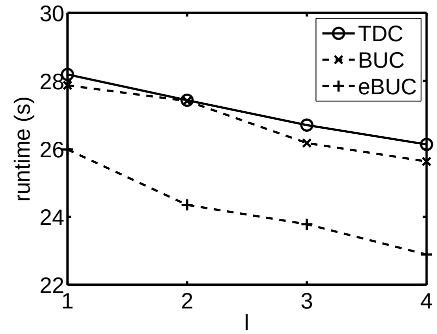
Figure 5.1: The running time of TDC, BUC and eBUC with respect to the number of tuples

sizes of the data set to test the scalability. Figure 5.1 shows the scalability of the three methods on the database size, where the outlier threshold $l = 1$. Please note that the smaller the value of l, the more outliers and thus the less pruning power eBUC has. The three methods have linear scalability. The runtime of TDC and BUC is very close. eBUC clearly can take the advantage of direct pruning using the outlier detection criterion, and thus has faster runtime.

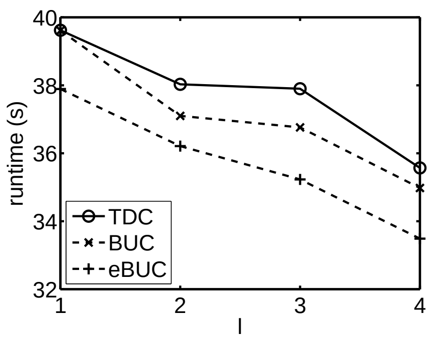
Figure 5.2 shows the scalability of the three methods with respect to parameter l and number of base level relationships on the real data set. The larger the value of l, the less outliers, and thus the less computation is needed in all three methods to determine the types of the outliers. Comparing to TDC and BUC, eBUC can further use the outlier criterion to prune normal relationships during the cubing process and thus reduces the runtime further.



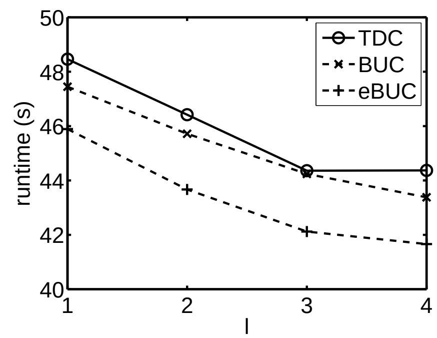
(a) number of tuples=5,000



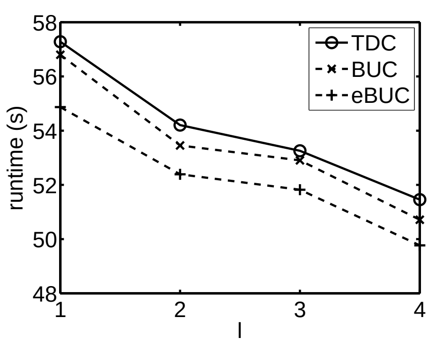
(b) number of tuples=10,000



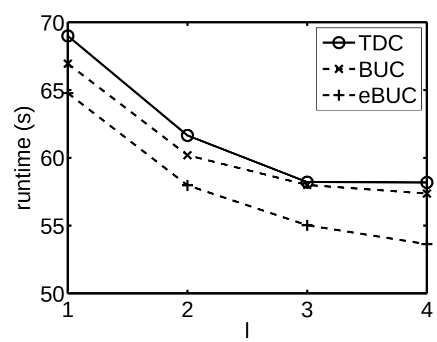
(c) number of tuples=15,000



(d) number of tuples=20,000



(e) number of tuples=25,000



(f) number of tuples=30,000

Figure 5.2: The running time of TDC, BUC and eBUC

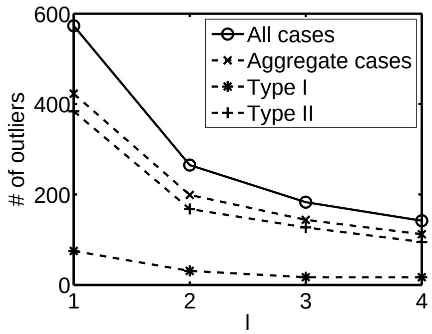
The larger the value of l, the more advantage eBUC has. This figure also indicates that the determination of types of outliers takes substantial cost.

Figure 5.3 shows the numbers of relationship outliers with respect to parameter l and number of base level relationships on the real data set. As explained, the larger the value of l, the less outliers. The trend in Figure 5.3 is consistent with the runtime trend in Figure 5.2. Moreover, most of the outliers detected are aggregate ones. There are much more type-II outliers than type-I ones. The results clearly justify the effectiveness of our method in summarizing the outlier information.

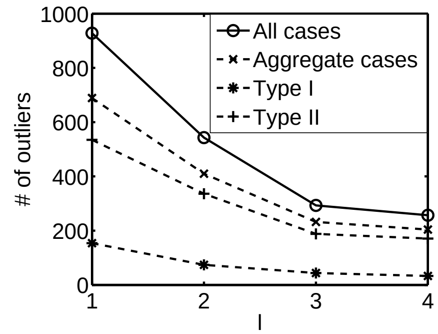
We also tested the efficiency and scalability using synthetic data sets. We generated synthetic data sets with dimension attributes in discrete domains and measurement in continuous domain. We consider 3 factors in out experiments, the dimensionality d, the number of tuples n in the data set and the distribution on the dimensions (uniform distribution versus normal distribution).

To be specific, we generated two data sets, each of $100,000$ tuples and 4 dimensions. The cardinality in each dimension is 20. The tuples in the first data set follow the uniform distribution in each dimension, and the tuples in the second data set follow the (discretized) normal distribution. That is, we used the normal distribution $\mu = 10$ and $\theta = 2$ to generate synthetic data, and round the values to 20 bins in the range of $[1, 20]$. Figure 5.4 shows the results, where threshold parameter $l = 1$. Clearly, our outlier detection methods work much faster on the data set of normal distribution, where outliers are meaningful.

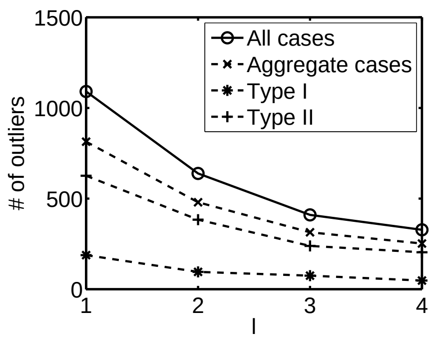
Figure 5.5 shows the scalability of our detection methods on the synthetic data set following normal distribution. In Figure 5.5(a), the number of tuples is set to $10,000$. The runtime increases dramatically when the dimensionality goes up. This is as expected since computing a data cube of high dimensionality is known challenging. In Figure 5.5(b), the dimensionality is set to 4, and the number of tuples varies from $100,000$ to 1 million. Interestingly, the runtime grows slower than the number of tuples. The reason is given the cardinality of each dimension, the number of possible groupbys is fixed. When a fact table becomes very large, many aggregate cells will be populated with a non-trivial number of tuples where the number of groupbys grows slower (Figure 5.6). The results clearly show that our methods are scalable on large data sets.



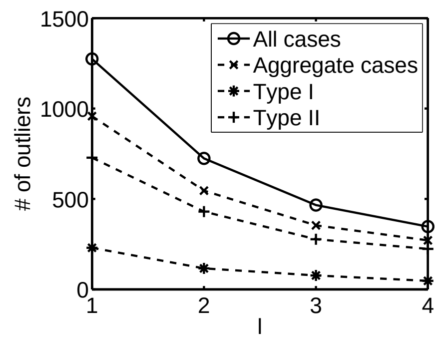
(a) number of tuples=5,000



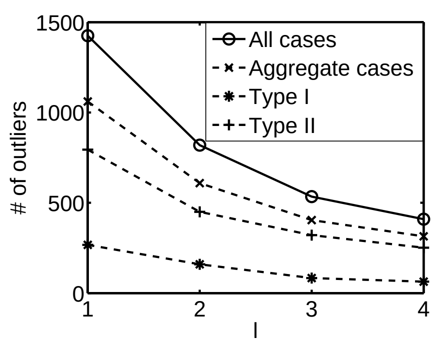
(b) number of tuples=10,000



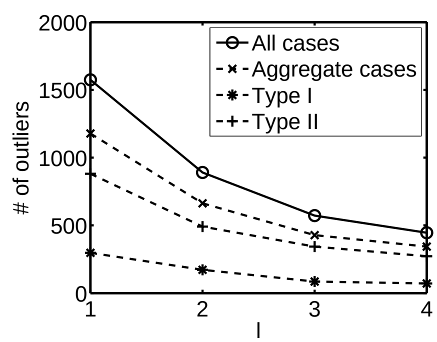
(c) number of tuples=15,000



(d) number of tuples=20,000

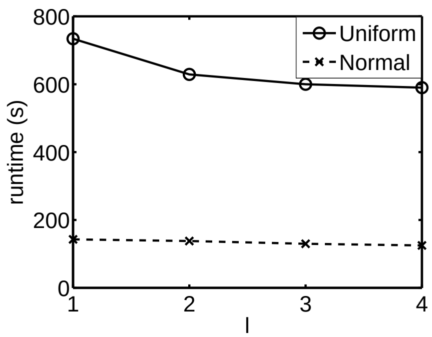


(e) number of tuples=25,000

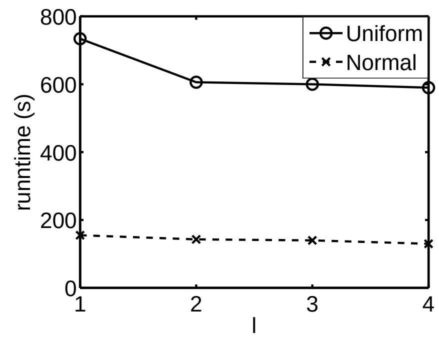


(f) number of tuples=30,000

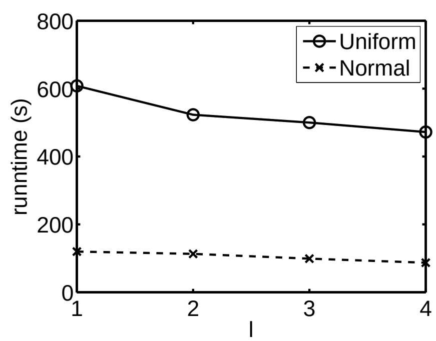
Figure 5.3: Number of Detected Outliers



(a) TDC algorithm

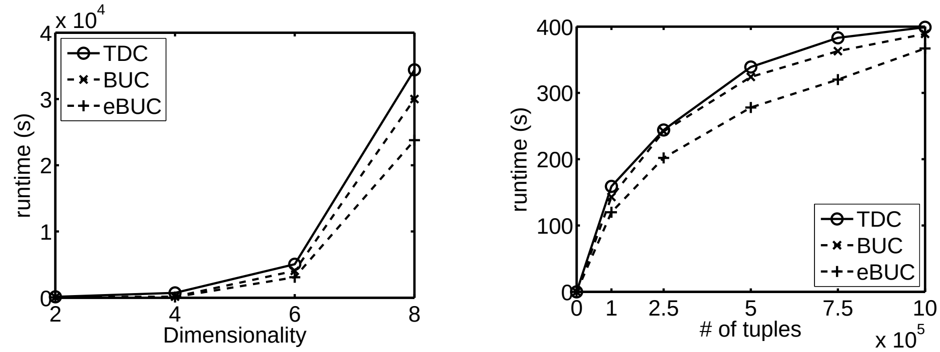


(b) BUC algorithm



(c) eBUC algorithm

Figure 5.4: The running time of TDC, BUC and eBUC with different distributions



(a) Dimensionality (b) Database size

Figure 5.5: Scalability on synthetic data.

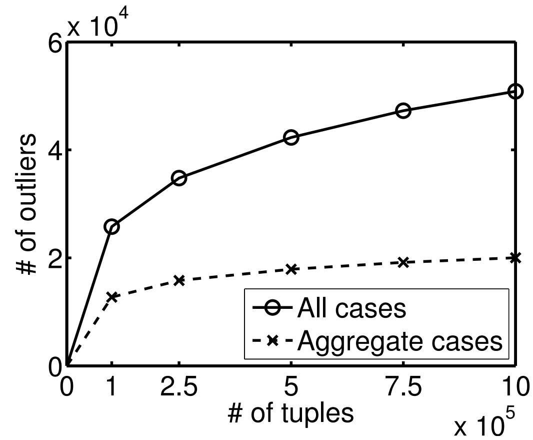


Figure 5.6: Number of detected outliers

Chapter 6

Conclusion

In this thesis, we tackle the problem of detecting multi-level relationship outliers. Relationship is very important in business. Good relationships build a win-win situation for both consumers and providers. On the other hand, fraudulent relationships destroy normal business patterns and make a vicious circle between consumers and providers. In this thesis, we develop a simple yet effective model to detect and handle relationship outliers between groups and individuals in groups. The most important thing is that the attributes in the relationships form a hierarchy so that traditional outlier detection methods cannot directly be used in our problem. We apply three algorithms: TDC, BUC and eBUC for cubing. We use KL-divergence as similarity to identify the outlier type. In business, the most challenging thing for business analysts is that, they do not know the fraud pattern for the existing relationships. The outlier type in this thesis means fraud pattern. Once we know how the most popular fraud pattern looks like, we can come up with policies to prevent such situation. We use KL-divergence as similarity measure because relationships are complicated and not independent. Merely compare the values of two relationships is not sufficient to estimate their similarity. We need to see the distributions of the two relationships. KL-divergence is tested to be useful to find potential outlier patterns in our data set. The detected outlier results show that our detection methods are effective. The running time tested on both real and synthetic data sets shows that our detection methods are efficient.

As future work, there are several interesting directions we can further work on.

- Extending our techniques to arbitrarily multiple parties. In this thesis, we only tackle a fact table of 2 parties. In the future we can test our detection methods on data sets with multiple parties. The problem will be much more complicated and challenging.

- Improve the cubing process. In this thesis, we use eBUC algorithm to improve the BUC cubing. It has been tested that eBUC is more efficient than BUC for our problem. One interesting question is that, can we do even better? The answer is yes. We can further improve the cubing process using Dynamic Ordering (DYNO). In [43], DYNO shows better performance than eBUC.

- Refine the results with data summarization to make our methods more useful in business. Our methods can detect a list of outlier relationships, however, to make it useful in business we need to further refine it. For example, we can apply algorithm in [43] to get "probable groups". Probable group is an aggregate cell that satisfies some requirement like "What are the groups of relationships that have 80% or more outlier descendants?" We can generate the most general probable groups. For example, if both relationships $(female, 20, B.C.)$ and $(*, 20, B.C.)$ are probable, we only return the latter one. In this way, the pattern of fraudulent relationships between service providers and consumers will be succinct and the detected results can be better understood by business people.

Bibliography

[1] M. Agyemang, K. Barker, and R. Alhajj. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intell. Data Anal.*, 10:521–538, 2006.

[2] Z. A. Bakar, R. Mohemad, A. Ahmad, and M. M. Deris. A comparative study for outlier detection techniques in data mining. In *Proc. 2006 IEEE Conf. Cybernetics and Intelligent Systems*, pages 1–6, Bangkok, Thailand, 2006.

[3] M. Berry and G. Linoff. *Mastering data mining: The art and science of customer relationship management*. John Wiley & Sons, Inc., 1999.

[4] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cube. *ACM SIGMOD Record*, 1999.

[5] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical Science*, 17:235–249, 2002.

[6] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 2009.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41:1–58, 2009.

[8] N.V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[9] P.L. Chebyshev. *Sur les valeurs limites des intégrales*. Imprimerie de Gauthier-Villars, 1874.

[10] J.B.R. Cheng and AR Hurson. Effective clustering of complex objects in object-oriented databases. *ACM SIGMOD Record*, 1991.

[11] M.C. Cooper, D.M. Lambert, and J.D. Pagh. Supply chain management: more than a new name for logistics. *The International Journal of Logistics Management*, 1997.

[12] D. Dasgupta and N.S. Majumdar. Anomaly detection in multidimensional data using negative selection algorithm. In *Proc. 2002 Congress on Evolutionary Computation (CEC'02)*, pages 1039–1044, Washington DC, 2002.

[13] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proc. 17th Int. Conf. Machine Learning (ICML'00)*, 2000.

[14] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Proc. 2002 Int. Conf. of Data Mining for Security Applications*, 2002.

[15] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of Data Mining in Computer Security*, 6:77–102, 2002.

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

[17] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1:291–316, 1997.

[18] R. Fujimaki, T. Yairi, and K. Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proc. 2005 Int. Workshop on Link Discovery (LinkKDD'05)*, pages 401–410, Chicago, Illinois, 2005.

[19] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational operator generalizing group-by, cross-tab and sub-totals. In *Proc. 1996 Int. Conf. Data Engineering (ICDE'96)*, pages 152–159, New Orleans, Louisiana, Feb. 1996.

[20] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, 3^{rd} Edition.* Morgan Kaufmann, 2011.

[21] D.M. Hawkins. *Identification of outliers.* Chapman and Hall London, 1980.

[22] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.

[23] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 2004.

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

[25] J. Huang, H. Shimizu, and S. Shioya. Clustering gene expression pattern and extracting relationship in gene network based on artificial neural networks. *Journal of bioscience and bioengineering*, 2003.

[26] W.H. Inmon. *Building the data warehouse.* Wiley-India, 2005.

[27] B. Jiang, J. Pei, Y. Tao, and X. Lin. Clustering uncertain data based on probability distribution similarity. *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[28] M.V. Joshi, R.C. Agarwal, and V. Kumar. Mining needle in a haystack: classifying rare classes via two-phase rule induction. In *ACM SIGMOD Record*, volume 30, pages 91–102. ACM, 2001.

[29] M.V. Joshi, R.C. Agarwal, and V. Kumar. Predicting rare classes: Can boosting make any weak learner strong? In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 297–306. ACM, 2002.

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

[31] Y. Kou, C.T. Lu, S. Sirwongwattana, and Y.P. Huang. Survey of fraud detection techniques. In *IEEE International Conference on Networking, Sensing and Control*. IEEE, 2004.

[32] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 1951.

[33] C. X. Lin, B. Zhao, Q. Mei, and J. Han. A statistical model for popular event tracking in social communities. In *Proc. 2010 ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD'10)*, Washington D.C., July 2010.

[34] A. Payne and P. Frow. A strategic framework for customer relationship management. *Journal of Marketing*, 2005.

[35] C. Phua, D. Alahakoon, and V. Lee. Minority report in fraud detection: classification of skewed data. *ACM SIGKDD Explorations Newsletter*, 6(1):50–59, 2004.

[36] K. Ross and D. Srivastava. Fast computation of sparse datacubes. In *Proc. 1997 Int. Conf. Very Large Data Bases (VLDB'97)*, pages 116–125, Athens, Greece, Aug. 1997.

[37] D.W. Scott. *Multivariate density estimation*. Wiley Online Library, 1992.

[38] B.W. Silverman. *Density estimation for statistics and data analysis*. Chapman & Hall/CRC, 1986.

[39] A. Strehl and J. Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 2003.

[40] R. Vilalta and S. Ma. Predicting rare events in temporal domains. In *Data Mining, 2002. ICDM 2002. Proceedings. 2002 IEEE International Conference on*, pages 474–481. IEEE, 2002.

[41] G.M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 359–363, 1998.

[42] D. Xin, J. Han, X. Li, and B. W. Wah. Star-cubing: Computing iceberg cubes by top-down and bottom-up integration. In *Proc. 2003 Int. Conf. Very Large Data Bases (VLDB'03)*, pages 476–487, Berlin, Germany, Sept. 2003.

[43] H. Yu, J. Pei, S. Tang, and D. Yang. Mining most general multidimensional summarization of probable groups in data warehouses. In *Proceedings of the 17th international conference on Scientific and statistical database management*. Lawrence Berkeley Laboratory, 2005.

[44] K. Zhang, S. Shi, H. Gao, and J. Li. Unsupervised outlier detection in sensor networks using aggregation tree. *Advanced Data Mining and Applications*, pages 158–169, 2007.

[45] Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'97)*, pages 159–170, Tucson, AZ, May 1997.