Multidimensional Benchmarking

by

Akiko Campbell

P.B.D, Simon Fraser University, 1999
B.A., Keio University, 1987

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY

Spring 2016

Approval

Name:	**Akiko Campbell**
Degree:	**Doctor of Philosophy**
Title:	***Multidimensional Benchmarking***
Examining Committee:	**Chair:** Dr. Jiannan Wang Assistant Professor

Dr. Jian Pei
Senior Supervisor
Professor

Dr. Uwe Glässer
Supervisor
Professor

Dr. Fred Popowich
Internal Examiner
Professor
School of Computing Science

Dr. Hui Xiong
External Examiner
Professor
Management Science and Information Systems
Rutgers, the State University of New Jersey

Date Defended/Approved: March 14, 2016

Abstract

Benchmarking is a process of comparison between performance characteristics of separate, often competing organizations intended to enable each participant to improve its own performance in the marketplace (Kay, 2007). Benchmarking sets organizations' performance standards based on what "others" are achieving. Most widely adopted approaches are quantitative and reveal numerical performance gaps where organizations lag behind benchmarks; however, quantitative benchmarking on its own rarely yields actionable insights. It is important for organizations to understand key drivers for performance gaps such that they can develop programs for improvement around them. In this thesis, we develop a multidimensional analysis approach to benchmarking to characterise the properties of key drivers as a step towards "qualitative" benchmarking. Specifically, our approach systematically identifies significant benchmarks, compares organizations in statistical manners, and reveals the most manifesting aspects of uniqueness of an organization of interest. We also evaluate our algorithmic development using systematic empirical studies and show that our methods are effective and efficient.

Keywords: benchmarking, key drivers, multidimensional analysis, business intelligence, industry applications

Dedication

To my extended family for their continued encouragement.

Acknowledgements

I would like to thank my supervisory committee that supported my desire to work on industry oriented topics and provided direction for my thesis. Most importantly, I would like to express my utmost gratitude to Professor Pei for his tireless effort in guiding me to the path which led to the completion of this thesis. Finally, I would like to thank Guanting Tang and Xiango Mao for their assistance and collaboration in authoring important components in chapters 3 and 5 of the thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1.

Introduction

1.1. Motivation

Benchmarking has been given many different definitions by different organizations and authors. One of the widely accepted definitions of benchmarking is by Camp (1989) which states, “The continuous process of measuring products, services and practices against the toughest competitors or those companies recognized as industry leaders.”

In practice, most widely adopted approaches for benchmarking are quantitative. Quantitative benchmarking aims to provide a single numerical estimate of organization’s efficiency relative to that of comparable organizations. For example, utility regulators often use quantitative benchmarking to inform their assessments of the current efficiency of utility organizations (e.g. water suppliers, electricity distribution companies, etc.) (Holder et al., 2006). Quantitative benchmarking can help uncover numerical performance gaps; however, it does not reveal why the gaps exist or indicate whether the gaps are significant. According to Baldry et al. (2009), “virtually every credit card issuer practices benchmarking. Yet all too often, issuers limit themselves to quantitative performance measures, without taking into account either more qualitative measures or the strategic demands of the markets in which they operate. As a result, just 20% of issuers gain any tangible benefits from their benchmarking efforts.”

Organizations conduct benchmarking to identify areas for continuous improvement to remain competitive in the market. To that end, it is important for organizations to understand key drivers as “qualitative” measures for performance gaps. For example, once a utility firm determines that its service efficiency is 20% below the benchmark (quantitative), it would want to understand what is driving the inefficiency (qualitative) such that a service improvement program can be developed.

How can one establish systematic measures to characterize key drivers; that is, primary factors driving the performance gaps? One way is to develop contexts and determine how organizations are positioned in them. Do they stand out positively or negatively in certain contexts, or are they part of the big mass? If they stand out, what facets make up such contexts?

A context is a set of circumstances or facts that surround a particular event or situation. Put in the benchmarking setting, performance metrics can be the circumstances that surround organizations' situations; as such, the contexts can be formed by multiple metrics or dimensions in which organizations want to appraise their performance; hence, the name "multidimensional benchmarking".

A key benefit of the multidimensional approach to benchmarking lies in its ability to corroborate the findings of each metric and derive further insights from the combination of the metrics. It facilitates organizations' understanding in the underlying reasons for performance gaps and permits them to move beyond target setting to design change initiatives that close those gaps and improve their competitive positioning. However, a multidimensional benchmarking result cannot be represented by a single numeric value produced by a simple arithmetic; that is, sum all performance metric measures and average them over the number of metrics involved. There may be some weighting functions that can be incorporated into the arithmetic to represent importance of certain metrics over others; however, weighting is somewhat biased subject to the opinions of those who determine the weights or the importance. In the age of Big Data, there should be tools to allow "data" to provide us with a more "objective" perspective.

1.2. Problem Statement

Benchmarking is not specific to competitive analyses in commercial settings. Metrics can be defined according to a framework on how an organization wants to classify its performance; for example, the state of the health and the wellness of their employees compared to the general population, within specific industry verticals, within specific employment types, etc.

In a hypothetical scenario, an organization is interested in benchmarking its workforce health against the general workforce population. The results may be graphically represented as quantitative performance gaps of individual metrics as shown in Figure 1. In Figure 1, an organizational health index consists of 10 metrics; each metric quantifying the prevalence rate of an illness condition (e.g. mental health) across its workforce. The blue bar indicates the organization's performance of a metric while the orange line denotes the benchmark (i.e. a standard or point of reference against which the organization is compared). Taking the mental health and heart & stroke as examples, this organization is performing at the rate above the benchmark for mental health and below the benchmark for heart & stroke.

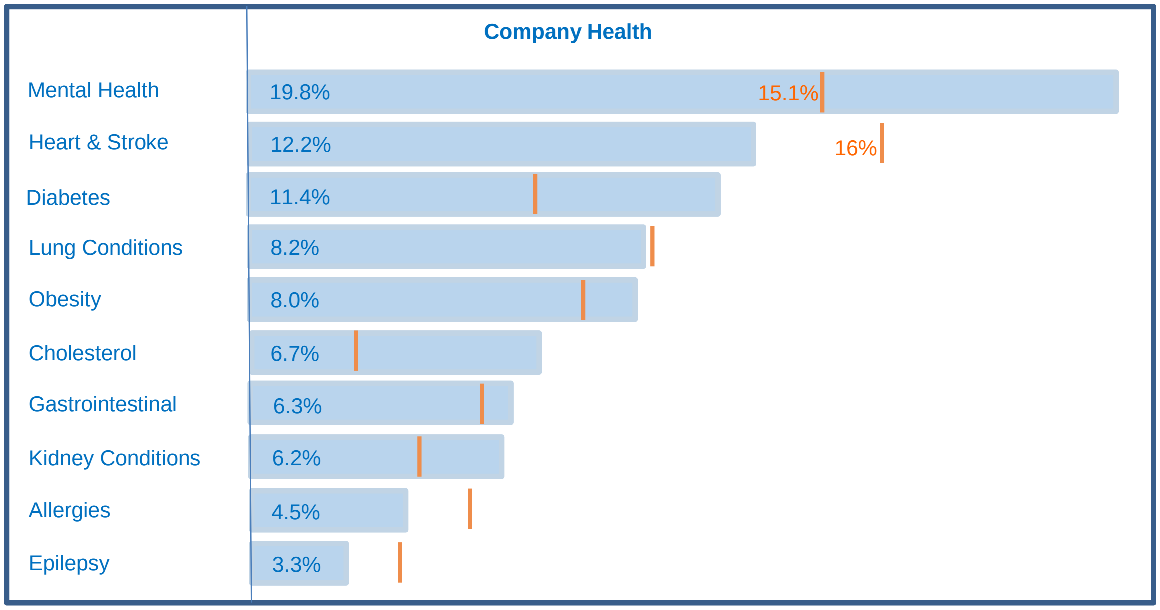


Figure 1 Example Benchmarking

This clearly shows a quantitative performance gap for each metric. However, these individual performance gaps do not divulge what characteristics and circumstances of the workforce are driving these performance gaps. Can the drivers be gender, age group, occupation type, location, work hours, or combinations of them?

In this thesis, we develop systematic methods to compare the performance of an object of interest (a query object) against others (benchmarks) in the context of multiple dimensions (i.e. characteristics and circumstances) surrounding the query object as well as the benchmark. If there is a significant performance gap between the query object and the benchmark, then these dimensions can be translated as key drivers for the gap. Technically, we divide the problem into 3 sub-problems and develop solutions for them:

1. **Benchmarks in data warehouse**. We develop two algorithmic approaches for efficient computation for identifying meaningful benchmarks in data warehouses; Sorted Inverted Index Cube (SIIC) and Dominant Answer Materialization (DAM) method.
2. **Reflective benchmarking**. We develop methods to understand how the exceptions and outliers within an aggregate group contribute to the overall deviation of the group from the norm. We identify two types of groups. The first type corresponds to the groups where the deviation is mainly caused by a small number of outlying units and the second type to those where a majority of underlying units are outliers.
3. **Subspace analysis**. We conduct multidimensional analysis to identify key drivers which form subspaces that manifest the uniqueness of the group of interest. We consider both situations where the units are not labeled and where the units belong to different classes.

1.3. Structure of Thesis

The rest of the thesis is structured as follows:

- **Chapter 2 Related Work** will outline benchmarking, data warehouse and Online Analytical Processing (OLAP), outlier detection, and subspace analysis.
- **Chapter 3 Benchmarks in Data Warehouse** will develop SIIC and DAM for efficient computation for online analytical queries.
- **Chapter 4 Reflective Benchmarking** will demonstrate the use of KL-divergence to compare two probability distributions to determine the type of outliers.
- **Chapter 5 Subspace Analysis** will demonstrate 2 techniques for subspace analysis: Contrast Subspaces and Outlying Aspects.
- **Chapter 6 Conclusion** concludes this thesis and suggests some future directions.

Chapter 2.

Related Work

The primary objective of the thesis is to explore and evaluate techniques in computing science that can be applied to multidimensional benchmarking. This chapter first presents the general overview of benchmarking, covering the concept, the history, and the methods commonly adopted in the industry. The overview is intended to set the context for the proposed work in this thesis and highlight the claim that although benchmarking is common in business, techniques in computing science have been largely unexplored for effective benchmarking.

The chapter then introduces related work in areas of computing science that we wish to consider for multidimensional benchmarking. These areas include data warehouse and online analytical processing (OLAP), outlier detection, and subspace analysis.

2.1. Benchmarking

Key motive for benchmarking is continuous improvement. Throughout history, people have developed methods and tools for setting, maintaining and improving standards of performance. Desire to improve performance and the actual improvement can be traced far back to prehistoric forms of benchmarking in the industrial history. For example, in the early 1800's, an American industrialist, Francis Lowell, traveled to England where he studied leading textile manufacturing techniques and industrial design of the mill factories. He realized that although the factory equipment was sophisticated, there was room for improvement in the way the plants were laid out for labour. Using technology very similar to what he had seen in England, Lowell built a new plant in the U.S.; however, the factory functioned in a less labour intensive fashion (Bogan et al., 1994).

Many researchers agree that the recognition of benchmarking as a useful management tool was formalized in early 1980's when Xerox employed benchmarking as part of its “Leadership through Quality”, a program to find ways to reduce manufacturing costs. In the early 1980’s, Xerox found itself increasingly vulnerable to intense competition from both the US and the Japanese competitors. Its operating cost was high and its products were of relatively inferior quality in comparison to its competitors’. In 1982, Xerox determined that the average manufacturing cost of copies in Japanese companies was 40-50% of that of Xerox’s and they were able to undercut Xerox’s prices effortlessly. As part of the “Leadership through Quality”, Xerox established the benchmarking program which played a major role in pulling Xerox out of trouble in the years to come. Xerox since then has become one of the best examples of the successful implementation of benchmarking (IBS, 2006).

2.1.1. Definitions

Benchmarking has been given many different definitions by different organizations and authors; however, all definitions concur that benchmarking is an integral step for continuous improvement. Table 1 lists representative definitions of benchmarking:

Table 1 Benchmarking Definitions

Author	Definition
Camp (1989)	The continuous process of measuring products, services and practices against the toughest competitors or those companies recognized as industry leaders.
Geber (1990)	A process of finding the world class examples of a product, service or operational system and then adjusting own products, services or systems to meet or beat those standards.
Vaziri (1992)	A continuous process comparing an organisation's performance against that of the best in the industry considering critical consumer needs and determining what should be improved.
Watson (1993)	The continuous input of new information to an organisation.
Klein (1994)	An excellent tool to use in order to identify a performance goal for improvement, identify partners who have accomplished these goals and identify applicable practices to incorporate into a redesign effort.
Cook (1995)	A kind of performance improvement process by identifying, understanding and adopting outstanding practices from within the same organization or from other businesses.
American Productivity and Quality Center (1999)	The process of continuously comparing and measuring an organization against business leaders anywhere in the world to gain information that will help the organisation take action to improve its performance.

2.1.2. Types of benchmarking

Different types of benchmarking can be identified on the basis of what is being compared. Generally, there are 3 types of benchmarking: strategic, performance, and process benchmarking.

Strategic Benchmarking

Strategic benchmarking examines how organizations compete. It is used to identify strategic imperatives that have enabled high performing organizations to be successful.

Performance Benchmarking

Performance benchmarking pertains to the comparison of organization's key processes, products and services to assess its competitive positioning. It usually focuses on prices, quality, features, speed, reliability and other performance metrics.

Process Benchmarking

Process benchmarking is for organizations to learn how their selected processes are performing compared to most efficient operating practices from several organizations in similar operational functions. Unlike strategic and performance benchmarking, process benchmarking focuses on selected production processes in an organization rather than on the organization as a whole. The presumption behind the analysis is that by identifying best practice processes and comparing actual processes that organizations utilize, the management can improve the performance of sub-systems, leading to better overall performance.

2.1.3. Techniques for benchmarking

By the target of the comparison, benchmarking techniques can be categorized into two types: internal and external benchmarking.

Internal Benchmarking

Internal benchmarking is performed between departments within the same organization or between organizations operating as part of a chain in different countries (Cross et al., 1994; Breiter et al., 1995). When any part of an organization has a better performance indicator, others can learn how this was achieved; it can then be used as a baseline for extending benchmarking to include external organizations (McNair et al., 1992; Karlof et al., 1993).

External Benchmarking

External benchmarking requires a comparison of work with external organizations in order to discover new ideas, methods, products, and services (Cox et al., 1998). The objective is continuously to improve one's own performance by measuring how it performs, comparing it with that of others and determining how the others achieve their performance levels. External benchmarking provides opportunities for learning from the best practices and experience of others who are at the leading edge. Within external benchmarking, there are 3 types including competitive, functional, and generic benchmarking.

- **Competitive Benchmarking** refers to a comparison with direct competitors only. Its benefits include creating a culture that values continuous improvement to achieve excellence by increasing sensitivity to changes in the environment external to the organization (Vaziri, 1992). However, it is often difficult to obtain data from competitors and lessons to be learned from them.
- **Functional Benchmarking** refers to comparative research whereby a comparison of business performance is made not only against competitors but also against the best businesses operating in similar fields and performing similar activities or having similar problems but in a different industry (Davies, 1990; Breiter et al., 1995). For example, British Rail Network South East used benchmarking to improve the standard of cleanliness on trains. British Airways was chosen for comparison since a team of 11 people cleans a 250 seat jumbo aircraft in only 9 minutes. Following the benchmarking exercise, a team of 10 was able to clean a 660 seat train in 8 minutes (Cook, 1995).
- **Generic Benchmarking** refers to the comparisons of business functions that are the same regardless of the domain of business. For example, a finance department of an insurance company would be compared to the finance department of a telecom company that has been identified as having the most efficient operations (e.g. fastest turnaround time).

2.1.4. Benchmarking methods

Frontier Models

Common forms of quantitative methods lend from economic efficiency analysis which involve parametric and non-parametric techniques. The primary objective of both is to measure the **technical efficiency**, which is defined as the ability of a producer to produce maximum output from a given set of inputs. Technical efficiency thus is translated as the success indicator of performance measure by which producers are evaluated. Given the importance of technical efficiency analysis, several models of frontiers have been developed. Frontier models are based on the premise that efficient producers are those that operate on the production frontier, while inefficient producers are those operating below the production frontier and the level of inefficiency is measured by the level of deviation from the frontier (Ajibefun, 2008).

Stochastic Frontier Production Function

The core economic theory underlying the formulation of a cost frontier supposes that the minimum cost a producer can achieve, when using the most efficient technology available, are a function of its output and the prices of its inputs. The cost function is based on the behaviour of a representative cost-minimising producer who is able to control the amount of each input used subject to producing a given output. The method assumes a particular specification of the relationship between an organization's costs and a set of cost drivers, which may include, for example, the outputs produced, input prices and a range of exogenous factors. Econometric analysis is then used to estimate the parameters of that relationship. Having estimated a cost function, inefficiency is one of the factors (alongside others, such as, omitted variables, measurement errors, etc.) that can explain the differences between the observed level of costs for a particular organization and the level of cost predicted by the estimated cost function (Holder et al., 2006).

The stochastic frontier production function illustrates a producer using n inputs ($x_1, x_2, \dots, x_n$) to produce output y. It assumes the presence of technical inefficiency of production and is defined as:

$$y_i = f(x_i; \beta) \exp(v_i - u_i)\,, i = 1, 2, \dots, n$$

where y_i is the observed scalar output of the producer i, x_i is a vector of n inputs used by the producer i, $f(x_i; \beta)$ is the production frontier, β is a vector of technology parameters to be estimated, v is a random error associated with random factors (hence stochastic) and u is the amount by which the producing unit fails to reach the optimum (i.e. the frontier).

The technical efficiency TE_i of a producer i is defined in terms of the ratio of the observed output to the corresponding optimal frontier output:

$$TE_i = \frac{y_i}{y_i^*} = \frac{f(x_i, \beta) \exp(v_i - u_i)}{f(x_i, \beta) \exp(v_i)} = \exp(u_i)$$

where y_i is the observed output and y_i^* is the frontier output. $TE_i = 1$ indicates that the organization i obtains the maximum feasible output, while $TE_i < 1$ provides a measure of the shortfall of the observed output from maximum feasible output.

The major advantage of this method is that it allows the test of hypothesis concerning the goodness of fit of the model. The “stochastic” aspect of the model allows it to handle appropriately measurement problems and other stochastic influences that would otherwise show up as causes of inefficiency (Greene, 2005). However, the major drawback is that it requires specification of technology, which may be restrictive in most cases (Ajibefun, 2008).

Data Envelopment Analysis (DEA)

DEA is a non-parametric linear programming technique widely used in the operations research and management science literature (Holder et al., 2006).

DEA estimates the cost level an efficient organization should be able to achieve in a particular market. The model seeks to determine an envelopment surface, also referred to as the efficiency frontier. Rather than estimating the impact of different cost drivers, DEA establishes an efficiency frontier (taking account of all relevant variables) based on the “envelope” of observations. Each organization is then assigned an efficiency score based on its proximity to the estimated efficiency frontier.

With DEA, the efficient frontier is the benchmark against which the relative performance of organizations is measured. Given a certain sample of organizations, all organizations should be able to operate at an optimal efficiency level which is determined by the efficient organizations in the sample. These efficient organizations determine the efficiency frontier. The organizations that form the efficient frontier use the minimum quantity of inputs to produce the same quantity of outputs. The distance to the efficiency frontier provides a measure for the efficiency or its lack thereof.

The objective of the linear programming is to maximize efficiency, where efficiency is the ratio of weighted outputs to weighted inputs and restricted to a range of 0 to 1. To maximize the efficiency score θ for producer 0:

$$Maximize\ \theta = \frac{\sum_{r=1}^{s} u_r y_{r0}}{\sum_{i=1}^{m} v_i x_{i0}}$$

where:

$\theta =$ efficiency of the producer 0

$u_r = s$ output coefficients of the producer 0

$y_{r0} = s$ output weighting coefficients for the producer 0

$v_i = m$ input coefficients for the producer 0

$x_{i0} = m$ input weighting coefficients for the producer 0

This is subject to the constraint that when the same set of u and v coefficients is applied to all other producers being compared, no producer will be more than 100% efficient such that:

$$\frac{\sum_{r=1}^{s} u_r y_{rj}}{\sum_{i=1}^{m} v_i x_{ij}} \leq 1 \text{ for } j = 1, \ldots, n, \text{and}$$

$$u_{r,} v_j \geq 0 \text{ for } r = 1, \ldots, s \text{ and } i = 1, \ldots, m.$$

The main advantage of this method is its ability to accommodate a multiplicity of inputs and outputs. However, the results are potentially sensitive to the selection of inputs and outputs; thus, their relative importance needs to be analyzed prior to the calculation. Further, there is no way to test their appropriateness. The number of efficient organizations on the frontier tends to increase with the number of inputs and output variables. When there is no relationship between explanatory factors (within inputs and/or within outputs), DEA views each organization as unique and fully efficient and efficient scores are very close to 1, which results in a loss of discriminatory power of the method (IBNET, 2015).

Frontier Analysis Case Study

The use of frontier analysis is widespread in incentive-based regulation of utilities in which reimbursement is guided by the cost efficiency of service provision. Lovell (2003) claims that the setting in which hospitals are reimbursed is structurally similar to that of the setting of revenue caps in utilities regulation and demonstrates the value of frontier analysis in the hospital reimbursement exercise. Given a vector $x = (x_1, \ldots, x_n)$ of resources to produce a vector $y = (y_1, \ldots, y_m)$ of services, in the provision of its services, each hospital incurs expense $w^T x = \sum_{i=1}^{n} w_i x_i$ where $w = (w_1, \ldots, w_n)$ is a vector of resource prices. Figure 2 shows 12 hospitals with scalar output representing the multidimensional service vector. The relationship is generally positive although some hospitals provide more service at lower costs than some others.

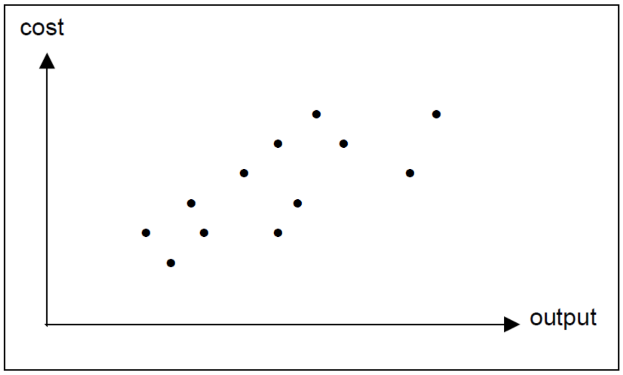


Figure 2 Relationship between Cost and Output

The objectives of frontier analysis in this case study are to uncover the nature of the relationship between service provision and expenditure and to evaluate the performance of each hospital. “Performance” in this context means the ability to minimize expenditure required to provide a service vector y in light of input price vector w and other exogenous variables represented by vector $z = (z_1, \ldots, z_k)$ whose elements characterize the operating environment. The minimum cost frontier $c(y, w, z)$ expresses the desired nature of the relationship between service provision and minimum required expenditure. This supplies the benchmark against which to evaluate the performance of individual hospitals. The performance is evaluated in terms of the cost efficiency $CE(y, w, z, x) = \frac{c(y,w,z)}{w^T x} \leq 1$ with cost efficient hospitals having $CE(y, w, z, x) = 1$.

Stochastic cost frontier is $w^T x = c(y, w, z) \cdot \exp\{v + u\}$ where the actual expenditure $w^T x$ equals minimum expenditure $c(y, w, z)$ times the two error components $\exp\{v + u\}$. As noted earlier, $\exp\{v\}$ captures the statistical noise reflecting random events beyond the control of the hospital and $\exp\{u\}$ expresses the magnitude of the hospital's inefficiency. The line in Figure 3 depicts the stochastic cost frontier for the same hospitals in Figure 2.

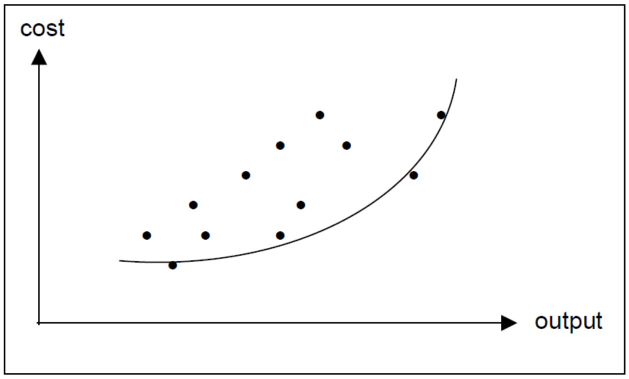


Figure 3 Deterministic Kernel of a Stochastic Cost Frontier

Figure 4 illustrates the DEA formulation of the minimum cost frontier. This formulation constructs the tightest fitting piecewise linear surface that envelops the cost-output combinations. There are 4 cost efficient hospitals and the cost efficiency of any other hospital is calculated as the ratio of the minimum cost of providing its service vector to its actual expenditure.

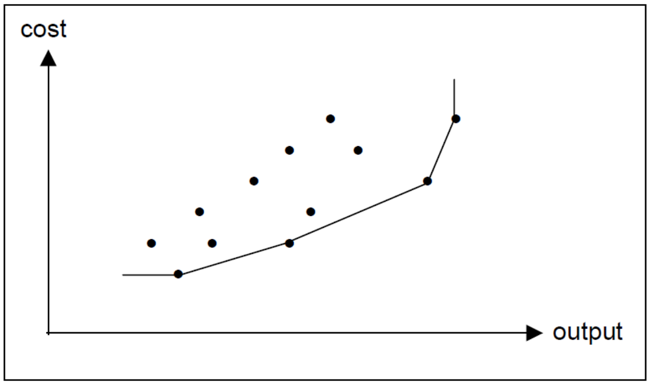


Figure 4 DEA Cost Frontier

Other Models

Among the most successful commercialization of "benchmarking" ideas in the Information Technology industry are Gartner Magic Quadrant (Figure 5) and Forrester Wave (Figure 6) which provide, in a 2-dimensional performance metric space, visual representations of technology vendors’ positions in the markets in which they compete. Many buyers rely on Gartner Magic Quadrant or Forrester Wave to understand the competitive positioning of technology vendors in different markets and technology vendors in turn use them as strategic marketing tools.

A Gartner Magic Quadrant is a culmination of research in a specific market, giving audience a wide-angle view of the relative positions of the market's competitors. It provides a competitive positioning of four types of technology providers including the following (Gartner, 2015):

- **Leaders** execute well against their current vision and are well positioned for tomorrow;
- **Visionaries** understand where the market is going or have a vision for changing market rules, but do not yet execute well;
- **Niche Players** focus successfully on a small segment, or are unfocused and do not out-innovate or outperform others;
- **Challengers** execute well today or may dominate a large segment, but do not demonstrate an understanding of market direction.

Technology vendors are positioned in the 4 quadrants representing these 4 types in the 2-dimensional performance space. Vendors positioned in the upper right quadrant (i.e. Leader’s quadrant) are the strongest in the market in terms of the completeness of the vision and their ability to execute.

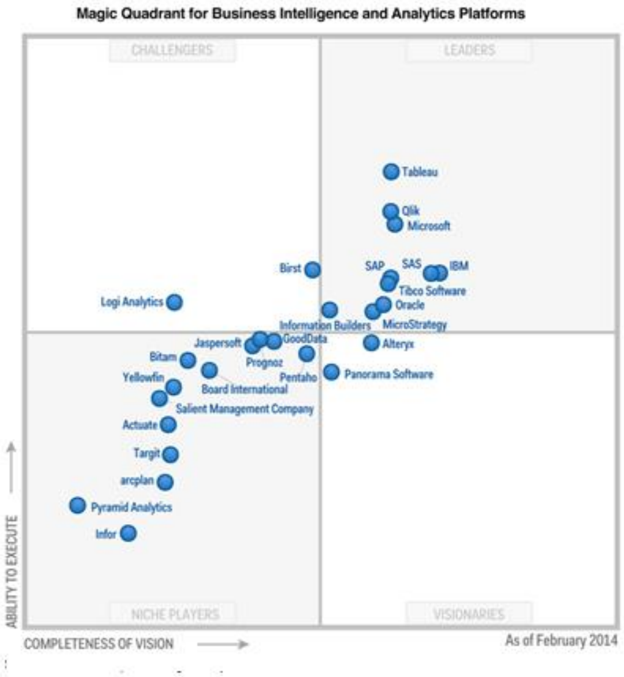


Figure 5 Sample Gartner Magic Quadrant (Business Intelligence and Analytics Platforms, Q1 2014)

However, the method Gartner uses to benchmark vendors are not publically known and it has been criticised for catering more towards investors and large vendors than towards buyers; much of the criticism centred on the lack of disclosure of the money received from the vendors it rates, raising conflict of interest (Wikipedia, 2015).

The Forrester Wave is Forrester's evaluation of vendors in a software, hardware, or services market. In the Forrester Wave reports and spreadsheets, it exposes both the criteria that it uses to grade the vendor offerings and how it scores and weight those criteria. Forrester Wave evaluations are driven by Forrester’s analysis of data collected from the marketplace and the experience of its analysts. Technology vendors are positioned in a 2-dimensional space according to the weighted average of the scores they are evaluated for (Forrester, 2015). Vendors positioned in the upper right corner are the strongest in the market in terms of the strength of the strategy and their current offering. Forrester Wave overlays another dimension, “Market presence” on the 2-dimensional space.

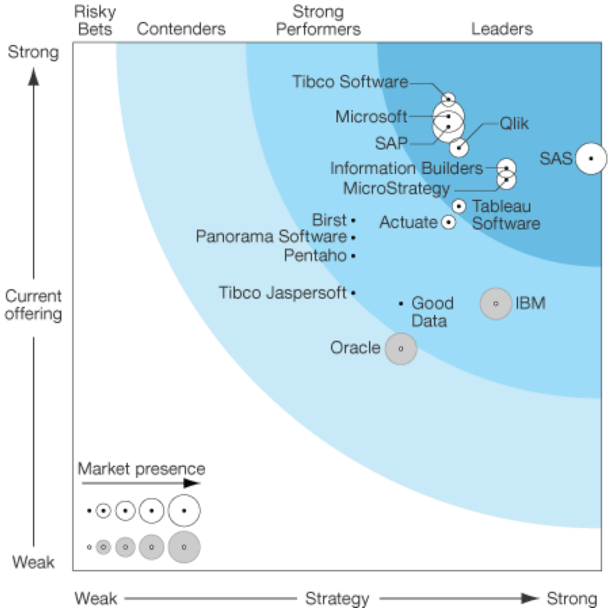


Figure 6 Sample Forrester Wave (Agile Business Intelligence Platforms, Q3 2014)

2.2. Data Warehouse and Online Analytical Processing (OLAP)

Data warehouses and Online Analytical Processing or OLAP are two fundamental components of business intelligence systems.

A data warehouse is a database containing multidimensional data that usually represents the business history of an organization. The historical data is used for analysis that supports business decisions at various levels, from strategic planning to performance evaluation of a discrete organizational unit. OLAP enables data warehouses to be used effectively for online analysis, providing rapid responses to iterative complex analytical queries. OLAP's multidimensional data model and data aggregation techniques organize and summarize large amounts of data such that it can be evaluated quickly using online analysis and graphical tools. The answer to a query into multidimensional data often leads to subsequent queries as analysts search for answers or explore further possibilities. OLAP provides the speed and the flexibility to support analysts in real time (Microsoft, 2015).

The introduction of data cube (Gray et al., 1997) is considered a landmark in data warehousing. A data cube consists of dimensions and facts and allows materialization of multidimensional data in large data repositories to facilitate fast online data analysis. However, a data cube typically has many dimensions and the curse of dimensionality becomes a technical problem; for example, a data cube with 20 dimensions, each containing 99 distinct values, has $(99 + 1)^{20} = 10^{40}$ base and high-level cells. This is too large a volume to be pre-computed and stored with reasonable resources. This warrants computing iceberg cubes (Beyer et al., 1999) instead of complete cubes. An iceberg cube contains only those cells that meet an aggregate condition. It is called an iceberg cube because it contains only some of the cells of the full cube, like the tip of an iceberg. The aggregate condition could be, for example, minimum support or a lower bound on count, average, minimum or maximum. The purpose of the iceberg cube is to identify and compute only those values that will most likely be required for decision support queries. The aggregate condition specifies which cube values are more meaningful and should therefore be stored.

The value of iceberg cube is obvious. A data cube can be viewed as a lattice of cuboids whereby cuboids whose group-by's include more dimensions are at a lower level than those that include fewer dimensions and the cuboid that include all dimensions is at the bottom. The lower level cuboids likely contain trivial aggregate values and are unlikely to satisfy threshold conditions; thus, no need to be computed. This not only saves processing time and disk space but also focuses analysis only on interesting data.

With iceberg cubes, the emphasis is to develop algorithms to answer iceberg queries efficiently. Beyer et al. (Beyer et al., 1999) proposed the algorithm BUC which computes iceberg cubes with monotonic aggregate functions. Han et al. (Han et al., 2001) developed a method for computing iceberg queries with non-monotonic aggregate functions. Ng et al. (Ng et al., 2001) studied iceberg queries with distributed systems. Chen et al. (Chen et al., 2008) explored iceberg cube computation in shared-nothing clusters. Lo et al. (Lo et al., 2008) extended iceberg queries to sequence data. Chen et al. (Chen et al., 2009) extended iceberg queries to graphs. Recently, He et al. (He et al., 2013) used patterns as “dimensions” in iceberg queries on sequences.

While we can adopt some of these algorithms for efficient computation of data cubes, when we consider benchmarking, we need a notion of a "query object" and the ability to compare the properties of the query object to those of the others' at different levels of aggregate hierarchy.

Sarawagi et al. (1998) proposes a discovery-driven exploration paradigm which guides analysts to explore anomalies (referred to as "exceptions") by means of pre-computed indicators of exceptions at various levels of details in the cube. It considers all descendant cells for each aggregate cell and aims to provide a navigation guidance for analysts to browse interesting regions of a cube.

In gradient analysis (Dong et al., 2001), given a probe aggregate cell q, one can find all pairs of aggregate cells (q, v) such that q is an ancestor of v and the change of the aggregate value from q to v is significant. For example, given that the average house price in Vancouver is $1.1 million, one can find all regions of Vancouver where the average house price is 20% higher or lower than $1.1 million.

Cubegrade is a notion introduced by Imielinski et al. (2002) which reports how characteristics of a data cube cell is associated with the characteristics of its gradient cells; namely, ancestors (by roll-up), descendants (by drill-down) and siblings (by mutation). Cubegrade is a generalization of association rules and data cubes; though, Cubegrade queries are significantly more expressive than association rules since they can handle arbitrary measures and not just count as with the association rules. However, Cubegrade needs to compare each cell in the cube with its associated cells (i.e. gradients) generated by generalization (roll-up), specialization (drill-down) and mutation and even with iceberg cubes, it may generate a large number of pairs. To address this issue, Dong et al. (2005) introduces "probe constraints" to select a subset of cells, referred to as probe cells, from all possible cells. This is based on the pragmatic observation that analysts are often interested in examining only a small set of cells in the cube and a majority of the cells in the cube are outside their focal areas of exploration. Using this constraint, the analysis can centre only on probe cells and the relationships with their gradients (gradient constraint). Wang et al. (2006) applies these constraints to closed itemset mining.

When a computaion of an iceberg cube is confined with simple measures, such as, count and sum, antimonotonicity property of the cube can be exploited to prune a significant number of cells. For example, if the count of a cell c in a cuboid C is less than a threshold value v, then the count of any c's descendant cells cannot be more than v and thus all descendants of c ca be pruned. When the measure is not an antimonotonic function, for example, average or sume of positive and negative elements, a weaker but still antimonotonic property of top-k average (where k is the minimum support) can be considered to prune search space effectively (Dong et al., 2005; Yu et al., 2005; Wang et al., 2006).

2.3. Outlier Detection

The primary objective of benchmarking is to find areas for continuous improvement; thus, organizations are interested in identifying performance areas in which they are anomalies (or outliers) as opposed to "normal". As such, outlier analysis in multidimensional subspaces lends itself to viable multidimensional benchmarking. To this end, we explore the application of outlier detection techniques to multidimensional benchmarking.

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." (Hawkins, 1980). Data may be generated by a process that reflects normal activities of an underlying system; however, when the process behaves in an unusual manner, it results in the generation of anomalous data, or outliers. Outliers hence convey useful information regarding different characteristics of the process of the underlying system.

Outlier detection methods can be categorized according to whether or not prior knowledge is available to model normality and abnormality. Prior knowledge typically consists of samples that are tagged as normal or abnormal by subject matter experts. If prior knowledge is available, the detection approach is analogous to supervised classification. If not, the detection approach is essentially unsupervised clustering (Hodge et al., 2004). Semi-supervised outlier detection methods can be regarded as applications of semi-supervised learning approach where the normal class is taught but the algorithm learns to recognize abnormality.

The challenge to the supervised approach is that the outlier population is often much smaller than the normal population; thus, an additional consideration for handling imbalanced data must be taken into account and techniques, such as, oversampling and artificial outliers have been devised (Weiss et al., 1998; Joshi et al., 2001, 2002; Vilalta et al., 2002; Phua et al., 2004).

The unsupervised outlier detection approach makes certain assumptions and based on the assumptions made, the approach can be categorized into 3 types including statistical, proximity, and clustering.

The statistical methods assume a probability model of normal data and the data points that have a low probability of having been generated by the same model are considered outliers.

The proximity-based methods (Ester et al., 1996; Knorr et al., 1998) assume that the data points that have many proximate neighbours are normal while points that are far away from their neighbours are outliers.

The clustering-based methods assume that normal data points belong to large and dense clusters while outliers belong to small or sparse clusters, or do not belong to any cluster at all. More recent methods (Eskin et al., 2002; He et al., 2003; Zhang et al., 2007) incorporate ideas to handle outliers without explicitly and completely finding clusters of normal objects.

2.4. Subspace Analysis

While most outlier detection methods focus on finding outliers, recently there have been studies on “properties” of outliers by identifying subspaces in a multidimensional database where outliers may exist.

For example, Keller et al. (2012) and Böhm et al. (2013) proposed statistical approaches, CMI and HiCS, to select subspaces in which there may exist outliers. They select highly contrasting subspaces for all possible outliers.

Kriegel et al. (2009) introduced SOD, a method to detect outliers in axis-parallel subspaces. It uses the nearest neighbours as references in the full space to calculate outlying scores.

Müller et al. (2012b) presented a framework, OutRules, to find outliers in different contexts. For each outlier, OutRules finds a set of rules $A \rightarrow B$ where A and B are subspaces and an object is normal in A but is an outlier in B. It computes the degree of deviation using outlier scores, such as, LOF (Breunig et al., 2000) and produces a ranked list of rules as the explanation for objects being outliers.

Tang et al. (2013) proposed a framework to identify contextual outliers in a multidimensional categorical database.

Müller et al. (2012a) computes an outlier score for each object in a database providing a single global measure of how outlying an object is in different subspaces.

Given a multidimensional categorical database and an object (which preferably is an outlier), Angiulli et al. (2009) finds top-k subspaces from which the outlier receives the highest outlier scores. The score for a given object in a subspace is calculated based on how frequent the object's value appears in the subspace. It tries to find subspaces E and S such that the value is frequent in one subspace and much less frequent in the other. Since searching all such rules incurs a significant computational cost, it takes two parameters, δ and θ, to constrain the frequency of the given object's values in subspaces E and S. If an object is not an outlier, no outlier properties may be detected.

There are a few density based outlier detection methods in subspaces, such as, Breunig et al. (2000), Aggarwal et al. (2001), He et al. (2005), and Kriegel et al. (2008). Müller et al. (2011) proposed OUTRES which aims to assess the contribution of selected subspaces in which an object deviates from its neighbours (i.e. the object has a significantly low density). OUTRES employs Kernel Density Estimation (KDE) and uses

Epanechnikov kernel. The focus of OUTRES is to find outliers; thus, it only considers subspaces that satisfy a statistical test for non-uniformity.

To some extent, outlying property or outlyingness is related to uniqueness and uniqueness mining. Paravastu et al. (2008) finds feature-value pairs that make a particular object unique. Their task formulation is reminiscent of infrequent itemset mining and uses a level-wise Apriori enumeration strategy (Agrawal et al., 1994).

Chapter 3.

Benchmarks in Data Warehouses

Many organizations store multidimensional data in their enterprise data warehouses. A data warehouse thus provides an essential information infrastructure for online analytical queries that simultaneously consider multiple dimensions of data. However, to the best of the author's knowledge, conducting multidimensional benchmarking in data warehouses has not been explored from a technical efficiency perspective. A simple application of existing data warehouse techniques, such as, data cubes, cannot answer multidimensional benchmarking queries online due to the exponential growth in computational power required as the number of dimensions increases.

In this chapter, as the first step towards multidimensional benchmarking, we formulate benchmark queries technically and develop two algorithmic approaches for efficient computation in data warehouses; Sorted Inverted Index Cube (SIIC) and Dominant Answer Materialization (DAM).

3.1. Preliminaries

We largely follow the notations in the conventional data cube and data warehouse literature (Gray et al., 1997).

Consider a relational table $T = (TID, A_1, \dots, A_n, M)$ and an aggregate function f where TID is a tuple-id attribute to ensure every tuple in the table is unique, $A_1, \dots, A_n$ are dimensions and M is a measure attribute. We assume all dimension attributes are categorical and the measure attribute can be categorical or numeric. For a tuple $t \in T$, the value of t for TID is denoted as $t.TID$, the value for dimension A_i as $t.A_i$, and the value for measure M as $t.M$.

Let $D = \{A_{i_1}, \ldots, A_{i_l}\}$ be a subset of dimensions where $1 \le i_1 < i_2 < \cdots < i_l \le n$. D is often referred to as a **subspace**. A **cuboid** of D is the group-by's using attributes in D and denoted by C_D. Note that D may be empty. An aggregate cell in C_D is a tuple $c = (*, a_{i_1}, *, a_{i_2}, \ldots, *, a_{i_l}, *, aggr) \in C_D$ where value a_{i_l} belongs to the domain of attribute $A_{i_j} (1 \le j \le l)$, meta-symbol $*$ indicates that the dimension is generalized, and $aggr = f(\{t.M | t.A_{i_j} = a_{i_j}, 1 \le j \le l\})$ is the aggregate of all tuples in the group $(*, a_{i_1}, *, a_{i_2}, \ldots, *, a_{i_l}, *)$. To simplify presentation, we overload the symbol $c.M = aggr$ and ignore empty aggregate cells that do not contain any tuple in the base table.

We can define a partial order $\prec$ on cuboids: $C_{D_1} \prec C_{D_2}$ if $D_1 \subset D_2$. The set of cuboids form a lattice with respect to partial order $\prec$. Further, we can define a partial order $\prec$ on aggregate cells. The cells $t_1 \prec t_2$ if for each dimension $A_i (1 \le i \le n)$ when $t_1.A_i \ne *$, then $t_1.A_i = t_2.A_i$. This also means that t_1 is an ancestor of t_2 and t_2 a descendant of t_1. For two aggregate cells t_1 and t_2, t_1 is a sibling of t_2 if t_1 and t_2 have identical values for all dimensions except for one in which neither has value $*$.

A **data cube** is a set of cuboids for all subsets of dimensions including the empty set. Equivalently, a data cube is a set of all aggregate cells. For two aggregate cells u and v, if there does not exist a dimension A_i such that neither $u.A_i$ nor $v.A_i$ has value $*$ and $u.A_i \ne v.A_i$, then the **concatenation** of u and v, denoted by $w = u \otimes v$, is an aggregate cell such that for attribute $A_i, w.A_i = u.A_i$ if $u.A_i \ne *$; otherwise, $w.A_i = v.A_i$.

Example 1 (Preliminaries). Consider a relational table T = {TID, age-group, gender, location, salary} for sales representatives of an organization. Suppose we use $avg()$ as the aggregate function. c = ($*$, male, $*$, $avg()$) is an aggregate cell which represents the average salary of all male sales representatives in the organization. Consider aggregate cells u = (senior, $*$, $*$), t = (senior, male, $*$) and t' = (senior, female, $*$). We have $u \prec t$ which means u is an ancestor of t and t is a descendant of u. t and t' are siblings. Aggregate cell v = ($*$, male, North America) means male sales representatives in North America. We can use the concatenation operator to get all senior male sales representatives in North America: $w = u \otimes v$ = (senior, male, North America).

3.2. Benchmark Queries

We consider a relational table $T = (TID, A_1, \dots, A_n, M)$. The attributes in T that will be used in a benchmark query can be divided into three groups: unit-id attributes UID, dimension attributes DIM, and measure attributes M where $UID \cup DIM \subseteq \{A_1, \dots, A_n\}$. $UID \cap DIM = \emptyset$ is not assumed; thus, UID and DIM are not exclusive. This means that an attribute may be both a unit-id and a dimension attribute. However, we can always create a copy of an attribute that can be used as either unit-id or dimension attribute; as such, without loss of generality, we assume that the unit-id and dimension attributes are exclusive for the rest of this chapter.

The unit-id attributes are used to group tuples in T into aggregate units. Since the term "group" can mean different things, we refer to a group as a **unit** for clarification. We are essentially considering a data cube formed using the unit-id attributes UID in which each aggregate cell corresponds to a unit. In the context of benchmarking, we are interested in comparing units (e.g. organizations, departments, etc.).

The dimension attributes are used to conduct multidimensional comparative analysis between two units.

The measure attribute is used to calculate aggregates and derive quantitative difference between two units, referred to as "performance gap". We are interested in finding benchmarks that yield largest performance gaps to the query unit. For simplicity, we only have one measure attribute in this thesis; however, our method can be extended to scenarios where multiple measure attributes are considered to derive sophisticated aggregates. In practice, a measure attribute has non-negative values; for example, measures, such as, sum, volume, and amount are used in business intelligence applications. Even when a measure has negative values, we can normalize the attribute such that the normalized measure attribute has non-negative values.

For each non-empty unit that consists of at least one tuple in the base table with dimension and measure attributes, we can form a data cube which reflects performance of the unit in multidimensional aspects.

Example 2 (Attributes). Consider a base table $T =$ {age-group, gender, location, position, education, salary} for employees of an organization. For simplicity, we omit the tuple-id attribute.

We can use $UID =${age-group, gender} as unit-id attributes; that is, we are interested in comparing units formed by the group-by operation by these two attributes. For example, (young, male) and (mid-age, $*$) are two aggregate units.

We use attributes $DIM =${location, position, education} as the dimension attributes; that is, we compare two units by these three dimensions.

Finally, we use attribute $M =$ {salary} as the measure attribute. Using the aggregate function $avg()$, we can compare the average salaries between different units with respect to different locations, positions, education levels, and their combinations. For example, we may find that for the position of "technical support" at location "Vancouver", the age-group [25, 35] has much lower average salary than the age group [35, 50]. The reasoning behind this difference may be seniority and the years of experience.

To quantitatively compare two aggregate cells c and c', we need the ratio of their measures: $\frac{c.M}{c'.M}$. For a unit u, an aggregate cell c is defined using the dimension attributes and is called an **aspect** of u if $u \otimes c$ is in the data cube $Cube(B, UID \cup DIM, M, f)$ for the base table B. Given two units u and v defined using the unit-id attributes and an aggregate cell c defined by the dimension attributes such that c is an aspect of both u and v, $\frac{(u \otimes c).M}{(v \otimes c).M}$ indicates the performance gap between u and v in aspect c. The larger the ratio, the larger the performance gap between u and v in c. We denote by $R\left(\frac{u}{v}\middle|c\right) = \frac{(u \otimes c).M}{(v \otimes c).M}$ the **performance gap** of u against v.

From this example, we define a benchmark query as follows:

- a base table T and the specification of the unit-id attributes UID, dimensions DIM, and the measure M;
- a query unit q that is an aggregate cell in the data cube formed by the unit-id attributes UID;
- the search scope; that is, ancestors, descendants, and siblings; and

- a parameter k.

Let u be a unit formed by the unit-id attributes and c be an aspect of the query unit q. (u, c) is a top-k answer to the benchmark query Q if:

- u is in the search scope; that is, an ancestor, descendant, or a sibling of $q.UID$ as specified in the input;
- $\frac{(u \otimes c).M}{(q \otimes c).M} > 1$; and
- there are at most $k - 1$ pairs (u', c') such that u' is also in the search scope, $c' \neq c$ is another aspect of u and $\frac{(u' \otimes c').M}{(q' \otimes c').M} > \frac{(u \otimes c).M}{(q \otimes c).M}$.

The requirement $\frac{(u \otimes c).M}{(q \otimes c).M} > 1$ ensures that the performance gap is not trivial and u is a significant benchmark for q. To this end, we ignore aggregate cells c where $q \otimes c$ is empty because it is uninteresting from a benchmark perspective. For each (u, c) in the top-k answers, u is called a benchmark unit and the subspace c is the benchmark aspect of u. Given a benchmark query Q, we want to compute all top-k answers to the query. Note that in the event where there are multiple answers (i.e. the same performance gap), we return more than k answers.

Example 3 (Benchmark query). Consider a base table $T =$ {age-group, gender, location, position, education, salary} for employees of an organization. Table 2 shows samples of T.

Table 2 Sales Representatives of an Organization

age-group	gender	location	position	education	sales volume
young	M	Vancouver	staff	University	200
young	F	Seattle	manager	Diploma	230
young	F	Seattle	manager	University	220
mid-age	M	Vancouver	staff	Diploma	220
mid-age	M	Seattle	staff	University	200
mid-age	M	Vancouver	manager	University	224

Let $UID =$ {age-group, gender}, $DIM =$ {location, position, education}, and $M =$ {sales volume}. We use $avg()$ as the aggregate function.

Suppose the query unit is $q =$ (young, M) and $k = 2$. The top-2 answers are ((young, F), ($*, *, *$)) and ((mid-age, M), (Vancouver, $*$, University)). The ratio is $\frac{225}{200} = 1.125$ for the first and $\frac{224}{200} = 1.12$ for the second answer. This simple example illustrates that when we consider the sales performance of a group of young males, the two most significant benchmarks (i.e. have the largest performance gaps) for this unit are young females in all aspects, and the mid-age males who work in Vancouver and are university graduates.

Aggregate functions can be categorized into two types: monotonic and non-monotonic aggregates. An aggregate function f is monotonic if for any aggregate cells c_1 and c_2 such that $c_1 \prec c_2, f(c_1) \leq f(c_2)$. An aggregate function is non-monotonic if it does not have this property. For example, if the measure attribute only has non-negative values, then aggregate functions $sum()$ and $count()$ are monotonic while $avg()$ is non-monotonic.

Answering a benchmark query for a monotonic aggregate function is straightforward since the apex cell $(*,*,\ldots,*)$ always has the maximum aggregate value but uninteresting for benchmarking. As such, we assume that the aggregate functions used for benchmarking are non-monotonic.

3.3. Sorted Inverted Index Cube (SIIC) Method

We assume a data cube materialization method $Cube(B, A_1, \ldots, A_n, M, f)$ that computes a data cube on a multidimensional base table B using attributes as dimensions $A_1, \ldots, A_n$, M as the measure and the aggregate function f.

We use BUC (Beyer et al., 1999) to materialize a data cube. For each unit u, let B_u be the set of tuples in the base table that belong to u; that is, $B_u = \{t | t \in B \ \wedge u \preceq t\}$. Given a query unit q, a benchmark query compares the data cubes $Cube(B_q, DIM, M, f)$ and $Cube(B_u, DIM, M, f)$ for every unit u in the search scope. This is equivalent to materializing the whole data cube $Cube(B, UID \cup DIM, M, f)$ since all units using attributes UID need to be considered.

A naïve method is to search every unit u in the scope, given a query unit q, and compute the performance gap between q and u for every possible aggregate cell c formed by the set of attributes DIM. It is time consuming to perform computation in every aspect c for every unit u. We can organize the units and the aspects such that the search can ignore many aggregate cells that are trivial.

3.3.1. Inverted Index for Fast Search

In this section, we use two simple ideas to facilitate fast search.

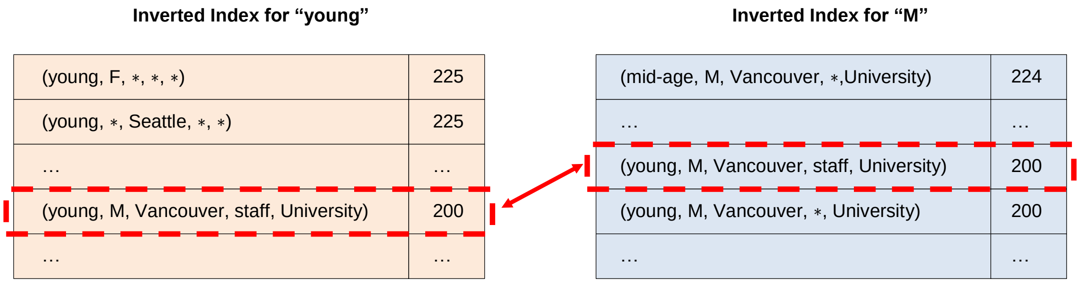
As the first idea, we sort all aggregate cells in the cube $Cube(B, UID \cup DIM, M, f)$ in the aggregate value descending order. We search aggregate cells in this order for answering a query. In this order, we visit the aggregate cells of larger values earlier on and thus heuristically we have a better chance of finding cells with larger performance gaps for the query cell. Let $\prec_{aggr}$ be the aggregate value descending order of all aggregate cells in $Cube(B, UID \cup DIM, M, f)$. For any aggregate cells u and v, if $u \prec_{aggr} v$, then $u.M \geq v.M$. Note that if there are two or more aggregate cells having the same value, the tie can be broken in any arbitrary way.

The second idea is to use inverted index (Wikipedia, 2016). For each value in the domain of every unit-id attribute, we maintain an inverted index to record the list of aggregate cells containing this value. Suppose a_{ij} is a value in the domain of unit-id attribute A_i. The inverted index $Index_{a_{ij}}$ is a list of aggregate cells $u \in Cube(B, UID \cup DIM, M, f)$ such that $u.A_i = a_{ij}$. All aggregate cells in every inverted index are sorted according to the order $\prec_{aggr}$.

We can retrieve all aggregate cells of cube $Cube(B_q, DIM, M, f)$ using inverted indices efficiently in a way similar to merge-sort. Let q be the query unit and $q.A_{i_1}, \dots, q.A_{i_l}$ are the unit-id attribute values that are not $*$. To find all aggregate cells of q, we only need to search inverted indices $Index_{q.A_{i_1}}, \dots, Index_{q.A_{i_l}}$ and find all aggregate cells c such that c appears in every $Index_{q.A_{i_j}}$ and takes value $*$ in all other unit-id attributes. Since we scan the inverted indices in the order of $\prec_{aggr}$, we can find all aggregate cells of q in one scan.

The inverted index also facilitates efficient retrieval of all unit aggregate cells in the search scope; that is, ancestors, descendants, and siblings of q. To search for the ancestor units and their aggregate cells, we scan the inverted indices $Index_{q.A_{i_1}}, \dots,$ $Index_{q.A_{i_l}}$ in a synchronized manner. Except for the unit $(*, \dots, *)$ which can be checked separately as a special case, an aggregate cell c is an ancestor of q if (1) c appears in at least one of the inverted indices $Index_{q.A_{i_1}}, \dots, Index_{q.A_{i_l}}$; and (2) $c.A_{i_j} = *$ if c does not appear in $Index_{q.A_{i_j}}$. Again, since we scan the inverted indices in the order of $\prec_{aggr}$, we can find all ancestor units of q and their aggregate cells in one scan. To find all descendant units of q, we search the inverted indices $Index_{q.A_{i_1}}, \dots, Index_{q.A_{i_l}}$ and find all cells c such that c appears in every inverted index $Index_{q.A_{i_j}}$ and takes a non-$*$ value in at least one unit-id attribute other than $A_{i_1}, \dots, A_{i_l}$. To find all siblings of q, we search the inverted indices $Index_{q.A_{i_1}}, \dots, Index_{q.A_{i_l}}$ and find all cells c such that (1) c appears in every inverted index $Index_{q.A_{i_j}}$ except for one, say $Index_{q.A_{i_{j0}}}$; (2) $c.A_{i_{j0}} \neq q.A_{i_{j0}}$ and $c.A_{i_{j0}} \neq *$; (3) $c.A_{i_j} = *$ if $q.A_{i_j} = *$. Both searches can be achieved in one scan of the inverted indices.

Example 4 (SIIC). We use sample data shown in Table 2 as an example. The $avg()$ is the aggregate function. Let $UID =$ {age-group, gender}, $DIM =$ {location, position, education}, and $M =$ {sales volume}. We first build an inverted index for each value in the domain of every unit-it attributes, in this example, age-group and gender as shown in Figure 7.



Inverted Index for “young”

(young, F, $*$, $*$, $*$)	225
(young, $*$, Seattle, $*$, $*$)	225
…	…
(young, M, Vancouver, staff, University)	200
…	…

Inverted Index for “M”

(mid-age, M, Vancouver, $*$,University)	224
…	…
(young, M, Vancouver, staff, University)	200
(young, M, Vancouver, $*$, University)	200
…	…

Figure 7 Example SIIC for values "young" and "M"

Suppose the query unit is $q =$ (young, M). We find that (young, M, Vancouver, staff, University) appears in both inverted indices for “young” and “M”; thus, (young, M, Vancouver, staff, University) must be an aggregate cell of q. Similarly, we can easily find all aggregate cells of q, {(young, M, Vancouver, staff, University):200, (young, M, Vancouver, $*$, University):200, …} with the aide of inverted indices.

To find all aggregate cells of ancestors, descendants, and siblings of q, we apply the same technique to the search scope of q. For example, to find all aggregate cells of a sibling (young, F), we only need to check aggregate cells that appear in both inverted indices for “young” and “F”. Again, we have all aggregate cells sorted in the search scope {(young, F, $*$, $*$, $*$):225, (mid-age, M, Vancouver, $*$, University):224, …}.

3.3.2. Pruning

Since we scan aggregate cells in the aggregate value descending order, we maintain the top-k answers and we can define the following property.

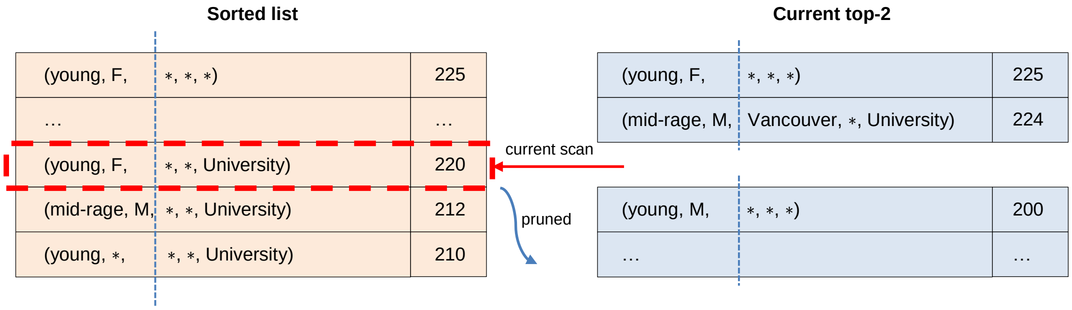
Lemma 1. Given a query unit q, consider an aggregate cell c for q‘s dimension attributes such that $q \otimes c$ is not empty. For two units u and u’ such that $u \otimes c \prec_{aggr} u' \otimes c$, then $\frac{(u \otimes c).M}{(q \otimes c).M} \geq \frac{u' \otimes c.M}{q \otimes c.M}$.

Proof. We only need to recall that if $u \otimes c \prec_{aggr} u' \otimes c, (u \otimes c).M \geq (u' \otimes c).M$, and the assumption that the aggregate values are positive.

Using Lemma 1, for any aggregate cell c for dimension attributes such that c is an aspect of q (i.e. $q \otimes c$ is not empty), if we scan an aggregate cell $v = u \otimes c$ such that (u, c) is not qualified to be a top-k answer among the aggregate cells processed so far, then any pair (u', c) to be scanned later is not qualified either; thus, c can be pruned.

Further, let v be the current aggregate cell we scan in the inverted indices. For any aspect c of q, if $\frac{v.M}{(q \otimes c).M}$ is less than the top-k answers we have seen so far, then no aggregate cells after v in the sorted list can form a pair (u, c) such that $v = u \otimes c$ and (u, c) is qualified as a top-k answer. In this case, the aspect c can be pruned as well. This rule applies to all aspects of q; that is, aggregate cells in cube $Cube(B_q,\ DIM, M, f)$. Once it is determined that all aspects of q are processed (i.e. either included in the current top-k answer or can be pruned), the search can terminate and the current top-k answers can be returned as the final answers to the benchmark query.

Example 5 (SIIC with Pruning). Figure 8 illustrates pruning with Example 4. Suppose the query unit is $q =$ (young, M) and we want to find top-2 units that give largest performance gaps. Assume we have {(young, F, $*$, $*$, $*$): 225, (mid-age, M, Vancouver, $*$, University): 224} as current top-2 benchmarks as shown in the figure. We are currently scanning the aggregate cell {(young, F, $*$, $*$, University): 220}. It is easy to see that {(young, F, $*$, $*$, University): 220} does not qualify to be top-2; as such, all following cells that are compatible with ($*$, $*$, University) can be pruned.



(young, F,	∗, ∗, ∗)	225
…		…
(young, F,	∗, ∗, University)	220
(mid-rage, M,	∗, ∗, University)	212
(young, ∗,	∗, ∗, University)	210

(young, F,	∗, ∗, ∗)	225
(mid-rage, M,	Vancouver, ∗, University)	224
(young, M,	∗, ∗, ∗)	200
…		…

Figure 8 Example SIIC with Pruning

3.4. Dominant Answer Materialization (DAM) Method

The SIIC method has a severe drawback; in the worst case, it still has to go through the list of all aggregate cells of the whole data cube $Cube(B, UID \cup DIM, M, f)$. When the data cube is large, the cost is significant in both space and time. The Dominant Answer Materialization (DAM) method addresses this issue.

3.4.1. Search Scope of Ancestors

We first consider the search scope of ancestors. Consider a query unit q and a unit u that is an ancestor of q; that is, $u \prec q$. u is called a maximal unit of q with respect to aspect c if c is an aspect of both q and u and there does not exist another ancestor u' of q such that $\frac{(u'\otimes c).M}{(q\otimes c).M} > \frac{(u\otimes c).M}{(q\otimes c).M}$. The following is observed.

Theorem 1 (Monotonicity). Given a unit q, if a unit u is a maximal unit of q with respect to aspect c, then for any unit q' such that $u \prec q' \prec q$, u is also a maximal unit of q' with respect to c.

Proof by contradiction. Assume that u is not a maximal unit of q' with respect to c. Then there exists another unit u' such that $u' \prec q'$ and $\frac{(u'\otimes c).M}{(q'\otimes c).M} > \frac{(u\otimes c).M}{(q'\otimes c).M}$. Since $u' \prec q'$ and $q' \prec q$, we have $u' \prec q$. Since $u' \prec q'$ and $\frac{(u'\otimes c).M}{(q'\otimes c).M} > \frac{(u\otimes c).M}{(q'\otimes c).M}$ and the measure values are non-negative, we have $(u' \otimes c).M > (u \otimes c).M$. Therefore, we have $\frac{(u'\otimes c).M}{(q\otimes c).M} > \frac{(u\otimes c).M}{(q\otimes c).M}$. That is, u is not a maximal unit of q with respect to c. A contradiction.

Theorem 1 presents a useful observation; that is, multiple query units may share a common aggregate unit as an answer to benchmark queries. To answer benchmark queries efficiently, we can pre-compute aggregate units and the associated aspects that may be answers to benchmark queries. The problem then is to determine, for an aggregate unit u, which query units may take u as a possible answer and with respect to which aspects. The following lemma answers this question.

Lemma 2. For aggregate units u and v such that $u \prec v$, let c be an aspect of both u and v. Then, u is not a maximal unit of v with respect to c if:

1. there exists an ancestor $u' \prec u$ such that $(u' \otimes c).M > (u \otimes c).M$; or
2. there exists a descendant u" such that $u \prec u'' \prec v$ and $(u \otimes c).M < (u'' \otimes c).M$.

Proof. If there exists an ancestor $u' \prec u$ such that $(u' \otimes c).M > (u \otimes c).M$, then $R\left(\frac{u'}{v}\middle|c\right) > R\left(\frac{u}{v}\middle|c\right)$. If there exists a descendant u" such that $u \prec u'' \prec v$ and $(u \otimes c).M < (u'' \otimes c).M,$ then $R\left(\frac{u''}{v}\middle|c\right) > R\left(\frac{u}{v}\middle|c\right)$. In both cases, u is not a maximal unit of v with respect to c.

According to the first item in Lemma 2, to answer benchmark queries whose search scope is the ancestors, we do not need to store the whole data cube $Cube(B, UID \cup DIM, M, f)$. Instead, we only need to store those aggregate units u and aspects c such that there does not exist another unit u' and aspects c' and $(u \otimes c).M < (u' \otimes c).M$. In other words, we only need to store units and aspects whose measure values are not dominated by any of their ancestors.

For aggregate unit u and aspect c, we call (u, c) a **dominant answer** if there does not exist another unit u' and c' and $(u \otimes c).M < (u' \otimes c).M$. To answer any benchmark query, we only need to materialize all dominant answers.

Once all dominant answers are materialized, we can organize dominant answers using inverted indices as per the SIIC method.

The last problem is to find how to compute dominant answers. A brute-force is to compute a full data cube and then select dominant answers from all aggregate cells. Since we are concerned with groups of aggregate cells with different measure values, we can adopt the quotient cube method (Lakshmanan et al., 2002).

Instead of computing all aggregate cells of a data cube, the quotient cube method groups aggregate cells according to the tuples in the base table that contribute most to the aggregate of cells. For an aggregate cell u, it considers the set of descendant tuples in the base table $cov(u) = \{t|u \prec t, t \in B\}$. If two aggregate cells u_1 and u_2 share the identical set of descendant tuples in the base table; that is, $cov(u_1) = cov(u_2)$, then the two cells are assigned to the same quotient group. It shows that each quotient group has a unique upper bound which is also in the same group. In other words, if there are u_1 and u_2 such that $cov(u_1) = cov(u_2)$ but $u_1 \nprec u_2$ and $u_2 \nprec u_1$ then there exists another aggregate cell u such that $u \nprec u_1, u \nprec u_2$ and $cov(u) = cov(u_1) = cov(u_2)$. Now we only need to materialize the upper bounds of the quotient groups that are dominant answers.

Example 6 (DAM). Using Table 2, we assume that the query unit is $q =$ (young, M) and use $avg()$ as the aggregate function. The set of ancestors of the query unit is {($*$, M), (young, $*$), ($*$, $*$)}. It is easy to verify that $u =$ (*, M) is a maximal unit of q with respect to aspect $c =$ (Vancouver, $*$, University) and $u =$ ($*$, $*$) is a maximal unit of q with respect to aspect $c =$ (Vancouver, staff, $*$). According to the base table, ((mid-age, M), (Vancouver, $*$, University)) is a dominant answer since there does not exist a unit u' that has a greater aggregate value than avg(mid-age, M) $\otimes$ (Vancouver, $*$, University)). As an example of quotient cube, we can verify that cov(mid-age, M, Vancouver, manager, $*$) $=$ cov(mid-age, M, $*$, manager, University); that is, they have the same set of descendants in the base table. Thus, these two aggregate cells are in the same quotient group. Further, (mid-age, M, $*$, manager, $*$) is the upper bound of the group. By using the quotient cube, we can materialize all dominant answers from the quotient group in Table 2; that is, {(young, F, $*$, $*$, $*$), (mid-age, M, Vancouver, $*$, University), …}. Unlike the SIIC method, we only store the dominant answers, reducing both the search space and the time. Once a query is given, we can use the inverted indices to answer the query efficiently.

3.4.2. Search Scope of Descendants

Consider a query unit q and a unit u that is a descendant of q; that is, $u \succ q$. Then u is called a maximal unit of q with respect to aspect c if c is an aspect of both q and u and there does not exist another descendant u' of q such that $\frac{(u' \otimes c).M}{(q \otimes c).M} > \frac{(u \otimes c).M}{(q \otimes c).M}$. Similar to Theorem 1, we have the following:

Corollary 1 (Monotonicity). Given a unit q, if a unit u is a maximal unit of q with respect to aspect c, then for any unit q' such that $u \succ q' \succ q$, u is also a maximal unit of q' with respect to c.

Also, similar to Lemma 2, we have the following:

Corollary 2. For aggregate units u and v such that $u \succ v$, let c be an aspect of both u and v. Then u is not a maximal unit of v with respect to c if:

1. there exists a descendant $u' \succ u$ such that $(u' \otimes c).M > (u \otimes c).M$; or
2. there exists an ancestor u'' such that $u \succ u'' \succ v$ and $(u \otimes c).M < (u'' \otimes c).M$.

3.5. Empirical Evaluation

The algorithms were implemented with Python 2.7 running with PyPy[2] JIT optimization. PyPy[2] is an advanced just-in-time complier which provides approximately 10 times faster running time and additional scalability than the standard Python. All experiments were conducted on a PC with an Intel Core i7-3770 3.40GHz CUP, 16GB memory and a 1 TB HDD, running the Ubuntu 14.04 operating system.

3.5.1. Data Sets and Experiment Settings

We evaluated our algorithms with both synthetic and real data:

- Synthetic data (TPC-H v2.17.1)

  TPC-H is a widely used data set that consists of a suite of business oriented ad-hoc queries and concurrent modifications. TPC-H has 8 separate individual base tables. We used the joined results of table PART and table LINEITEM as our evaluation base table.

- Real data (CDIAC Weather)

  The weather data set available from Carbon Dioxide Information Analysis Center (CDIAC) contains 1,015,367 tuples with attributes including station-id, longitude, latitude, solar-altitude, present-weather, day, hour, weather-change-code, and brightness.

The TPC Benchmark™H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance. This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions (TPC-H, 2016). Since benchmark queries and the underlying techniques are highly related to data warehousing and decision support, we chose TPC-H data sets.

We randomly generated 100 queries for each data set and conducted each experiment 10 times, reporting the average value. Using the $avg()$ as the aggregate function, we compared Sorted Inverted Index Cube without pruning (SIIC) as well as with pruning (SIICP), and Dominant Answer Materialization (DAM). We used BUC (Beyer et al., 1999) to materialize the cubes for SIIC/SIICP and Quotient Cube (Lakshmanan et al., 2002) to compute quotient groups for DAM.

3.5.2. Reduction of Aggregate Cells Computed and Indexed

We conducted two sets of experiments to evaluate the effectiveness of reducing the number of aggregate cells computed and indexed.

In the first set of experiments, we fixed the dimensionality of DIM and reported the number of computed and indexed cells with respect to the increase of dimensionality of UID. We sorted the dimensions according to the cardinalities in the descending order. For the TPC-H data set, we generated 9 testing data sets with 2 to 10 dimensions of UID. The dimensionality of DIM was fixed to 5. For the Weather data, we generated 4 testing data sets with 2 to 5 dimensions of UID. The dimensionality of DIM was fixed to 5. The results are shown in tables 3 and 4.

Table 3 TPC-H: number of computed and indexed cells (5 DIM dimensions)

Method	Dimensionality of UID	2	3	4	5	6	7	8	9	10
SIIC/SIICP	Computed (x10^6)	0.4	0.9	2.3	3.5	5.2	6.2	7.5	9.8	12
	Indexed (x10^5)	0.2	0.4	1.1	2.1	3.6	4.3	6.4	8.0	9.6
DAM	Computed (x10^5)	0.9	1.7	2.2	4.1	5.5	6.3	7.4	9.7	11
	Indexed (x10^4)	0.9	1.2	1.6	2.2	2.5	2.9	3.3	3.6	4.0

Table 4 Weather: number of computed and indexed cells (5 DIM dimensions)

Method	Dimensionality of UID	2	3	4	5
SIIC/SIICP	Computed (x10^5)	5.2	11	25	35
	Indexed (x10^4)	2.1	3.6	5.1	11
DAM	Computed (x10^5)	0.9	1.5	2.1	2.2
	Indexed (x10^3)	3.5	4.5	5.1	5.4

Since SIIC and SIICP have the same mechanism for materialization, the number of cells computed and indexed for these two methods is the same. DAM performs better than SIIC/SIICP for both TPC-H and Weather data. Figure 9 shows the reduction ratio of the computed and indexed cells for TPC-H where the reduction ratio is the number of cells in DAM over the number of cells in SIIC/SIICP. The reduction ratio in most cases is about 10%, meaning that DAM only computes and indexes about 10% of the cells that SIIC and SIICP do. Further, the ratio becomes smaller when the dimensionality of UID increases. This means that the more dimensions UID has, the more savings of materialization and indexing DAM achieves.

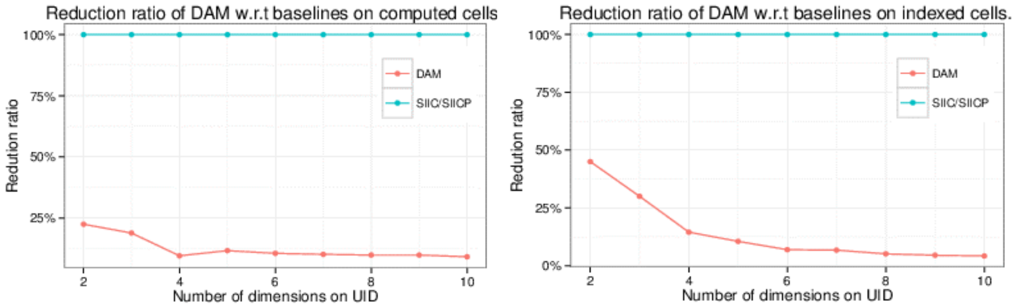


Figure 9 Reduction Ratio of DAM over SIIC/SIICP for TPC-H (DIM fixed)

Figure 10 shows the results for the Weather data set. The observation is similar to that for the TPC-H. The savings accomplished by DAM is due to the fact that DAM only stores and searches the dominant answers in the quotient groups.

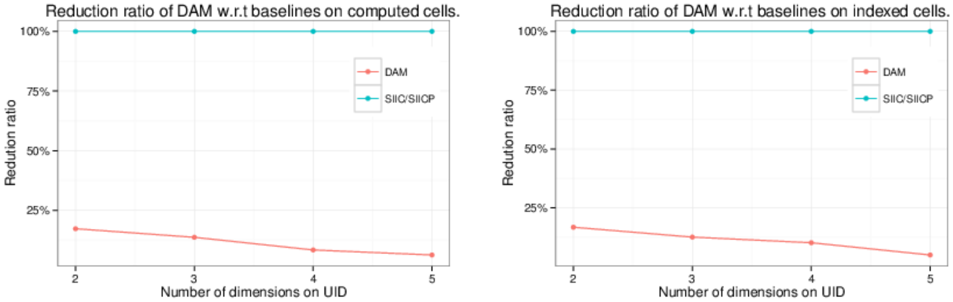


Figure 10 Reduction Ratio of DAM over SIIC/SIICP for Weather (DIM fixed)

In the second set of experiments, we fixed the dimensionality of UID and reported the number of computed and indexed cells with respect to DIM. For TPC-H, we generated 4 testing data sets with 2 to 5 dimensions of DIM and the dimensionality of UID was fixed to 10. For the Weather data, we generated 4 testing data sets with 2 to 5 dimensions of DIM and the dimensionality of UID was fixed to 5. The results are shown in tables 5 and 6.

Table 5 TPC-H: number of computed and indexed cells (10 UID dimensions)

Method	Dimensionality of UID	2	3	4	5
SIIC/SIICP	Computed (x10^6)	5.9	7.1	9.6	12
	Indexed (x10^5)	4.4	6.5	7.9	9.6
DAM	Computed (x10^5)	6.5	7.9	9.5	11
	Indexed (x10^4)	2.6	3.1	3.5	4.0

Table 6 Weather: number of computed and indexed cells (5 UID dimensions)

Method	Dimensionality of UID	2	3	4	5
SIIC/SIICP	Computed (x10^5)	4.2	9.8	22	35
	Indexed (x10^4)	1.5	2.6	5.9	11
DAM	Computed (x10^5)	0.8	1.4	1.9	2.2
	Indexed (x10^3)	3.2	4.2	4.8	5.4

Similar to the results of the first set of experiments, DAM significantly outperforms SIIC/SIICP for both synthetic and real data. Figure 11 shows the reduction ratio for TPC-H and Figure 12 the Weather data. The reduction ratio in most cases is about 10%. Similar to the observation described earlier, the ratio becomes smaller when the dimensionality of DIM increases.

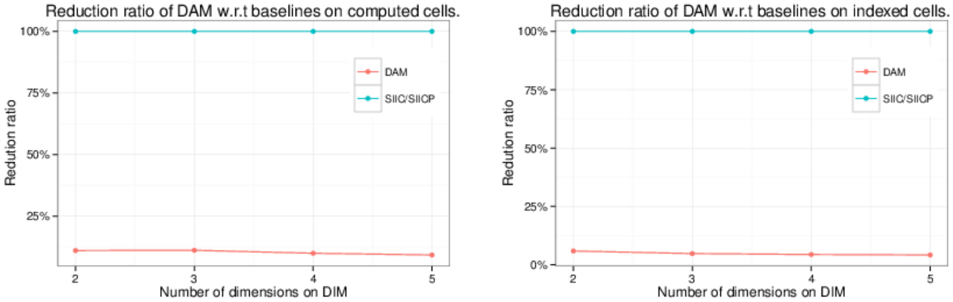


Figure 11 Reduction Ratio of DAM over SIIC/SIICP for TPC-H (UID fixed)

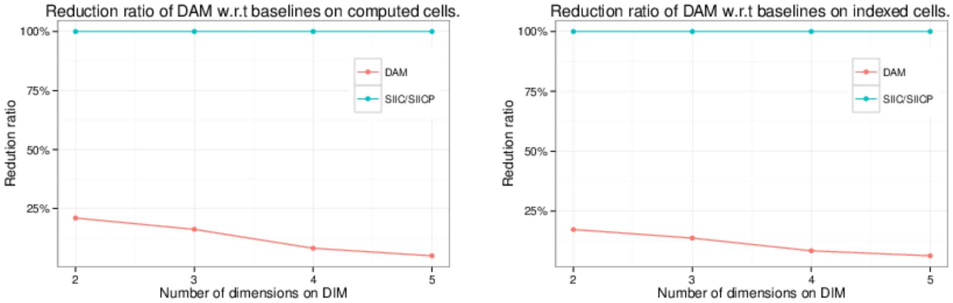


Figure 12 Reduction Ratio of DAM over SIIC/SIICP for Weather data (UID fixed)

3.5.3. Runtime and Memory Usage

We fixed the dimensionality of DIM and reported both runtime and memory usage in query answering with respect to the dimensionality of UID. The testing data sets were the same as those in the first set of experiments. The memory usage reported is the peak memory usage during query answering. Query answering was tested by 100 random queries, as such, the average query answering runtime and the standard deviation are reported. Figure 13 shows the results for TPC-H and Figure 14 the Weather data. SIICP is slightly faster and uses less memory than SIIC. DAM significantly outperforms SIIC and SIICP in runtime and memory usage. For runtime for index construction, DAM is less than half the runtime of SIIC/SIICP. Further, the DAM runtime increase is much slower than SIIC and SIICP with the UID dimensionality increase. DAM consumes a small amount of memory while SIIC/SIICP consume much larger amount of memory. Similar to runtime, the DAM memory usage increase is slower than SIIC/SIICP as the UID dimensionality increases.

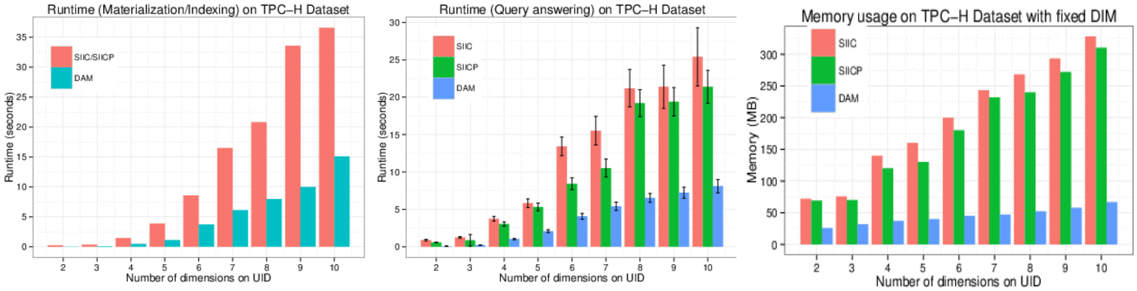


Figure 13 Runtime and Memory Usage with TPC-H (DIM fixed)

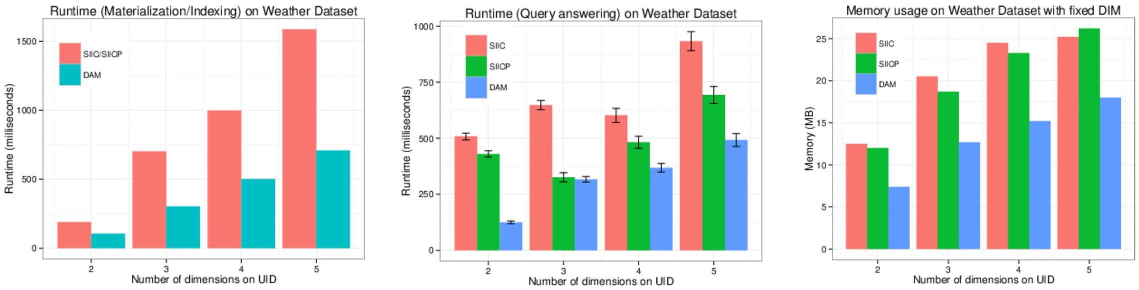


Figure 14 Runtime and Memory Usage with Weather (DIM fixed)

These results indicate that when UID has more dimensions, DAM can save more time and memory in query answering. As noted earlier, the savings come from the fact that DAM only computes and stores dominant answers in the quotient groups. Once a query is given, DAM only searches the dominant answers which leads to efficiency in both time and memory usage. Both SIIC and SIICP need to materialize the data cube using BUC and build the inverted indices. SIICP is faster than SIIC because it applies pruning in query answering.

Next, we fixed dimensionality of UID and reported the runtime and the memory usage with respect to the dimensionality of DIM, using the same testing data. The results are shown in figures 15 and 16. DAM clearly outperforms SIIC/SIICP.

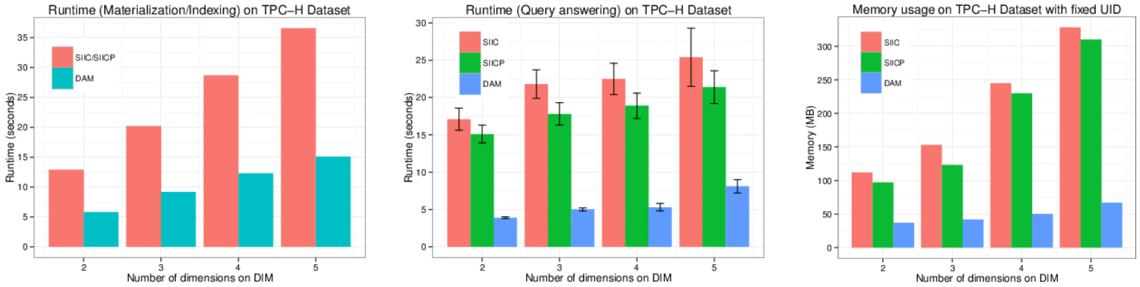


Figure 15 Runtime and Memory Usage with TPC-H (UID fixed)

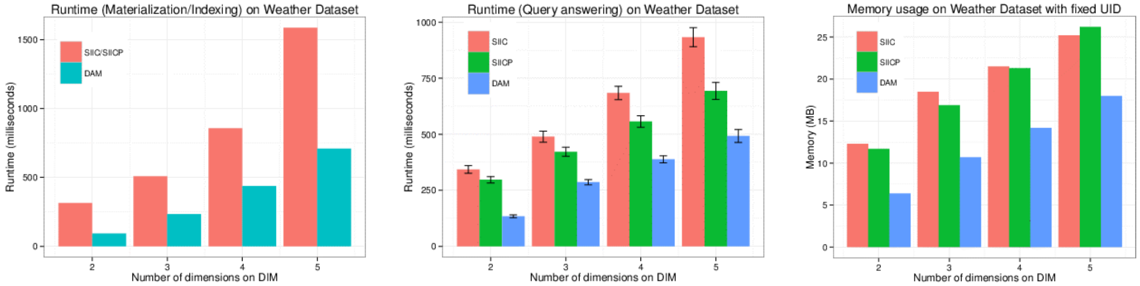


Figure 16 Runtime and Memory Usage with Weather (UID fixed)

3.5.4. Scalability

To assess scalability, we generated and used 4 TPC-H data sets with different size: 25%, 50%, 75%, 100% of 1GB. The dimensionality of UID was fixed to 10 and the dimensionality of DIM to 5. The results are shown in Figure 17. DAM is much more scalable than SIIC/SIICP for runtime. For memory usage, all 3 are scalable. DAM consistently uses much less memory than SIIC/SIICP.

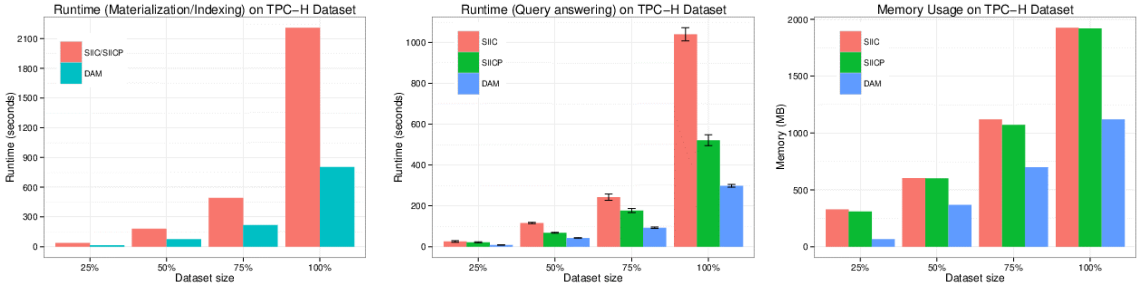


Figure 17 Scalability with TPC-H

Chapter 4.

Reflective Benchmarking

In Chapter 3, we developed methods to find significant benchmarks efficiently in data warehouses.

In this chapter, we consider outlier detection techniques in data warehouses for multidimensional benchmarking. As stated earlier in section 2.3, when organizations conduct benchmarking, they are mainly concerned with identifying areas for performance improvement; that is, areas where they perform exceptionally (positively or negatively). To this end, outlier analysis in multidimensional databases (i.e. data warehouses) lends itself to viable multidimensional benchmarking. By employing outlier detection techniques, we identify what causes organizations to deviate from the norm (i.e. benchmarks). Is an organization an outlier because a small number of underlying units or a majority of them are outliers?

Since what makes an organization an outlier is a reflection of self, we refer to the method presented in this chapter as “reflective benchmarking”.

4.1. Preliminaries

As in Chapter 3, we follow the notations in the conventional data cube and data warehouse literature (Gray et al., 1997).

Example 1 (Preliminaries). To develop a health index, an organization is interested in finding how prevalent certain illnesses are across its employees. While benchmarking, an organization may find that it has an exceptionally high rate of pain treatment provisions across its employees compared to other organizations in the same industry. The organization will be interested in finding what makes it an outlier and learning more about the characteristics of its internal structures.

Consider a table T = {employee-id, gender, age-group, service, count} which records the organization's employee attributes including unique identifiers of employees, genders, age-groups, services provided to treat certain illnesses, and the count of services provided. Suppose employees who suffer from chronic pain receive regular treatments, such as, physiotherapy, chiropractic, and massage services to ease pain. The prevalence rate of chronic pain across employees of the organization can be represented by how many employees of the organization provision such services. If the rate is an outlier, the organization would want to understand if it is an outlier because most of its employees are outliers. If so, it would then want to reflect on its internal structures, such as, the characteristics of its employees to understand what drives its employees to be outliers. The drivers could be genders, age-groups, types of occupations, employment types, departments, locations, number of dependents, etc. or combinations of them.

Definition 1 (Fact Table). We consider a fact table F for k parties (e.g. employees) whose attributes can be partitioned into $k+1$ subsets $F_i (1 \leq i \leq k+1)$. That is, $F = \bigcup_{i=1}^{k+1} F_i$.

- The subset $F_i (1 \leq i \leq k+1)$ contains the attributes of the $i\text{-}th$ party. We refer to F_i as $party\ i$.
- We assume that $\bigcup_{i=1}^{k} F_i$ is a key of the table F. That is, no two tuples have the same values of all attributes in $\bigcup_{i=1}^{k} F_i$.
- The subset F_{k+1} contains the measure (e.g. count). As in Chapter 3, the measure attribute is used to derive quantitative difference to indicate the performance gap between two parties.

Definition 2 (Groups). For the domain of each attribute $F = \bigcup_{i=1}^{k} F_i$, we introduce a meta-symbol $*$ which indicates that the attribute is generalized. A tuple t represents a **base level group** of $party\ i$, if t has a non-$*$ value for every attribute of F_i. Otherwise, t is 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 $F = \bigcup_{i=1}^{k} F_i$ where t_1 has a non-$*$ value, t_2 takes the same value as t_1. For example, (e12, male, 30-40, physiotherapy) is a base level group while ($*$,$*$, 30-40,$*$) is an aggregate group and ($*$,$*$,30-40,$*$) $\prec$ (e12, male, 30-40, physiotherapy).

It immediately follows that:

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

In theory, we can relax the requirement in Lemma 1; that is, as long as the domain of every attribute is either finite or countable, the lemma still holds. In practice, a fact table is always finite, as such, the domains of the attributes can be considered finite.

Definition 3 (Performance Gap). Given a fact table F of k parties, we extend the domain of each attribute in $\bigcup_{i=1}^{k} F_i$ such that meta-symbol $*$ is included as a special value. A performance gap is a group-by tuple $t \in F$. That is, for every attribute in $\bigcup_{i=1}^{k} F_i$, t takes either a value in the domain of the attribute or meta-symbol $*$. A performance gap is a base level gap if every party in t is a base level group. Otherwise, t is an aggregate performance gap.

If all groups are base level groups, a performance gap is simply a tuple in the fact table. When performance gaps contain some aggregate groups, we use an aggregate function to describe the aggregate performance gaps.

Definition 4 (**Measure of Performance Gap).** Given a fact table F of k parties, let $aggr: 2^{F_{k+1}} \rightarrow F_{k+1}$ be an aggregate function. For any aggregate performance gap t, the measure of t is the aggregate of the measures of all base level performance gaps that are descendants of t, that is,

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

For example, the prevalence rate of chronic pain across employees of an organization can be computed by taking the average of all counts of services provided to treat chronic pain in the fact table. Here the aggregate function is:

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

Theorem 1 (Performance Gap Lattice). Given a fact table F of k parties, if the domain of every attribute in $\bigcup_{i=1}^{k} F_i$ is finite, then all performance gaps form a lattice $L =$

$\prod_{i=1}^{k} L_{F_i}$ where L_{F_i} is the lattice of party F_i. Further, $|L| = |\prod_{i=1}^{k} L_{F_i}| = \prod_{A \in \cup_{i=1}^{k} F_i}(|A| + 1)$.

From Theorem 1, the size of the space required for performance gaps is exponential to the number of parties.

4.2. Outlier Types

Outliers can be modeled in many different ways (Campbell, 2014). In this section, we employ the model of statistical outliers which captures observation points that are distant from the majority of observations (Hodge et al., 2004). The rational for the model is that it is unlikely that distant points have been generated by the same mechanism that generated the majority of points. Given a set of samples where each sample is associated with a numerical measure, we can calculate the mean m and the standard deviation δ.

Theorem 2 (Chebyshev Inequality (Chebyshev, 1984)**).** Let X be a random variable with finite expected value m and non-zero variance δ. For any real number, $l > 0, P_r(|X - m| \geq l\delta) \leq \frac{1}{l^2}$.

We use l as an outlier threshold; the samples that are more than $l\delta$ away from m are deemed outliers.

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

Definition 5 can be easily extended to fact tables containing multiple measure attributes. Is there redundancy among performance gap outliers? We have the following observation.

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

Proof by contradiction. Without loss of generality, let $A \in F_1$. We can write $t = (t_1, t_2, \dots, t_k)$ and $t' = (t'_1, t_2, \dots, t_k)$ where $t_i (1 < i < k)$ are the groups from $party\ i$ and t'_1 is a group from $party$ 1 such that t'_1 is a child of t_1. Suppose t has n such children denoted by $t'^{(1)}, \dots, t'^{(n)}$. According to Definition 5, t is a performance gap outlier and $|t.F_{k+1} - m| > l\delta$. Suppose the children of t are $t_i(1 < i < n)$, then $t' \in t_i$. As per Definition 4 and using $average()$ as the aggregate function, we have $t.F_{k+1} = \frac{\sum_{i=1}^{n} t'^{(i)}.F_{k+1}}{n}$. Assume all possible performance $t'^{(i)}$ are not outliers. Then,

$$m - l\delta \le t'^{(i)}.F_{k+1} \le m + l\delta$$

$$n(m - l\delta) \le \sum_{i=1}^{n} t'^{(i)}.F_{k+1} \le n(m + l\delta)$$

$$m - l\delta \le \frac{\sum_{i=1}^{n} t'^{(i)}.F_{k+1}}{n} \le m + l\delta$$

$$\left| \frac{\sum_{i=1}^{n} t'^{(i)}.F_{k+1}}{n} - m \right| \le l\delta$$

$$|t - m| \le l\delta$$

A contradiction.

According to Theorem 3, if an aggregate performance gap t is an outlier, then some descendants of t must also be outliers. Consequently, we can classify performance gap outliers into two types:

- Aggregate performance gap t is a **type-I** outlier if most base level performance gaps of t are not outliers. In other words, a small number of descendants that are outliers are driving t to be an outlier. Thus, t being considered an outlier is a mere chance and may not be interesting; instead, outlying descendants of t could be more interesting and warrant further analyses.

- Aggregate performance gap t is a **type-II** outlier if many base level performance gaps of t are outliers. In other words, t is a good summary of a set of outlying descendants. Thus, t on its own may be interesting.

To quantify these two types of outliers, we use Kullback-Leibler divergence or KL-divergence (Kullback et al., 1951) which defines a measure of the difference between two distributions P and Q in information theory. In applications, P typically represents the true distribution of data, observations or precisely calculated theoretical distribution while Q represents a theory, model, description, or approximation of P.

Definition 6 (Kullback-Leibler divergence (Kullback et al., 1951)**).** For probability distributions P and Q of a discrete random variable, the KL-divergence is defined as:

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

It is the expectation of the logarithmic difference between probabilities P and Q where the expectation is taken using the probabilities P. The KL-divergence is defined only if $Q(i) = 0$ implies $P(i) = 0$ for all i (absolute continuity). Whenever $P(i) = 0$, the contribution of the i-term is interpreted as zero because $\lim_{x \to 0} x\log(x) = 0$.

For probability distributions P and Q of a continuous random variable, the KL-divergence is defined to be the integral:

$$KL(P|Q) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx$$

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

KL-divergence is non-negative but not symmetric; that is, generally $KL(P|Q) \neq KL(Q|P)$. For both discrete and continuous cases, the KL-divergence is only defined if P and Q sum to 1 and if $Q(x) > 0$ for any x such that $P(x) > 0$. The smaller the KL-divergence, the more similar the two distributions P and Q.

Definition 7 (Types of Outlier). Let t be a performance outlier and S be the set of base level performance gaps (i.e. the descendants of t). S can be divided into two exclusive groups; S_0 (the subset of normal) and S_1 (the subset of outliers). $S = S_0 \cup S_1$ and $S_0 \cap S_1 \neq \emptyset$. Performance gap t is a **type-I** outlier if $KL(S|S_1) \geq KL(S|S_0)$; otherwise, t is a **type-II** outlier.

4.3. Detection Methods

In this section, we first show that all performance gap outliers can be detected by computing an iceberg cube, we then discuss how to determine the types of outliers.

4.3.1. Outlier Detection by Iceberg Cubing

For a fact table F and a measure threshold τ, an iceberg cube contains all aggregate cells of F whose measures are at least τ. Given a fact table F of k parties, $F = \bigcup_{i=1}^{k+1} F_i$ where $\bigcup_{i=1}^{k} F_i$ are the dimensions and F_{k+1} are the measures, we use existing cube computation methods such as BUC (Beyer et al., 1999) and TDC (Zhao et al., 1997) and $|t.F_{k+1} - m| > l\delta$ as τ to compute an iceberg cube of F which only contains aggregate cells that are outliers.

Many existing iceberg cubing methods rely on the monotonicity property of iceberg conditions; however, the iceberg condition in this problem is not monotonic; that is, the child of a performance gap outlier t may not be an outlier. Since Theorem 3 identifies weak monotonicity of the problem, we use a special method, eBUC (Yu et al., 2005), which looks ahead to check whether an aggregate cell t is an ancestor of outliers. This only requires the storage of base level performance gaps that are outliers.

4.3.2. Outlier Type Determination

As discussed in 4.2, the KL-divergence determines whether an aggregate performance gap outlier is more similar to base level performance gaps that are normal or outlying. For an aggregate performance gap t, let $cov(t)$ be the set of descendants of t. We want to measure the distribution of $|t.F_{k+1} - m|$ for tuples $u \in cov(t)$.

Since $|t.F_{k+1} - m|$ is a continuous random variable, we can apply Kernel Density Estimation (Breiman et al., 1977) or KDE to approximate the distribution. KDE is a non-parametric approach to estimate the probability density function of a random variable. Let $(x_1, x_2, \dots, x_n)$ be an independent and identically distributed sample drawn from some distribution with an unknown density f. We are interested in estimating the shape of this function f. Its kernel density estimator is $\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h\,(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\,(\frac{x-x_i}{h})$ where $K(\cdot)$ is the kernel and is a non-negative function that integrates to one and has the mean zero. $h > 0$ is a smoothing parameter, referred to as **bandwidth**, and determines the width of the kernel.

A range of kernel functions are commonly used. We use Gaussian kernels and the Gaussian approximation (Silverman, 1986). We set $h = 1.06 \times \delta|P|^{-\frac{1}{5}}$ as suggested by Silverman (1986) where δ is the standard deviation of the samples in P. For a single dimension, the density estimator is:

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

Given two distributions P and Q, the KL-divergence returns the difference of distribution of Q given P; thus, the larger the KL-divergence, the more different the two distributions. For P and Q, 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 $P = \{p_1, p_2, \dots, p_m\}$ is the set of samples. Further, the KL-divergence can be estimated as:

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

As per Definition 7, we compare the KL-divergence of an aggregate performance gap outlier with its base level performance gaps that are both normal and outliers. That is, given an aggregate performance gap outlier t, we compare:

$$\widehat{KL}(S|S_0) = \frac{1}{|cov(t)|} \sum_{u \in cov(t)} \ln \frac{S(|u.F_{k+1} - m|)}{S_0(|u.F_{k+1} - m|)}$$

and

$$\widehat{KL}(S|S_1) = \frac{1}{|cov(t)|} \sum_{u \in cov(t)} \ln \frac{S(|u.F_{k+1} - m|)}{S_1(|u.F_{k+1} - m|)}$$

where $S(\cdot), S_0(\cdot), S_1(\cdot)$ are the density functions of S, S_0, S_1 estimated using KDE and $m = \frac{\sum_{u \in cov(t)} u.F_{k+1}}{|cov(t)|}$.

4.4. Empirical Evaluation

We conducted extensive experiments with both synthetic and real data to evaluate the proposed detection methods. The programs were implemented in C++ using Microsoft Visual Studio 2010. All experiments were conducted on a PC with Intel Core Duo E8400 3.0 GHz CPU and 4GB of memory running the Microsoft Windows 7 operating system.

4.4.1. Case Study

To test the effectiveness of the proposed methods, we used a real data set comprising of extended health insurance claims. For privacy protection, the data is anonymized and the service and the location information is coded in such a way that the original and meaningful information cannot be inferred. Note that this case study uses "amount" as the measure; however, the "count" quoted in the examples can just as easily be used as the measure. The average amount is used for each aggregate performance gap. The fact table contains 5,895 base level gaps; the average amount at the base level is $63.50 and the standard deviation is 83.51.

Table 7 shows two example performance gap outliers detected by the methods using the threshold $l = 2$. To fully illustrate these two outliers, all base levels are listed in tables 8 and 9.

Table 7 Example Performance Gap Outliers

ID	Performance Gap Outlier (gender, age-group, location, service code)	Type
R_1	(female, $*$, A101, S91)	I
R_2	(female, $*$, A101, S31)	II

Table 8 Base level Performance Gaps of R_1

ID	Base level Performance Gaps (gender, age-group, location, service 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)	1,106.25	Outlier
8	(female,41-60 , A101, S91)	1,900.00	Outlier

Table 9 Base level Performance Gaps of R_2

ID	Base level Performance Gaps (gender, age-group, location, service code)	Amount	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

As shown in Table 8, R_1 has 8 base levels and among them, 6 are normal and 2 are outliers. For R_1, $KL(S|S_0) = 2.39$ and $KL(S|S_1) = 63.74$. Thus, $KL(S|S_0) < KL(S|S_1)$. This means that the distribution of R_1 is more similar to the distribution of its normal descendants; as such, R_1 is an outlier of type-I.

On the other hand, R_2 is an outlier of type-II since $KL(S|S_0) = 5.01182$ and $KL(S|S_1) = 0.454034$. Thus, $KL(S|S_0) > KL(S|S_1)$.

4.4.2. Efficiency and Scalability

We tested the efficiency of detection methods with both real and synthetic data sets and compared three cubing methods: TDC, BUC, and eBUC. TDC and BUC compute the whole cube while eBUC computes an iceberg cube. We used a larger real data set to test the efficiency and random samples of various sized units of the data set to test the scalability.

Figure 18 shows the scalability of the three methods with respect to the number of tuples for threshold $l = 1$. Note that the smaller the value of l, the more outliers; thus, less pruning power eBUC has. The three methods have linear scalability. While the runtimes of TDC and BUC are very close, eBUC can take advantage of pruning using the outlier detection condition and its runtime is faster.

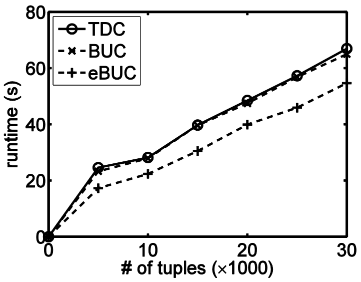
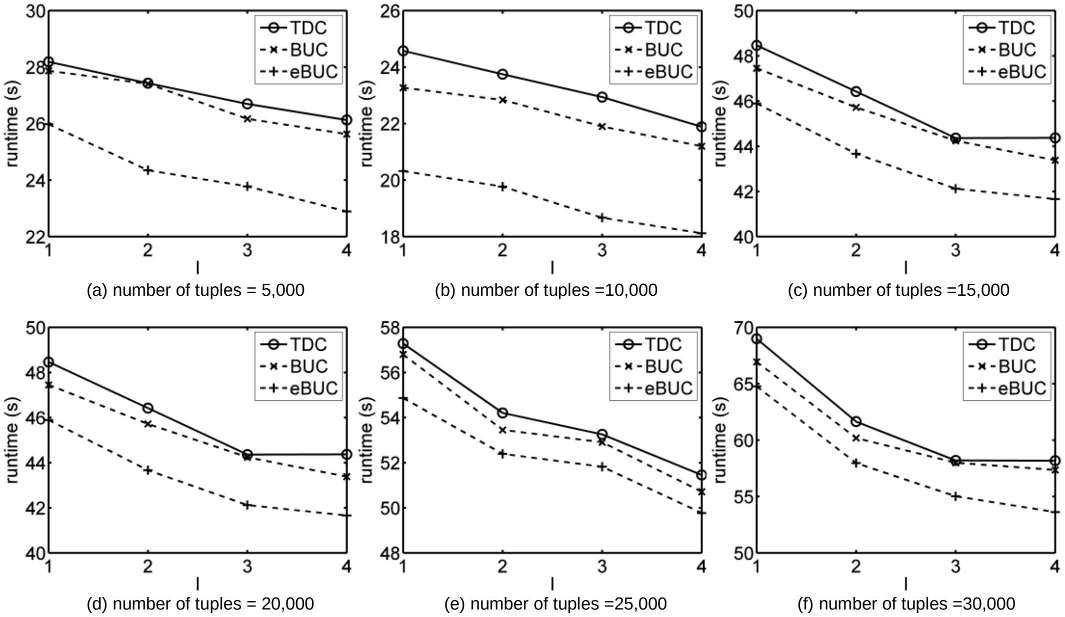


Figure 18 Runtime of TDC, BUC, and eBUC with respect to # of tuples

Figure 19 shows the scalability of the three methods with respect to parameter l and the number of base level performance gaps. The larger the value of l, the less outliers and the less computation required for all three methods to determine the types of outliers. Compared to TDC and BUC, eBUC is able to use the outlier condition to prune normal performance gaps during the cubing process, reducing the runtime further. The larger the value of l, the more advantage eBUC has over other methods. The figure also indicates that the determination of types of outliers incurs a substantial cost.



(a) number of tuples = 5,000

(b) number of tuples =10,000

(c) number of tuples =15,000

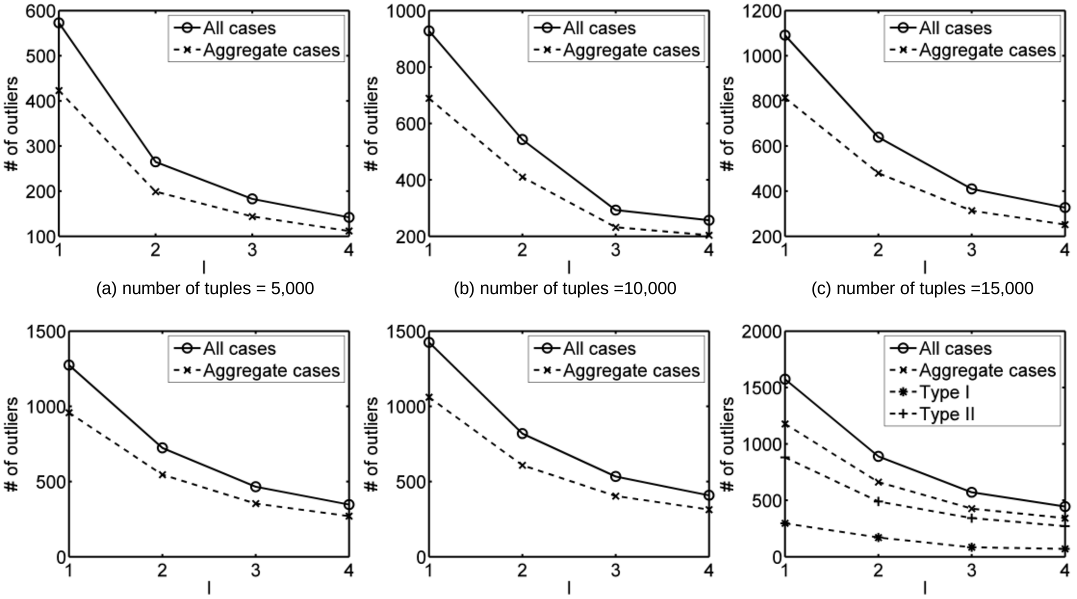
(d) number of tuples = 20,000

(e) number of tuples =25,000

(f) number of tuples =30,000

Figure 19 Runtime of TDC, BUC, and eBUC

Figure 20 shows the number of performance gap outliers with respect to parameter l and the number of base level performance gaps. As indicated earlier as the value of l increases, for all 3 methods, the less computation is needed to determine the types of outliers – the trend is consistent with that shown in Figure 19. Further, most outliers detected are at the aggregate level and there are much more type-II than type-I outliers. The result clearly demonstrates the effectiveness of the proposed method in summarizing outlier information.



(a) number of tuples = 5,000

(b) number of tuples =10,000

(c) number of tuples =15,000

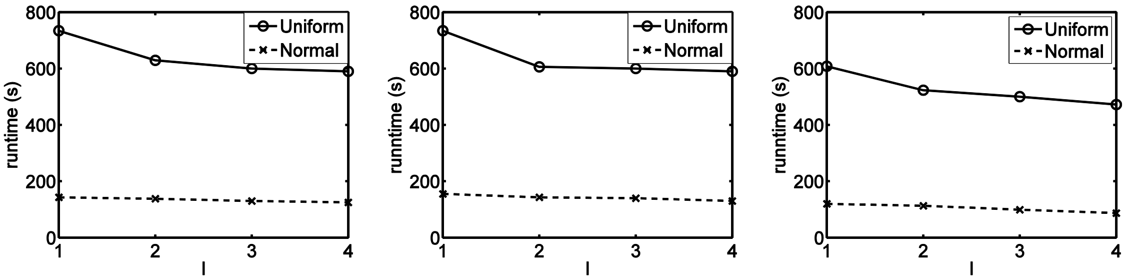
(d) number of tuples = 20,000

(e) number of tuples =25,000

(f) number of tuples =30,000

Figure 20 Number of Detected Outliers

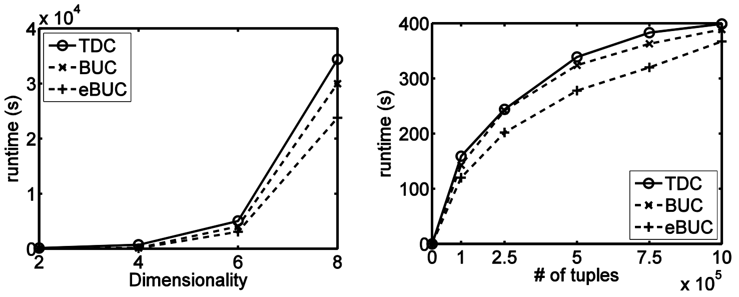
We also tested the efficiency and the scalability using synthetic data sets. Synthetic data sets were generated with dimension attributes in discrete domains and measure in continuous domain. We consider 3 factors in the experiments: the dimensionality d, the number of tuples n in the data set, and the distribution of the dimensions (uniform distribution vs. normal distribution). 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 uniform distribution in each dimension while the tuples in the second data set follow the (discretized) normal distribution; that is, we used normal distribution $\mu = 10$ and $\theta = 2$ to generate data and round the values to 20 bins in the range of $[1, 20]$. Figure 21 shows the results where the threshold parameter $l = 1$. The outlier detection methods work much faster with normally distributed data where outliers are meaningful.



(a) BUC algorithm (b) TDC algorithm (c) eBUC algorithm

Figure 21 Runtime of TDC, BUC, and eBUC with different distributions

Figure 22 shows the scalability of the detection methods with normally distributed data.



(a) Dimensionality (b) Database size

Figure 22 Scalability with Synthetic Data

In Figure 22(a), the number of tuples is set to 10,000. The runtime increases dramatically as the dimensionality increases. This is expected since computing a data cube of high dimensionality is known challenging. In Figure 22(b), the dimensionality is set to 4 and the number of tuples varies from 100,000 to 1 million. The rate of runtime increase grows slower as the number of tuples increases. Given the cardinality of each dimension, the number of possible group-by is fixed. When a fact table becomes very large, many aggregate cells will be populated with a significant number of tuples where the number of group-by grows slower (Figure 23). This result shows that the methods are scalable with large data sets.

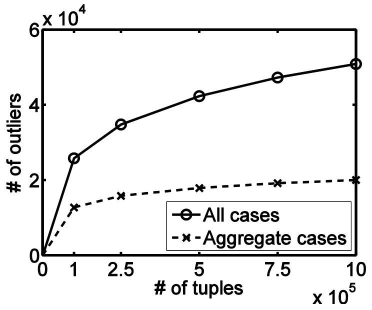


Figure 23 Number of Detected Outliers

Chapter 5.

Subspace Analysis

In Chapter 4, we developed a method to understand how outliers within an aggregate group contribute to the overall deviation of the group from the norm. By recognizing what type of outlier (type-I or type-II) an organization is, it can determine where further analysis should be led to.

In this chapter, we explore subspaces in which an organization's performance is deemed an outlier.

5.1. Contrast Subspace

In a multi-dimensional dataset of 2 classes, given a query object q and a target class, we want to find a subspace in which q most likely belongs to the target class and not to the other class. This subspace is called **contrast subspace** since it contrasts the likelihood of q in the target class to that in the other class. By mining contrast subspaces, we seek an answer for "In what context is the organization most similar to a group of organizations and different from another group?", for example. Using the example given in Figure 1, Figure 24 illustrates the subspace consisting of 2 dimensions: mental health prevalence rate and obesity prevalence rate. Organization q (represented with a red dot) is the query object. Organization q seems to belong to Cluster A of organizations and not to Cluster B; thus, this subspace characterizes a contrast subspace for q, signifying that when it comes to mental health and obesity, q is most contrasting to the majority of organizations (i.e. Cluster B). The insight given by this example may be that q needs to develop a program to reduce the rate of mental health and obesity across its workforce.

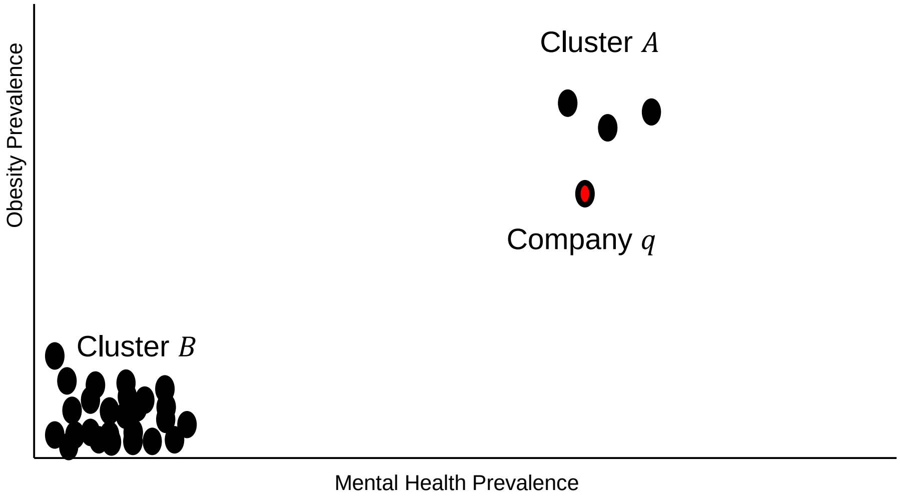


Figure 24 Example Contrast Subspace

5.1.1. Measure of Similarity

A measure to quantify the similarity between the query object and the target class, as well as the difference between the query object and the other class is expressed as the ratio of the likelihood of the query object in the target class against that of the query object in the other class. This is essentially a model selection problem in which one of the two models, M_1 and M_2, must be selected on the basis of observed data D. The probability is assessed by Bayes factor K given by $K = \frac{P_r(D|M_1)}{P_r(D|M_2)}$ where a value of $K > 1$ means that M_1 is more strongly supported by D than M_2.

Problem Definition

Let $D = \{D_1, \ldots, D_d\}$ be a d-dimensional space where the domain of D_i is $\mathbb{R}$, the set of real numbers. A subspace $S \subseteq D$ $(S \neq 0)$ is a subset of D and D is also referred to as the **full space**. The value of an object o in dimension D_i $(1 \leq i \leq d)$ is denoted as $o.D_i$. For a subspace $S = \{D_{i_1}, \ldots, D_{i_l}\} \subseteq D$, the projection of o in S is $o^S = \{o.D_{i_1}, \ldots, o.D_{i_l}\}$. For a set of objects $O = \{o_j | 1 \leq j \leq n\}$, the projection of O in S is $O^S = \{o_j^S | o_j \in O, 1 \leq j \leq n\}$.

Given a set of objects O, a latent distribution $\mathcal{Z}$ is assumed to have generated the objects in O. For a query object q, $L_D(q|\mathcal{Z})$ is the likelihood of q being generated by $\mathcal{Z}$ in full space D. The posterior probability of q given O, denoted by $L_D(q|O)$, can be estimated by $L_D(q|\mathcal{Z})$. For a non-empty subspace $S(S \subseteq D, S \neq 0)$, the projection of $\mathcal{Z}$ in S is $\mathcal{Z}^S$. The subspace likelihood of object q with respect to $\mathcal{Z}$ in S, denoted by $L_S(q|\mathcal{Z})$, can be used to estimate the posterior probability of object q given O in S, denoted by $L_S(q|O)$.

We assume that the objects in O mutually exclusively belong to one of the 2 classes, C_+ and C_-. Hence $O = O_+ \cup O_-$ and $O_+ \cap O_- = \emptyset$ where O_+ and O_- are the objects belonging to C_+ and C_- respectively. Given a query object q, we are interested in finding how likely q belongs to C_+ and does not belong to C_-. We define the measure **likelihood contrast** of q as $LC(q) = \frac{L(q|O_+)}{L(q|O_-)}$.

Likelihood contrast is effectively the Bayes factor of object q being the observation. As such, O_+ and O_- represent the 2 models to choose from based on the query object q. The ratio of probabilities indicates the likelihood of model O_+ selected against O_-. $LC(q)$ values in the range of {<1, 1 to 3, 3 to 10, 10 to 30, 30 to 100, > 100} correspond to {negative, barely worth mentioning, substantial, strong, very strong, decisive} based on the scale for interpretation of Bayes factor according to Jeffreys (1961).

The measure of likelihood contrast can be extended to subspaces. For a non-empty subspace $S(S \subseteq D)$, the likelihood contrast in a subspace S is defined as $LC_S(q) = \frac{L_S(q|O_+)}{L_S(q|O_-)}$. To avoid triviality where $L_S(q|O_+)$ is very small, only the subspaces $L_S(q|O_+) > \delta$, where $\delta \geq 0$ is the minimum likelihood threshold, are considered.

Given a multi-dimensional dataset O in full space D, a query object q, a minimum likelihood threshold $\delta \geq 0$ and a parameter $k \geq 0$, the problem of mining contrast subspace is to find top-k subspaces S ordered by the subspace likelihood contrast $LC_S(q)$ subject to $L_S(q|O_+) > \delta$.

KDE (Breiman et al., 1977) can be used to estimate the likelihood of q, $L_S(q|O)$. Following Silverman (1986), the general formula for multivariate kernel density estimation with kernel K and bandwidth parameter h_S in subspace S is defined as:

$$\hat{f}_S(q,O) = \hat{f}_S(q^S,O) = \frac{1}{|O|h_S^{|S|}} \sum_{o \in O} K\{\frac{1}{h_S}(q-o)\} \tag{5.1.1}$$

Choosing K to be a radially symmetric unimodal probability density function, we use Gaussian kernel:

$$K(x) = \frac{1}{(2\pi)^{\frac{|S|}{2}}} e^{-\frac{1}{2}x^T x} \tag{5.1.2}$$

Given a set of objects O, the density of a query object q in subspace S, denoted by $\hat{f}_S(q,O)$, can be estimated as:

$$\hat{f}_S(q,O) = \hat{f}_S(q^S,O) = \frac{1}{|O|(\sqrt{2\pi}h_S)^{|S|}} \sum_{o \in O} e^{\frac{-dist_S(q,o)^2}{2h_S{}^2}}$$

where $-dist_S(q,o)^2 = \sum_{D_i \in S}(q.D_i - o.D_i)^2$ and h_S is the bandwidth.

According to Silverman (1986), the optimal bandwidth value for smoothing normally distributed data with unit variance is $h_{S\_opt} = A(K)|O|^{\frac{-1}{|S|+4}}$ where $A(K) = (\frac{4}{|S|+2})^{\frac{1}{|S|+4}}$. Since the kernel is radially symmetric and the data in subspaces is not normalized, an inner scale σ_S in subspace S can be used to set $h_S = \sigma_S \cdot h_{S\_opt}$. The term $\sqrt{2\pi}h_S$ is the normalization constant and comes from the fact that the integral over the exponential function (Gaussian kernel) is not unity. With this constant, the Gaussian kernel is a normalized kernel; that is, the integral over its full domain is unity for every h_S. As per Silverman (1986), a possible choice of σ_S is the root of the average marginal variance in S.

The posterior probability of q in subspace S given O can be estimated as:

$$L_S(q|O) = \hat{f}_S(q,O) = \frac{1}{|O|(\sqrt{2\pi}h_S)^{|S|}} \sum_{o \in O} e^{\frac{-dist_S(q,o)^2}{2h_S{}^2}} \tag{5.1.3}$$

Thus, the likelihood contrast of query object q in subspace S is given by:

$$LC_S(q, O_+, O_-) = \frac{\hat{f}_S(q,O_+)}{\hat{f}_S(q,O_-)} = \frac{|O_-|}{|O_+|} \cdot \left(\frac{h_{S_-}}{h_{S_+}}\right)^{|S|} \cdot \frac{\sum_{o\in O_+} e^{\frac{-dist_S(q,o)^2}{2h_{S_+}^2}}}{\sum_{o\in O_-} e^{\frac{-dist_S(q,o)^2}{2h_{S_-}^2}}} \quad (5.1.4)$$

5.1.2. Complexity Analysis

Mining contrast subspaces is computationally challenging and the complexity can be proved by linear reduction or L-reduction from the emerging pattern mining problem (Dong et al., 1999), which has been shown as MAX SNP-hard (Wang et al., 2005). L-reduction is a transformation of optimization problems which linearly preserves approximability features.

Let $D' = \{D'_1, D'_2, \dots, D'_3\}$ be a set of d items. A transaction o'_i is represented by a binary vector of length d whose element $o'_{ij} = 1$ if item D'_j is present and 0 otherwise. A pattern S' is a subset of items in D'. A transaction o'_i satisfies S' if $o'_{ij} = 1, \forall D'_j \in S'$. A transaction database O' is a set of transactions. Let $Sat_{O'}(S')$ be the set of transactions in O' satisfying S'.

Definition 1 (Emerging Pattern Mining (EP)**).** Given two transactions databases O'_+ and O'_-, find pattern S' such that the cost function $c_{EP}(S') = |Sat_{O'_+}(S')|$ is maximized subject to the feasibility condition $|Sat_{O'_-}(S')| = 0$.

Definition 2 (Contrast Subspace Mining (CS)**).** Given $\{q, O_+, O_-\}$ where q is the query object and O_+ and O_- are the two classes, find the subspace S maximizing the cost function $c_{CS}(S, q) = \frac{\sum_{o\in O_+} \exp(\frac{-dist_S(q,o)^2}{2h^2})}{\sum_{o\in O_-} \exp(\frac{-dist_S(q,o)^2}{2h^2})}$.

Definition 3 (Complete Contrast Subspace Mining (Complete-CS)**).** Given $\{O_+, O_-\}$, find the subspace S such that the cost function $c(S) = \max_{o_i \in O_+} c_{CS(S,q=o_i)}$ is maximized.

Complete-CS can be solved by solving at most $|O_+|$ **CS** sub-problems corresponding to unique data points in O_+. We reduce emerging patterns to **Complete-CS** and prove that **Complete-CS** is MAX SNP-hard.

EP → Complete-CS reduction:

- For each item D'_i for EP, set up a corresponding dimension D_i in Complete-CS.
- For each transaction $o'_i \in O'_+$, insert 2 copies of o'_i into O_+.
- For each transaction $o'_i \in O'_-$, insert $2|O'_+|$ copies of o'_i into O_-.
- Insert 1 item (a numeric vector) with all 1's into O_-.
- Let h be an arbitrary user-specified bandwidth parameter. Replace each occurrence of 0 in $O = O_+ \cup O_-$ with a unique value in the set $\{2\gamma h, 3\gamma h, 4\gamma h \ldots\}$ where γ is some fixed large constant.
- Replace each occurrence of 1 in $O = O_+ \cup O_-$ with $1\gamma h$ where h is the same as the one used above.

The transformation can be done in $\boldsymbol{O}(|O_+||O_-|)$ time. An example transformation from a transaction database to a numeric dataset according to the **EP → Complete-CS reduction** is shown in Table 1.

Table 10 EP → Complete-CS reduction example

Database	Transactions (EP)	O_+ (Complete-CS)	O_- (Complete-CS)
O'_+	[0,1,1,0]	$[2\gamma h, 1\gamma h, 1\gamma h, 3\gamma h]$ $[4\gamma h, 1\gamma h, 1\gamma h, 5\gamma h]$	
	[0,1,0,0]	$[6\gamma h, 1\gamma h, 7\gamma h, 8\gamma h]$ $[9\gamma h, 1\gamma h, 10\gamma h, 11\gamma h]$	
O'_-	[1,1,0,0]		$[1\gamma h, 1\gamma h, 12\gamma h, 13\gamma h]$ $[1\gamma h, 1\gamma h, 14\gamma h, 15\gamma h]$ $[1\gamma h, 1\gamma h, 16\gamma h, 17\gamma h]$ $[1\gamma h, 1\gamma h, 18\gamma h, 19\gamma h]$
	[0,0,0,1]		$[20\gamma h, 21\gamma h, 22\gamma h, 1\gamma h]$ $[23\gamma h, 24\gamma h, 25\gamma h, 1\gamma h]$ $[26\gamma h, 27\gamma h, 28\gamma h, 1\gamma h]$ $[29\gamma h, 30\gamma h, 31\gamma h, 1\gamma h]$
			$[1\gamma h, 1\gamma h, 1\gamma h, 1\gamma h]$

Theorem 1. EP → Complete-CS reduction is an L-reduction, denoted by EP→ $_L$Complete-CS.

Definition 4 (L-reduction (Papadimitrious et al., 1991)**).** Let $\prod_1$ and $\prod_2$ be two optimization problems. We say that $\prod_1$ L-reduces to $\prod_2$ if there are two polynomial time algorithms f, g and constants $\alpha, \beta > 0$ such that, for any instance I of $\prod_1$, $f(I)$ forms an instance of $\prod_2$ and:

- $(c1)$ $OPT\big(f(I)\big) \leq \alpha OPT(I)$ where $OPT(\cdot)$ denotes the optimal value of the respective optimization problem.
- $(c2)$ Given any solution s of $f(I)$, algorithm g produces a solution $g(s)$ of I satisfying $|c_{\prod_1}\big(g(s)\big) - OPT(I)| \leq \beta |c_{\prod_2}(s) - OPT\big(f(I)\big)|$, where $c_{\prod_i}(\cdot)$ denotes the cost function of the corresponding optimization problem.

Proof. For any bandwidth value h, we can set γ to a large value such that $\exp\left(\frac{-dist_S(q,o)^2}{2h^2}\right)$ can be arbitrarily close to 0 for all $q \in O$ such that $q^S \neq o^S$. The cost function for CS can be computed as:

$$^{c}\mathrm{CS}(S,q) = \frac{\sum_{o \in O_+} \exp(\frac{-dist_S(q,o)^2}{2h^2})}{\sum_{o \in O_-} \exp(\frac{-dist_S(q,o)^2}{2h^2})} = \frac{|O_+^{S,q}| + \epsilon_+(S,q)}{|O_-^{S,q}| + \epsilon_-(S,q)}$$

where $O^{S,q}$ denotes the set of data points in O having values identical to q in subspace S and:

$$\epsilon_+(S,q) = \sum_{o \in O_+ \backslash O_+^{S,q}} \exp\left(\frac{-dist_S(q,o)^2}{2h^2}\right),$$

$$\epsilon_-(S,q) = \sum_{o \in O_- \backslash O_-^{S,q}} \exp(\frac{-dist_S(q,o)^2}{2h^2}).$$

Let $M > 1$ be the maximum integer value such that $M\gamma h$ is a value occurring in O (e.g. $M = 31$ in the example in Table 1). Then:

$|S|\gamma^2 h^2 < dist_S(q,o)^2 < M^2 |S| \gamma^2 h^2$ for all $o \in O_+ \cup O_-$.

Thus:

$$\big(|O_+| - |O_+^{S,q}|\big) \exp(-|S|\gamma^2 M^2) < \epsilon_+(S,q) < \big(|O_+| - |O_+^{S,q}|\big) \exp(-|S|\gamma^2) \ll 1.$$

Similarly:

$$(|O_-| - |O_-^{S,q}|) \exp(-|S|\gamma^2 M^2) < \epsilon_-(S,q) < (|O_-| - |O_-^{S,q}|) \exp(-|S|\gamma^2) \ll 1.$$

Note that $\lim_{\gamma \to \infty} \epsilon_+(S,q) = 0$ and $\lim_{\gamma \to \infty} \epsilon_-(S,q) = 0$.

We can observe that:

- If a pattern S' is an emerging pattern, then by construction, at least one object $q \in O_+$ must have $|O_+^{S,q}| \geq 2$ and $|O_-^{S,q}| = 1$. This is because S' only appears in O'_+ and for each transaction $o'_i \in O'_+$, we have inserted 2 copies of o'_i into O_+. On the other hand, S' does not appear in O'_- and the only object having values identical to q in subspace S is the object containing all γh's. Therefore, $^c\mathrm{CS}(S, q) = \frac{|O_+^{S,q}|+\epsilon_+(S,q)}{|O_-^{S,q}|+\epsilon_-(S,q)} \geq \frac{2+\epsilon_+(S,q)}{1+\epsilon_-(S,q)} > 1$.
- If a pattern S' is not an emerging pattern, then by construction, all objects $q \in O_+$ must have $|O_-^{S,q}| \geq \left|O_+^{S,q}\right| + 1 > |O_+^{S,q}|$. Therefore, $^c\mathrm{CS}(S, q) = \frac{|O_+^{S,q}|+\epsilon_+(S,q)}{|O_-^{S,q}|+\epsilon_-(S,q)} < 1$.

Further, we need to verify that the reduction $\mathrm{EP} \rightarrow \mathrm{Complete\text{-}CS}$ satisfies the two conditions $(c1)$ and $(c2)$ of the $\mathrm{L\text{-}reduction}$:

- $(c1)$ For any instance I of EP, if S' is the most frequent emerging pattern with $^c\mathbf{EP}(S') = |Sat_{o'_+}(S')|$ and $\left|Sat_{o'_-}(S')\right| = 0$, then the corresponding optimal S solution for $\mathrm{Complete\text{-}CS}$ must have a cost value of $c(S) = \frac{2|Sat_{o'_+}(S')|+\epsilon_+(S,q)}{1+\epsilon_-(S,q)} \simeq 2|Sat_{o'_+}(S')| = 2c_{\mathbf{EP}}(S')$ where q is any data point in O_+ corresponding to the transaction containing pattern S'. This is because for each transaction o'_i containing S' in O'_+, we have inserted 2 copies of o'_i into O_+. The '1' in the denominator is due to the object containing all γh in O_-. Thus, condition 1 is satisfied with $\alpha = 2$ when γ is sufficiently large.
- $(c2)$ For any solution S of $\mathrm{Complete\text{-}CS}$, if $c(S) = \lambda \geq 2$ then the corresponding pattern S' constructed from S will be an emerging pattern. Further, let $[\lambda]$ be the nearest integer to λ. Then, $[\lambda]$ must be even and $\frac{[\lambda]}{2}$ will be the cost of the corresponding EP problem. Let λ^* denote the optimal cost of $\mathrm{Complete\text{-}CS}$, then $|\frac{[\lambda]}{2} - \frac{[\lambda^*]}{2}| = \frac{1}{2}|[\lambda] - [\lambda^*]| \simeq \frac{1}{2}|\lambda - \lambda^*| \leq |\lambda - \lambda^*|$. Thus, condition 2 is satisfied with $\beta = 1$.

Since $\mathrm{EP} \rightarrow_L \mathrm{Complete\text{-}CS}$, if there exists a polynomial time approximation algorithm for $\mathrm{Complete\text{-}CS}$ with performance guarantee $1 - \epsilon$, then there exists a polynomial time approximation algorithm for $\mathbf{EP}$ with performance guarantee $1-\alpha\beta\epsilon$. Since EP is MAX SNP-hard, it follows that $\mathrm{Complete\text{-}CS}$ must also be MAX SNP-hard.

Finally, the relationship between $\mathrm{Complete\text{-}CS}$ and CS is established as follows.

Theorem 2. If there exists a polynomial time approximation scheme (PTAS) for $\mathbf{CS}$, then there must also be a PTAS for $\mathrm{Complete\text{-}CS}$.

Proof. The proof is straightforward since $\mathrm{Complete\text{-}CS}$ can be solved by a series of $|O_+|\mathbf{CS}$ problems.

Unless $\mathrm{P}=\mathrm{NP}$, there exists no PTAS for $\mathrm{Complete\text{-}CS}$, implying no PTAS for $\mathbf{CS}$.

The above theoretical result indicates that the problem of mining contrast subspaces is even hard to approximate; that is, it is impossible to design a good approximation algorithm unless $\mathrm{P}=\mathrm{NP}$. Practical heuristic methods are needed as a viable alternative.

5.1.3. Mining Methods

In this section, we first describe a baseline method which examines every possible non-empty subspace. We then present the design of $\mathrm{CSMiner}$ (for Contrast Subspace Miner) which employs a smarter search strategy.

Baseline Method

The baseline naïve method enumerates all possible non-empty spaces S and calculates the exact values of both $L_S(q|O_+)$ and $L_S(q|O_-)$, since neither $L_S(q|O_+)$ nor $L_S(q|O_-)$ is monotonic with respect to the subspace-superspace relationship. It then returns top-k subspaces S with the largest $LC_S(q)$ values. To ensure the completeness and efficiency of subspace enumeration, the baseline method traverses the set enumeration tree (Rymon, 1992) of subspaces in a depth-first manner. Figure 25 shows a set enumeration tree that enumerates all subspaces of $D=\{D_1,D_2,D_3,D_4\}$.

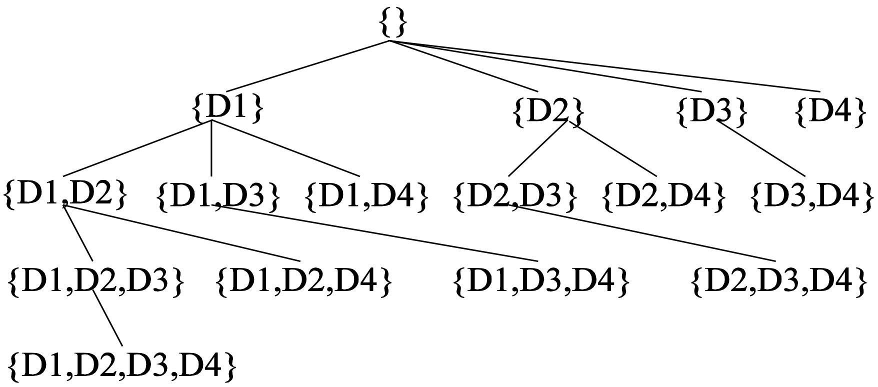


Figure 25 Set enumeration tree

Using Equations (5.1.3) and (5.1.4), the baseline algorithm shown in Algorithm 1 computes the likelihood contrast for every subspace where $L_S(q|O_+) \geq \delta$ and returns the top-k subspaces. The time complexity is $O(2^{|D|} \cdot (|O_+| + |O_-|))$.

Algorithm 1 The baseline algorithm

```
Input: $q$: query object, $O_+$: objects belonging to $C_+$, $O_-$: objects belonging to $C_-$, $\delta$: likelihood threshold, $k$: positive integer
Output: $k$ subspaces with the highest likelihood contrast
let $Ans$ be the current top-$k$ list of subspaces; initialize $Ans$ as $k$ $null$ subspaces associated with likelihood contrast 0
traverse the subspace set enumeration tree in a depth-first search manner
for each subspace $S$ do
    compute $\sigma_{S+}$, $\sigma_{S-}$, $h_{opt}$;
    compute $L_S(q|O_+)$ and $L_S(q|O_-)$ using Equation (5.1.3);
    if $L_S(q|O_+) \geq \delta$ and $\exists S' \in Ans\ s.t. \frac{L_S(q|O_+)}{L_S(q|O_-)} > LC_{S'}(q)$ then
        insert $S$ into $Ans$ and remove $S'$ from $Ans$;
    end if
end for
return $Ans$;
```

CSMiner *Framework*

$L_S(q|O_+)$ is not monotonic in subspaces. We develop an upper bound of $L_S(q|O_+)$ to prune subspaces using the minimum likelihood threshold δ. We sort all dimensions in their standard deviation descending order. Let S be the set of descendants of S in the subspace set enumeration tree using the standard deviation descending order. We define:

$$L_S^*(q|O_+) = \frac{1}{|O_+|(\sqrt{2\pi}\sigma'_{min}h'_{opt\_min})^{\tau}} \sum_{o \in O_+} e^{\frac{-dist_S(q,o)^2}{2\left(\sigma S^{h'}_{opt\_max}\right)^2}} \tag{5.1.5}$$

where $\sigma'_{min} = \min\{\sigma_{S'}|S' \in S\}$, $h'_{opt\_min} = \min\{h_{S',opt}|S' \in S\}$, $h'_{opt\_max} = \max\{h_{S',opt}|S' \in S\}$ and

$$\tau = \begin{cases} |S| & \text{if } \sqrt{2\pi}\sigma'_{min}h'_{opt\_min} \geq 1 \\ \max\{|S'|\ |S' \in S\} & \text{if } \sqrt{2\pi}\sigma'_{min}h'_{opt\_min} < 1 \end{cases}$$

Theorem 3 (Monotonic Density Bound). For a query object q, a set of objects O and subspaces S_1, S_2 such that S_1 is an ancestor of S_2 in the subspace set enumeration tree in which dimensions in full space D are sorted by their standard deviation descending order, it is true that $L^*_{S_1}(q|O) \geq L_{S_2}(q|O)$.

Proof. Let S be the set of descendants of S_1 in the subspace set enumeration tree using the standard deviation descending order in O. We define:

$\sigma'_{min} = \min\{\sigma_{S'}|S' \in S\}$,

$h'_{opt\_min} = \min\{h_{S',opt}|S' \in S\}$,

$h'_{opt\_max} = \max\{h_{S',opt}|S' \in S\}$,

and:

$$\tau = \begin{cases} |\ S_1| & \text{if } \sqrt{2\pi}\sigma'_{min}h'_{opt\_min} \geq 1 \\ \max\{|S'|\ |S' \in S\} & \text{if } \sqrt{2\pi}\sigma'_{min}h'_{opt\_min} < 1 \end{cases}$$

Computing σ'_{min}, h'_{opt_min} and h'_{opt_max} has linear complexity. $\sigma_{S'}$ is the root of the average marginal variance in S' and $h_{S',opt}$ depends on the values of $|O|$ and $|S'|$. Let $S'' \in S$ such that for any subspace $S' \in S$, $S' \subseteq S''$. Since dimensions in the set enumeration tree is sorted in the standard deviation descending order, σ'_{min} can be obtained by checking dimensions in $S'' \backslash S_1$ one by one in the standard deviation ascending order. Further, $h'_{opt\_min}(h'_{opt\_max})$ can be obtained by comparing $h_{S',opt}$ with different values of $|S'| \in [|S_1| + 1, |S''|]$. Since $S_2 \in S$, we have:

$1 \le |S_1| < |S_2| \le \max\{|S'| \mid S' \in S\}$, and $\sigma_{S_1} \ge \sigma_{S_2} \ge \sigma'_{min}$.

Then:

$\sigma_{S_2} h_{S_{2\_opt}} \ge \sigma'_{min} h'_{opt\_min}$.

Thus:

$(\sqrt{2\pi}\sigma s_2 h_{S_{2\_opt}})^{|S_2|} > (\sqrt{2\pi}\sigma'_{min} h'_{opt\_min})^{\tau}$.

For $o \in O, dist_{S_1}(q, o) \le dist_{S_2}(q, o)$. Accordingly, $\frac{-dist_{S_2}(q,o)^2}{2(\sigma s_2 h_{S_{2\_opt}})^2} \le \frac{-dist_{S_1}(q,o)^2}{2(\sigma s_1 h'_{opt\_max})^2}$.

By Equation (5.1.3):

$$
\begin{aligned}
L_{S_2}(q|O) &= \frac{1}{|O|\left(\sqrt{2\pi}\sigma s_2 h_{S_{2\_opt}}\right)^{|S_2|}} \sum_{o \in O} e^{\frac{-dist_{S_2}(q,o)^2}{2\left(\sigma s_2 h_{S_{2\_opt}}\right)^2}} \\
&\le \frac{1}{|O|\left(\sqrt{2\pi}\sigma'_{min} h'_{opt\_min}\right)^{\tau}} \sum_{o \in O} e^{\frac{-dist_{S_1}(q,o)^2}{2\left(\sigma s_1 h'_{opt\_max}\right)^2}} \\
&= L^*_{S_1}(q|O)
\end{aligned}
$$

Using Theorem 3, in addition to $L_S(q|O_+)$ and $L_S(q|O_-)$, we also compute $L^*_S(q|O_+)$ for each subspace S. We define the pruning rules based on the theorem.

Pruning Rule 1. Given a minimum likelihood threshold δ, if $L^*_S(q|O_+) < \delta$ in a subspace S, all descendants of S can be pruned.

By using the depth-first search, the distance between two objects in a superspace can be computed incrementally from the distance among the objects in a subspace. Given two objects q and o, let subspace $S' = S \cup \{D_i\}$. We have $dist_{S'}(q, o)^2 = dist_S(q, o)^2 + (q.D_i - o.D_i)^2$.

Algorithm 2 shows the pseudo code of the CSMiner. Similar to the baseline method (Algorithm 1), CSMiner conducts a depth-first search on the subspace set enumeration tree. For a candidate subspace S, CSMiner calculates $L^*_S(q|O_+)$ using Equation (5.1.5). If $L^*_S(q|O_+)$ is less than the minimum likelihood threshold, all descendants of S can be pruned by Theorem 3. Due to the difficulty of the problem shown in section 5.1.2 and the heuristic nature of this method, the time complexity of CSMiner is $O(2^{|D|} \cdot (|O_+| + |O_-|))$, the same time complexity as the naïve baseline method. However, as will be shown by the empirical evaluation (section 5.1.4), CSMiner is substantially faster than the baseline method.

Algorithm 2 CSMiner (q, O_+, O_-, δ, k)

```
Input: $q$: query object, $O_+$: objects belonging to $C_+$, $O_-$: objects belonging to $C_-$, $\delta$: likelihood threshold, $k$: positive integer
Output: $k$ subspaces with the highest likelihood contrast
1: let $Ans$ be the current top-$k$ list of subspaces; initialize $Ans$ as $k$ $null$ subspaces associated with likelihood contrast 0
2: traverse the subspace set enumeration tree in a depth-first search manner
3: for each subspace $S$ do
4:     compute $\sigma_{S+}$, $\sigma_{S-}$, $\sigma'_{min}, h_{opt}, h'_{opt\_min}, h'_{opt\_max}$;
5:     compute $L^*_S(q|O_+)$ using Equation (5.1.5);
6:     if $L^*_S(q|O_+) < \delta$ then
7:         prune all descendants of $S$ and go to Step 2; //Pruning Rule 1
8:     else
9:         compute $L_S(q|O_+)$ and $L_S(q|O_-)$ using Equation (5.1.3);
10:        if $L_S(q|O_+) \geq \delta$ and $\exists S' \in Ans\ s.t. \frac{L_S(q|O_+)}{L_S(q|O_-)} > LC_{S'}(q)$ then
11:            insert $S$ into $Ans$ and remove $S'$ from $Ans$;
12:        end if
13:    end if
14: end for
15: return $Ans$;
```

A Bounding-Pruning-Refining Method

For a query object q and a set of objects O, the likelihood $L_S(q|O)$, computed by Equation (5.1.3), is the sum of density contributions of objects in O to q in subspace S. In Gaussian kernel estimation, given object $o \in O$, the contribution from o to $L_S(q|O)$ is $\frac{1}{|O|\sqrt{2\pi}h_S{}^{|S|}} e^{\frac{-dist_S(q,o)^2}{2h_S{}^2}}$. We observe that the contribution of o decays exponentially as the distance between q and o increases; thus, $L_S(q|O)$ can be bounded.

For a query object q and a set of objects O, the ϵ-$neighbourhood$ ($\epsilon > 0$) of q in subspace S is $N^\epsilon_S(q) = \{o \in O | dist_S(q,o) \leq \epsilon\}$. We can divide $L_S(q|O)$ into two parts; $L_S(q|O) = L_{N^\epsilon_S}(q|O) + L^{rest}_S(q|O)$. The first part is contributed by the objects in the

$\epsilon\text{-}neighbourhood$; that is, $L_{N_S^\epsilon}(q|O) = \frac{1}{|O|(\sqrt{2\pi}h_S)^{|S|}} \sum_{o \in N_S^\epsilon(q)} e^{\frac{-dist_S(q,o)^2}{2h_S{}^2}}$ and the second part is by the objects outside the $\epsilon\text{-}neighbourhood$; that is, $L_S^{rest}(q|O) = \frac{1}{|O|(\sqrt{2\pi}h_S)^{|S|}} \sum_{o \in O \setminus N_S^\epsilon(q)} e^{\frac{-dist_S(q,o)^2}{2h_S{}^2}}$.

Let $\overline{dist_S}(q|O)$ be the maximum distance between q and all objects in O in subspace S. We have $\frac{|O|-|\,N_S^\epsilon(q)|}{|O|(\sqrt{2\pi}h_S)^{|S|}} \cdot e^{-\frac{\overline{dist_S}(q,o)^2}{2h_S{}^2}} \leq L_S^{rest}(q|O) \leq \frac{|O|-|\,N_S^\epsilon(q)|}{|O|(\sqrt{2\pi}h_S)^{|S|}} \cdot e^{-\frac{\epsilon^2}{2h_S{}^2}}$.

The example in Figure 26 illustrates a $\epsilon\text{-}neighbourhood$ of object q with respect to object set O in a 2-dimensional subspace S. We can see that $N_S^\epsilon(q) = \{o_1, o_2, o_3, o_4, o_5\}$ and $\overline{dist_S}(q|O) = dist_S(q, o_{10})$.

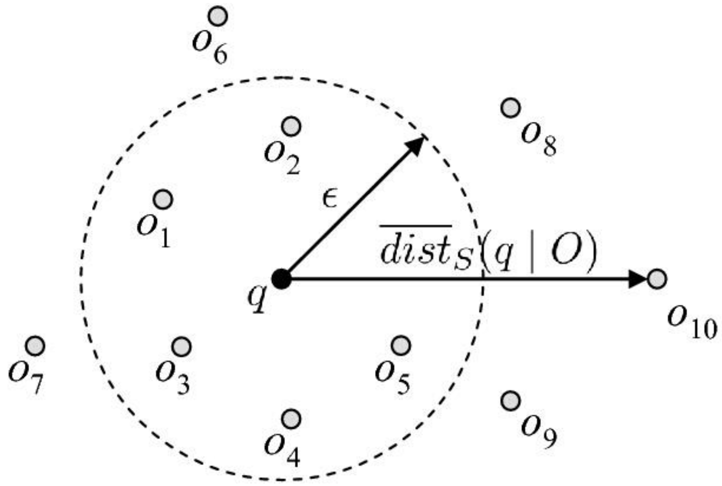


Figure 26 $\boldsymbol{\epsilon\text{-}neighbourhood}$ (within the dashed circle)

An upper bound of $L_S^*(q|O_+)$ using $\epsilon\text{-}neighbourhood$ denoted by $L_S^{*\epsilon}(q|O_+)$ is:

$$L_S^{*\epsilon}(q|O_+) = \frac{\sum_{o \in N_S^\epsilon(q)} e^{\frac{-dist_S(q,o)^2}{2\left(\sigma S_{opt\_max}^{h\prime}\right)^2}} + (|O_+| - |N_S^\epsilon(q)|) e^{-\frac{\epsilon^2}{2\left(\sigma S_{opt\_max}^{h\prime}\right)^2}}}{|O_+|\left(\sqrt{2\pi}\sigma'_{min} h'_{opt\_min}\right)^\tau}$$

where the meanings of $\sigma'_{min}, h'_{opt\_min}, h'_{opt\_max}$ and τ are the same as those in Equation (5.1.5).

Pruning Rule 2. Given a minimum likelihood threshold δ, if $L_S^{*\epsilon}(q|O_+) < \delta$ in a subspace S, all descendants of S can be pruned.

Using the ϵ-$neighbourhood$, we have the following upper and lower bounds of $L_S(q|O)$.

Theorem 4 (Bounds). For a query object q, a set of objects O and $\epsilon \geq 0$, $LL_S^{\epsilon}(q|O) \leq L_S(q|O) \leq UL_S^{\epsilon}(q|O)$

where:

$$LL_S^{\epsilon}(q|O) = \frac{1}{|O|(\sqrt{2\pi}h_S)^{|S|}}\left(\sum\nolimits_{o \in N_S^{\epsilon}(q)} e^{\frac{-dist_S(q,o)^2}{2{h_S}^2}} + (|O| - |N_S^{\epsilon}(q)|)e^{\frac{-dist_S(q,o)^2}{2{h_S}^2}}\right)$$

and:

$$UL_S^{\epsilon}(q|O) = \frac{1}{|O|(\sqrt{2\pi}h_S)^{|S|}}\left(\sum\nolimits_{o \in N_S^{\epsilon}(q)} e^{\frac{-dist_S(q,o)^2}{2{h_S}^2}} + (|O| - |N_S^{\epsilon}(q)|)e^{-\frac{\epsilon^2}{2{h_S}^2}}\right).$$

Proof. For any object $o \in O \backslash N_S^{\epsilon}(q), \epsilon^2 \leq dist_S(q,O)^2 \leq \overline{dist}_S(q,O)^2$.

Then:

$$e^{-\frac{\epsilon^2}{2{h_S}^2}} \geq e^{\frac{-dist_S(q,O)^2}{2{h_S}^2}} \geq e^{-\frac{\overline{dist}_S(q,O)^2}{2{h_S}^2}}.$$

Thus:

$$(|O| - |N_S^{\epsilon}(q)|)\, e^{-\frac{\epsilon^2}{2{h_S}^2}} \geq (|O| - |N_S^{\epsilon}(q)|)e^{\frac{-dist_S(q,O)^2}{2{h_S}^2}} \geq (|O| - |N_S^{\epsilon}(q)|)e^{-\frac{\overline{dist}_S(q,O)^2}{2{h_S}^2}}.$$

Accordingly, $LL_S^{\epsilon}(q|O) \leq L_S(q|O) \leq UL_S^{\epsilon}(q|O)$. We obtain an upper bound of $LC_S(q)$ based on Theorem 4 and Equation (5.1.4).

Corollary 1 (Likelihood Contrast Upper Bound). For a query object q, a set of objects O_+, a set of objects O_- and $\epsilon \geq 0$, $LC_S(q) \leq \frac{UL_S^{\epsilon}(q|O_+)}{LL_S^{\epsilon}(q|O_-)}$.

Proof. By Theorem 4, we have $L_S(q|O_+) \leq UL_S^{\epsilon}(q|O_+)$ and $L_S(q|O_-) \geq LL_S^{\epsilon}(q|O_-)$. Then, $LC_S(q) = \frac{L_S(q|O_+)}{L_S(q|O_-)} \leq \frac{UL_S^{\epsilon}(q|O_+)}{L_S(q|O_-)} \leq \frac{UL_S^{\epsilon}(q|O_+)}{LL_S^{\epsilon}(q|O_-)}$.

Using Corollary 1, we have the following rule.

Pruning Rule 3. For a subspace S, if there are at least k subspaces whose likelihood contrast are greater than $\frac{UL_S^{\epsilon}(q|O_+)}{LL_S^{\epsilon}(q|O_-)}$, then S cannot be a top-k subspace of the largest likelihood contrast.

We implement the bounding-pruning-refining method in CSMiner to compute bounds of likelihood and contrast ratio. We call this version CSMiner-BPR. For a candidate subspace S, CSMiner-BPR calculates $UL_S^{\epsilon}(q|O_+), LL_S^{\epsilon}(q|O_-)$ and $L_S^{*\epsilon}(q|O_+)$ using the ϵ-*neighbourhood*. If $UL_S^{\epsilon}(q|O_+)$ is less than the minimum likelihood threshold (δ), CSMiner-BPR checks whether it is true that $L_S^{*\epsilon}(q|O_+) < \delta$ (Pruning Rule 2) or $L_S^{*}(q|O_+) < \delta$ (Pruning Rule 1). Otherwise, CSMiner-BPR checks whether the likelihood contrasts of the current top-k subspaces are larger than the upper bound of $LC_S(q)$ $(= \frac{UL_S^{\epsilon}(q|O_+)}{LL_S^{\epsilon}(q|O_-)})$. If not, CSMiner-BPR refines $L_S^{*}(q|O_+), L_S(q|O_+)$ and $L_S(q|O_-)$ by involving objects that are out of the ϵ-*neighbourhood*. S will be added to the current top-k list if $L_S^{*}(q|O_+) \geq \delta$ and the ratio of $L_S(q|O_+)$ to $L_S(q|O_-)$ is larger than one of the current top-k ones. The computational cost for $L_S^{*}(q|O_+)$ is high when the size of O_+ is large. Thus, for efficiency, we consider a tradeoff between Pruning Rule 1 and Pruning Rule 3. Specifically, when we are searching a subspace S, once we can determine that S cannot be a top-k contrast subspace, then we terminate the search of S immediately. In this manner, CSMiner-BPR accelerates CSMiner by avoiding the cost for computing the likelihood contributions of objects outside the ϵ-*neighbourhood* to q when $L_S^{*\epsilon}(q|O_+) < \delta$ (Pruning Rule 2) or $\frac{UL_S^{\epsilon}(q|O_+)}{LL_S^{\epsilon}(q|O_-)} < \delta$ (Pruning Rule 3).

The ϵ-*neighbourhood* is a critical instrument for decreasing the computational cost for CSMiner-BPR. However, when dimensionality increases, the distance between objects increases, as such, the value of ϵ should not be fixed. Standard deviation is a measure that expresses the variability of a set of data. For subspace S, we set $\epsilon = \sqrt{\alpha \cdot \sum_{D_i \in S}(\sigma_{D_i+}^2 + \sigma_{D_i-}^2)}$ $(\alpha \geq 0)$ where $\sigma_{D_i+}^2$ and $\sigma_{D_i-}^2$ are the marginal variances of O_+ and O_- respectively on dimension D_i $(D_i \in S)$, and α is a system defined parameter. Our experiments show that α can be set in the range of 0.8 to 1.2 and is not sensitive. Algorithm 3 provides the pseudo code of CSMiner-BPR. Theorem 5 guarantees that no matter how varied the neighbourhood distance (ϵ) may be, the mining result of $\mathrm{CSMiner\text{-}BPR}$ remains unchanged.

Theorem 5. Given data set O, query object q, minimum likelihood threshold δ and parameter k, for any neighbourhood distance ϵ_1 and ϵ_2, $CS^{\epsilon_1}(q|O) = CS^{\epsilon_2}(q|O)$ where $CS^{\epsilon_1}(q|O)$ ($CS^{\epsilon_2}(q|O)$) is the set of contrast subspaces discovered by $\mathrm{CSMiner\text{-}BPR}$ using ϵ_1 (ϵ_2).

Proof by contradiction. Assume that subspace $S \in CS^{\epsilon_1}(q|O)$ but $S \notin CS^{\epsilon_2}(q|O)$. As $S \in CS^{\epsilon_1}(q|O)$, we have $(*)\, L_S(q|O_+) \geq \delta$. On the other hand, $S' \notin CS^{\epsilon_2}(q|O)$ means that $(i)\, L_S^{*\epsilon_2}(q|O_+) < \delta$, or $(ii)\, \exists S' \in CS^{\epsilon_2}(q|O)$ such that $S' \notin CS^{\epsilon_1}(q|O)$ and $\frac{UL_S^{\epsilon_1}(q|O_+)}{LL_S^{\epsilon_1}(q|O_-)} < LC_{S'}(q)$. For case (i), as $L_S(q|O_+) \leq L_S^*(q|O_+) \leq L_S^{*\epsilon_2}(q|O_+)$, we have $L_S(q|O_+) < \delta$, contradicting $(*)$. For case (ii), as $LC_S(q) \leq \frac{UL_S^{\epsilon_1}(q|O_+)}{LL_S^{\epsilon_1}(q|O_-)}$, we have $LC_S(q) < LC_{S'}(q)$, contradicting $S' \notin CS^{\epsilon_1}(q|O)$.

Corollary 2. Given data set O, query object q, minimum likelihood threshold δ and parameter k, the mining result of $\mathrm{CSMiner\text{-}BPR}$, no matter what the value of parameter α is, the output is the same as that of $\mathrm{CSMiner}$.

Proof. For subspace S, suppose ϵ, computed by parameter α, is not less than $\overline{dist_S}(q, O)$. We have $N_S^{\epsilon}(q) = \emptyset$. As such, $UL_S^{\epsilon}(q|O_+) = L_S(q|O_+), LL_S^{\epsilon}(q|O_-) = L_S(q|O_-)$ and $L_S^{*\epsilon}(q|O_+) = L_S^*(q|O_+)$. This means that the execution flow of $\mathrm{CSMiner\text{-}BPR}$ (Algorithm 3) is the same as that of $\mathrm{CSMiner}$ (Algorithm 2). Further, by Theorem 5, the value of neighbourhood distance does not change the mining result of $\mathrm{CSMiner\text{-}BPR}$.

Algorithm 3 $\mathrm{CSMiner\text{-}BPR}\,(q, O_+, O_-, \delta, k, \alpha)$

```
Input: $q$: query object, $O_+$: objects belonging to $C_+$, $O_-$: objects belonging to $C_-$, $\delta$: likelihood threshold, $k$: positive integer, $\alpha$: neighbourhood parameter
Output: $k$ subspaces with the highest likelihood contrast
let $Ans$ be the current top-$k$ list of subspaces; initialize $Ans$ as $k$ $null$ subspaces associated with likelihood contrast 0
for each subspace $S$ in the subspace set enumeration tree searched in the depth-first manner
```

```
    do
3:    compute $\epsilon$, $\sigma_{S+}$, $\sigma_{S-}$, $\sigma'_{min}, h_{opt}, h'_{opt\_min}, h'_{opt\_max}$;
4:    $N_S^\epsilon(q)_+ \leftarrow \emptyset; N_S^\epsilon(q)_- \leftarrow \emptyset; \overline{dist_S}(q|O_-) \leftarrow 0$;
5:    for each object $o \in O_+ \cup O_-$ do
6:      $dist_S(q,o)^2 \leftarrow dist_{S^P}(q,o)^2 + (q.D' - o.D')^2$ //$S^P(= S \cup \{D'\})$ is the parent of $S$.
7:      if $o \in O_+$ and $dist_S(q,o) < \epsilon$ then
8:        $N_S^\epsilon(q)_+ \leftarrow N_S^\epsilon(q)_+ \cup \{o\}$;
9:      end if
10:     if $o \in O_-$ then
11:       if $dist_S(q,o) < \epsilon$ then
12:         $N_S^\epsilon(q)_+ \leftarrow N_S^\epsilon(q)_+ \cup \{o\}$;
13:       end if
14:       if $\overline{dist_S}(q|O_-) < dist_S(q,o)$ then
15:         $\overline{dist_S}(q|O_-) \leftarrow dist_S(q,o)$;
16:       end if
17:     end if
18:   end for
19:   compute $UL_S^\epsilon(q|O_+)$, $LL_S^\epsilon(q|O_-), L_S^{*\epsilon}(q|O_+)$ ; //bounding
20:   if $UL_S^\epsilon(q|O_+) < \delta$ then
21:     if $L_S^{*\epsilon}(q|O_+) < \delta$ then
22:       prune all descendants of $S$ and go to step 2; //Pruning Rule 2
23:     end if
24:     compute $L_S^*(q|O_+)$;
25:     if $L_S^*(q|O_+) < \delta$ then
26:       prune all descendants of $S$ and go to step 2; //Pruning Rule 1
27:     end if
28:   else
29:     if $\exists S' \in Ans\ s.t. \frac{UL_S^\epsilon(q|O_+)}{LL_S^\epsilon(q|O_-)} \geq LC_{S'}(q)$ then
30:       compute $L_S^*(q|O_+)$ using Equation (5.1.5); //refining
31:       if $L_S^*(q|O_+) < \delta$ then
32:         prune all descendants of $S$ and go to step 2; //Pruning Rule 1
33:       else
34:         compute $L_S(q|O_+)$ and $L_S(q|O_-)$ using Equation (5.1.3); //refining
35:         if  $L_S(q|O_+) \geq \delta$ and $\exists S' \in Ans\ s.t. \frac{L_S(q|O_+)}{L_S(q|O_-)} \geq LC_{S'}(q)$ then
36:           insert $S$ into $Ans$ and remove $S'$ from $Ans$;
37:         end if
38:       end if
39:     end if
```

```
        end if
    end for
return $Ans$;
```

5.1.4. Empirical Evaluation

In this section, we present a systematic empirical study using real data sets to demonstrate the effectiveness and efficiency of CSMiner (or CSMiner-BPR). The focus of the evaluation includes:

- How sensitive our methods are to the running parameters (δ, k, α) in terms of discovered contrast subspaces and running time;
- How sensitive our methods are to different bandwidth values and kernel function in terms of similarity among mined results.

All experiments were conducted on a PC with Intel Core i7-3770 3.40 GHz CPU and 8GB RAM, running Windows 7 operating system. All algorithms were implemented in Java and compiled with JDK 7. Defaults were set to $\delta = 0.001, k = 10, \alpha = 0.8$.

Effectiveness

Table 11 summarizes the 6 data sets obtained from UCI machine learning repository (Bache et al., 2013) and characteristics of each data set. Non-numerical attributes and all records that are missing values were removed from the data sets.

Table 11 Data Set Characteristics

Data Set	# of Objects	# of Attributes
Breast Cancer Wisconsin (BCW)	683	9
Climate Model Simulation Crashes (CMSC)	540	18
Glass Identification (Glass)	214	9
Pima Indians Diabetes (PID)	768	8
Waveform	5000	21
Wine	178	13

For each data set, we select one object as a query object q at a time and put all objects belonging to the same class as q in the set, O_1 (except q). All remaining objects are put in O_2. Using CSMiner, for each object, we compute:

(1) **inlying contrast subspace** taking O_1 as O_+ and O_2 as O_- and:

(2) **outlying contrast subspace** taking O_1 as O_+ and O_2 as O_-.

For this experiment, we only compute the top-1 subspace. For clarity, we denote the likelihood contrasts of inlying contrast subspace by $LC_S^{in}(q)$ and those of outlying contrast subspace by $LC_S^{out}(q)$. Tables 12 to 17 show the joint distributions of $LC_S^{in}(q)$ and $LC_S^{out}(q)$ in each data set. For most objects, $LC_S^{in}(q)$ are larger than $LC_S^{out}(q)$. This is expected since q and all objects in O_1 belong to the same class. However, a good number of objects have strong outlying contrast subspaces. For example, in CMSC, more than 40% of objects have outlying contrast subspaces, satisfying $LC_S^{out}(q) \geq 10^3$. Further, except for PID, a considerable number of objects in each data set have both strong inlying contrast subspaces and outlying contrast subspaces (e.g. $LC_S^{in}(q) \geq 10^4$ and $LC_S^{out}(q) \geq 10^2$).

Table 12 Distribution of $\boldsymbol{LC_S(q)}$ in BCW ($\boldsymbol{\delta = 0.001, k = 1}$)

		$\boldsymbol{LC_S^{out}(q)}$					
$\boldsymbol{LC_S^{in}(q)}$		< 1	$[1, 3)$	$[3, 10)$	$[10, 10^2)$	$\geq 10^2$	$\boldsymbol{Total}$
	$< 10^4$	0	3	0	7	23	33
	$[10^4, 10^5)$	7	4	2	4	7	24
	$[10^5, 10^6)$	21	21	5	8	9	64
	$[10^6, 10^7)$	184	33	5	4	9	235
	$\geq 10^7$	121	31	74	66	35	327
$\boldsymbol{Total}$		333	92	86	89	83	683

Table 13 Distribution of $\boldsymbol{LC_S(q)}$ in CMSC ($\boldsymbol{\delta = 0.001, k = 1}$)

		$\boldsymbol{LC_S^{out}(q)}$					
$\boldsymbol{LC_S^{in}(q)}$		$[10, 10^2)$	$[10^2, 10^3)$	$[10^3, 10^4)$	$[10^4, 10^5)$	$\geq 10^5$	$\boldsymbol{Total}$
	$< 10^3$	1	11	12	2	0	26
	$[10^3, 10^4)$	6	35	47	6	6	100
	$[10^4, 10^5)$	10	46	44	8	2	110
	$[10^5, 10^6)$	11	40	32	8	2	93
	$\geq 10^6$	39	110	50	11	1	211
$\boldsymbol{Total}$		67	242	185	35	11	540

Table 14 Distribution of $LC_S(q)$ in Glass ($\boldsymbol{\delta = 0.001, k = 1}$)

		$\boldsymbol{LC_S^{out}(q)}$					
		< 1	$[1, 3)$	$[3, 10)$	$[10, 10^2)$	$\geq 10^2$	$\boldsymbol{Total}$
$\boldsymbol{LC_S^{in}(q)}$	$< 10^2$	0	0	0	1	7	8
	$[10^2, 10^3)$	2	8	4	4	7	25
	$[10^3, 10^4)$	28	91	6	4	5	134
	$[10^4, 10^5)$	1	4	0	0	3	8
	$\geq 10^5$	0	1	0	30	8	39
$\boldsymbol{Total}$		31	104	10	39	30	214

Table 15 Distribution of $LC_S(q)$ in PID ($\boldsymbol{\delta = 0.001, k = 1}$)

		$\boldsymbol{LC_S^{out}(q)}$					
		< 1	$[1, 3)$	$[3, 10)$	$[10, 30)$	≥ 30	$\boldsymbol{Total}$
$\boldsymbol{LC_S^{in}(q)}$	< 1	0	0	1	0	0	1
	$[1, 3)$	2	241	62	8	2	315
	$[3, 10)$	36	328	31	3	0	398
	$[10, 30)$	23	23	2	0	0	48
	≥ 30	3	3	0	0	0	6
$\boldsymbol{Total}$		64	595	96	11	2	768

Table 16 Distribution of $LC_S(q)$ in Waveform ($\boldsymbol{\delta = 0.001, k = 1}$)

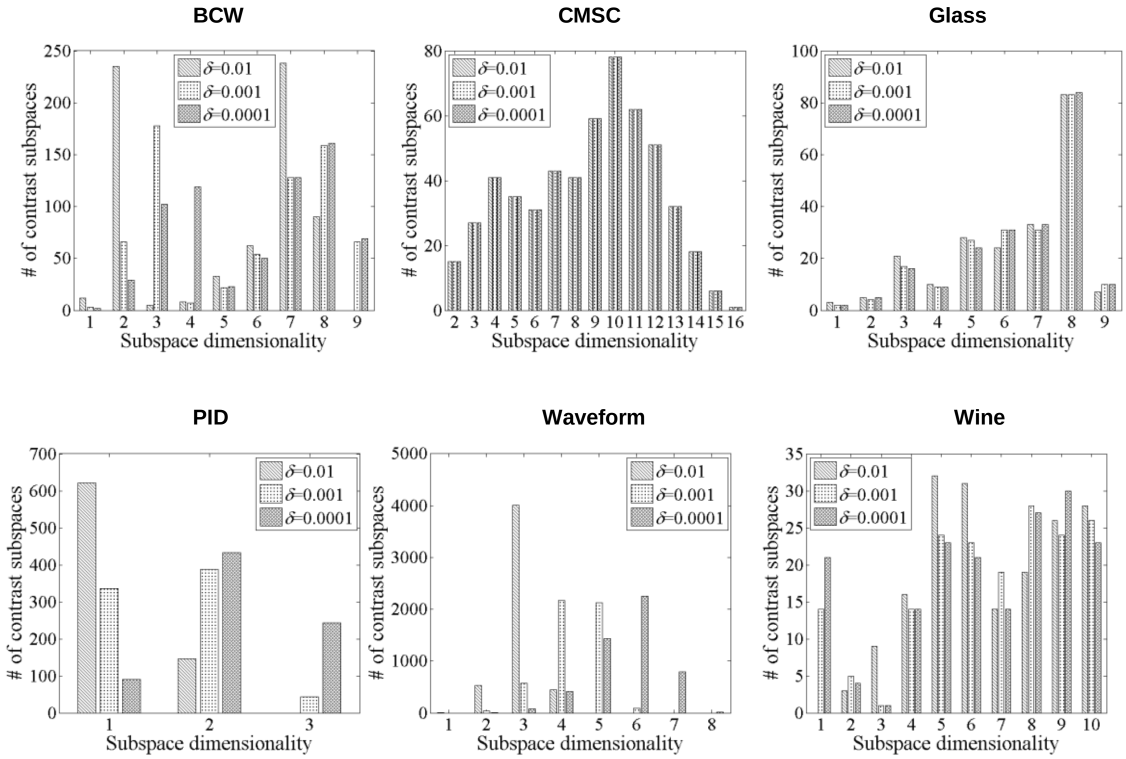
		$\boldsymbol{LC_S^{out}(q)}$					
		$[1, 3)$	$[3, 10)$	$[10, 10^2)$	$[10^2, 10^3)$	$\geq 10^3$	$\boldsymbol{Total}$
$\boldsymbol{LC_S^{in}(q)}$	< 10	0	24	34	8	2	68
	$[10, 10^2)$	204	676	772	190	71	1913
	$[10^2, 10^3)$	471	1049	981	228	56	2785
	$[10^3, 10^4)$	53	103	67	4	4	231
	$\geq 10^4$	0	2	1	0	0	3
$\boldsymbol{Total}$		728	1854	1855	430	133	5000

Table 17 Distribution of $LC_S(q)$ in Wine ($\boldsymbol{\delta = 0.001, k = 1}$)

		$\boldsymbol{LC_S^{out}(q)}$					
		< 1	$[1, 3)$	$[3, 10)$	$[10, 10^2)$	$\geq 10^2$	$\boldsymbol{Total}$
$\boldsymbol{LC_S^{in}(q)}$	$< 10^3$	0	13	8	7	5	33
	$[10^3, 10^4)$	1	18	11	4	0	34
	$[10^4, 10^5)$	2	23	12	5	2	44
	$[10^5, 10^6)$	3	7	5	1	0	16
	$\geq 10^6$	7	20	16	4	4	51
$\boldsymbol{Total}$		13	81	52	21	11	178

Figures 27 and 28 show the distributions of dimensionality of top-1 inlying contrast subspaces and outlying contrast subspaces with different minimum likelihood thresholds

(δ) respectively. In most cases, the contrast subspaces tend to have low dimensionality; however, in CMSC and Wine, the inlying contrast subspaces tend to have high dimensionality. Finally, as the value of δ decreases, the number of subspaces with higher dimensionality typically increases.



BCW

CMSC

Glass

PID

Waveform

Wine

Figure 27 Dimensionality distribution of top inlying contrast subspace ($k = 1$)

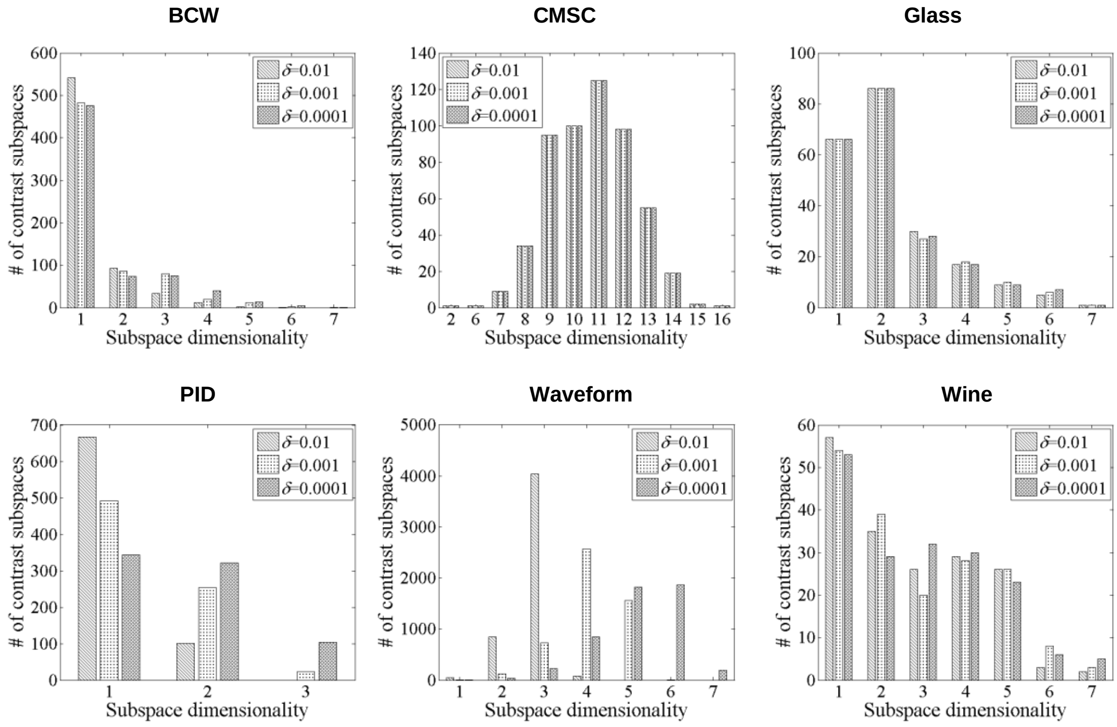


Figure 28 Dimensionality distribution of top outlying contrast subspace ($k = 1$)

Efficiency

To the best of our knowledge, there is no previous method addressing the same mining problem. As such, we will evaluate the efficiency of CSMiner and its variations; that is, comparisons amongst Algorithms 1 (baseline), 2 (CSMiner), and 3 (CSMiner-BPR). Since Waveform data set is the largest, we use this data set only and randomly select 100 objects as query objects and report the average runtime.

Figure 29 shows the runtime with respect to the minimum likelihood threshold δ. A logarithm scale has been used for the runtime to better demonstrate the difference in the behaviours. Since the baseline method performs exhaustive subspace search, its runtime is the same across all values of δ. For CSMiner and CSMiner-BPR, as δ decreases, their runtime increases exponentially. However, the heuristic pruning techniques implemented in CSMiner and CSMiner-BPR accelerate the search substantially. CSMiner-BPR is slightly more efficient than CSMiner.

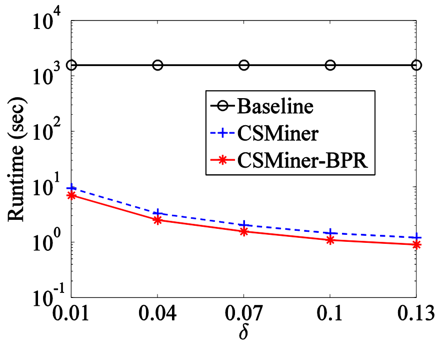


Figure 29 Scalability test with δ ($k = 10, \alpha = 0.8$)

Figure 30 shows the runtime with respect to the data set size. As can be observed, the pruning techniques can achieve a roughly linear runtime. Both CSMiner and CSMiner-BPR are considerably faster than the baseline method and CSMiner-BPR is slightly more efficient than CSMiner.

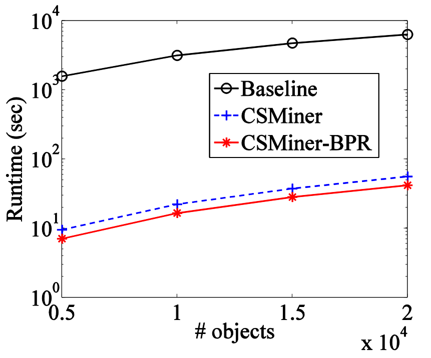


Figure 30 Scalability test with data set size ($k = 10, \delta = 0.01, \alpha = 0.8$)

Figure 31 shows the runtime with respect to the dimensionality of the data set. As the dimensionality increases, more candidate subspaces are generated and runtime increases exponentially. Both CSMiner and CSMiner-BPR are considerably faster than the baseline method and CSMiner-BPR is slightly more efficient than CSMiner.

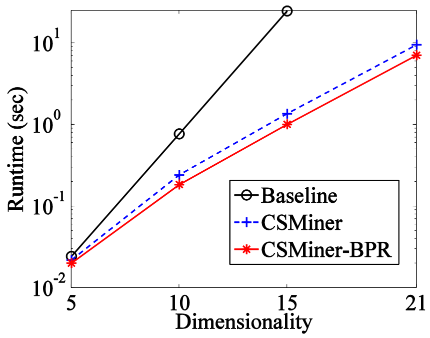


Figure 31 Scalability test with dimensionality ($k = 10, \delta = 0.01, \alpha = 0.8$)

CSMiner-BPR employs a user defined parameter α to define the ϵ-*neighbourhood*. Table 18 lists the average runtime of CSMiner-BPR for a query object with respect α to for each data set. The runtime of CSMiner-BPR is not sensitive to α in general. The experiments show the shortest runtime of CSMiner-BPR can be obtained when α is in $[0.6, 1.0]$.

Table 18 Average runtime of CSMiner-BPR with α ($k = 10, \delta = 0.01$)

Data Set	Average Runtime (milliseconds)				
	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$	$\alpha = 1.2$	$\alpha = 1.4$
BCW	20.97	20.14	17.76	16.32	15.69
CMSC	11446.20	11643.50	12915.10	14125.00	15210.20
Glass	16.13	15.83	15.62	15.69	15.76
PID	4.21	4.17	4.23	4.25	4.29
Waveform	6807.10	7102.30	7506.70	7874.70	8183.70
Wine	18.51	18.16	18.42	18.69	19.12

Figure 32 shows the relative runtime of CSMiner-BPR with respect to k for each data set. It illustrates that CSMiner-BPR is linearly scalable as the value of k increases.

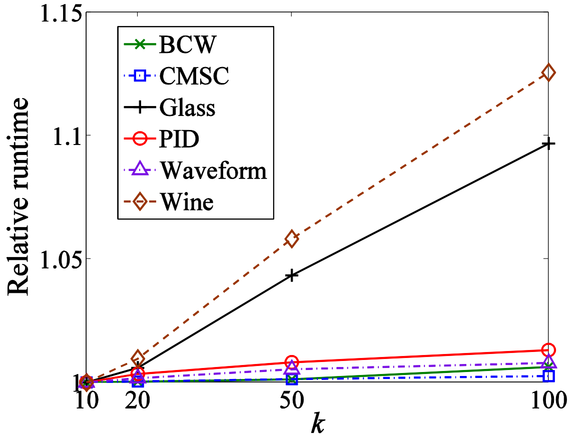


Figure 32 Relative performance of CSMiner-BPR ($\boldsymbol{\delta = 0.01, \alpha = 0.8}$)

Sensitivity to the Bandwidth

To test the sensitivity of the top-k contrast subspaces with respect to the bandwidth value, we define the similarity measure for two lists of top-k contrast subspaces.

For any two subspaces S_1 and S_2, we measure the similarity between S_1 and S_2 by the Jaccard similarity coefficient denoted by $Jaccard(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Given a positive integer r, let $\mathbb{P}^r$ be the set of all permutations of the set $\{i | 1 \leq i \leq r\}$. Thus, $|\mathbb{P}^r| = r!$. For permutation $P \in \mathbb{P}^r$, we denote the j-th $(1 \leq j \leq r)$ element in P by $P[i]$. For example, by writing each permutation as a tuple, we have $\mathbb{P}^3 = \{(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)\}$. Suppose $P = (2,3,1)$, then $P[2] = 3$.

Given two ranked lists of top-k contrast subspaces ℓ_1 and ℓ_2, without loss of generality, we follow the definition of **average overlap** (Webber et al., 2010), also referred to as **average accuracy** (Wu et al., 2003), or **intersection metric** (Fagin et al., 2003), which derives the similarity measure by averaging the overlaps of two ranked lists at each rank in order to measure the similarity between ℓ_1 and ℓ_2. Additionally, in consideration for the fact that each subspace in a list is a set of dimensions, we incorporate the Jaccard similarity coefficients into the overlap calculation. Specifically, let $\ell_1[i]$ be the element (subspace) at rank $i (1 \leq i \leq k)$ in list ℓ_1. The **agreement** of lists ℓ_1 and ℓ_2 at rank $r (1 \leq r \leq k)$, denoted by $Agr(\ell_1, \ell_2, r)$ is:

$$Agr(\ell_1, \ell_2, r) = \frac{1}{r}\max\{\sum_{i=1}^{r} Jaccard\big(\ell_1[P_1[i]], \ell_2[P_2[i]]\big) | P_1, P_2 \in \mathbb{P}^r\}.$$

Then the similarity between ℓ_1 and ℓ_2, denoted by $Sim(\ell_1, \ell_2)$ is:

$$Sim(\ell_1, \ell_2) = \frac{1}{k}\sum_{r=1}^{k} Agr(\ell_1, \ell_2, r).$$

Clearly, $0 \leq Sim(\ell_1, \ell_2) \leq 1$. The larger the value of $Sim(\ell_1, \ell_2)$, the more similar ℓ_1 and ℓ_2 are.

Given a set of objects O and a query object q, to find top-k contrast subspaces for q with respect to O by $\mathrm{CSMiner}$, we fix the bandwidth value $h_S = \sigma_S \cdot h_{S\_opt}$ and use the Gaussian kernel function to estimate the subspace likelihood of q with respect to O in subspace S. We then vary the bandwidth value from $0.6h_S$ to $1.4h_S$ for density estimation in S. Let ℓ_{h_S} be the top-k contrast subspaces computed using the default bandwidth value h_S and $\ell_{\widetilde{h_S}}$ be the top-k contrast subspaces computed using other bandwidth values. For each object $q \in O$, we discover top inlying contrast subspaces and top outlying contrast subspaces of q by $\mathrm{CSMiner}$ using different bandwidth values. Figure 33 illustrates the average value of $Sim(\ \ell_{h_S}, \ell_{\widetilde{h_S}})$ of inlying contrast subspaces with respect to k and Figure 34 illustrates the average value of $Sim(\ \ell_{h_S}, \ell_{\widetilde{h_S}})$ of outlying contrast subspaces with respect to k. From the results, we can see that the contrast subspaces computed using different bandwidth values are similar in most data sets. As expected, using a bandwidth whose value is closer to h makes less difference. Finally, we also observe that with increasing k, the value of $Sim(\ \ell_{h_S}, \ell_{\widetilde{h_S}})$ slightly increases.

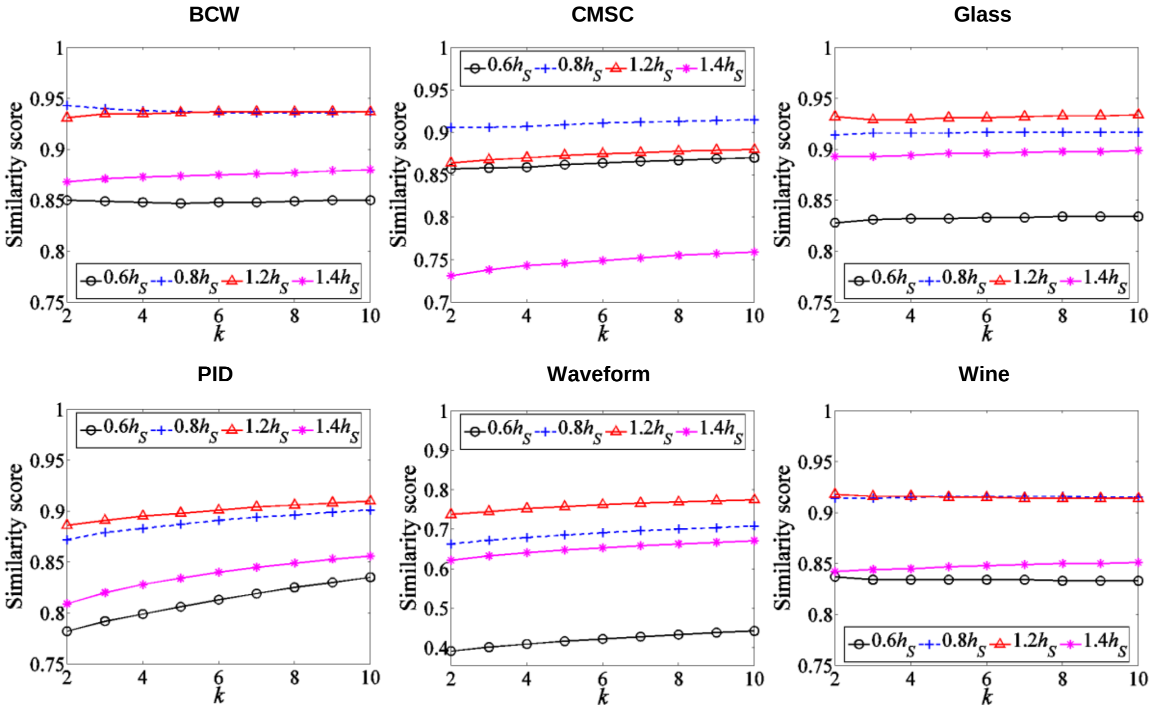


Figure 33 Similarity scores of inlying contrast subspaces using different bandwidth values with respect to k ($\delta = 0.001$)

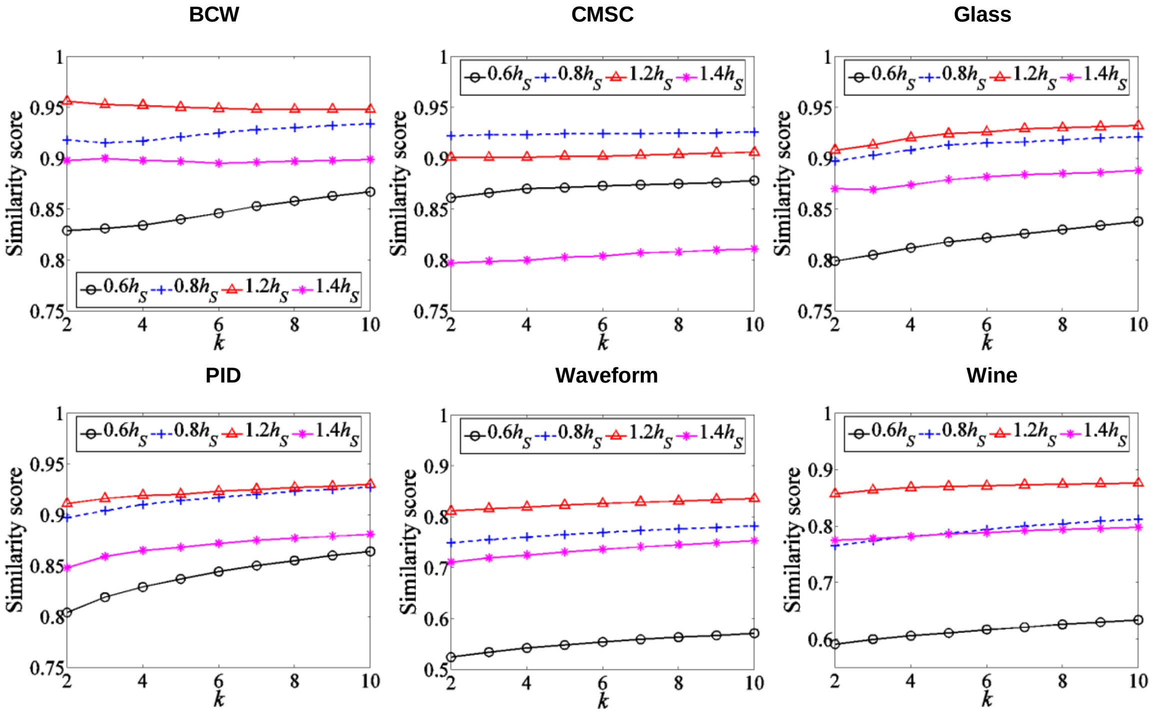


Figure 34 Similarity scores of outlying contrast subspaces using different bandwidth values with respect to k ($\delta = 0.001$)

Comparison with Epanechnikov Kernel

Alternative to Gaussian kernel (Equation (5.1.2)) for multivariate kernel density estimation is the Epanechnikov kernel:

$$K_e(x) = \begin{cases} \frac{1}{2} c_d{}^{-1}(d+2)(1 - x^T x) & \text{if } x^T x < 1 \\ 0 & \text{otherwise} \end{cases}$$

where c_d is the volume of the unit d-dimensional sphere and can be expressed by making use of the Gamma function:

$$c_d = \frac{\pi^{\frac{d}{2}}}{\Gamma(1 + \frac{d}{2})} = \begin{cases} \dfrac{\pi^{\frac{d}{2}}}{\left(\frac{d}{2}\right)!} & \text{if } d \geq 0 \text{ is even} \\ \dfrac{\pi^{\lfloor \frac{d}{2} \rfloor} 2^{\lceil \frac{d}{2} \rceil}}{d!!} & \text{if } d \geq 0 \text{ is odd} \end{cases}$$

where is $d!!$ the double factorial.

Plugging $K_e(x)$ into Equation (5.1.1), the density of a query object q for a set of objects O in subspace S can be estimated as:

$$\hat{f}_S(q,O) = \frac{1}{|O|h_S^{|S|}} \sum_{o \in O \wedge \frac{dist_S(q,o)^2}{h_S^2} < 1} (\frac{1}{2} c_{|S|^{-1}} (|S| + 2)(1 - \frac{dist_S(q,o)^2}{h_S^2})) \qquad (5.1.6)$$

where h_S is the bandwidth for subspace S. We calculate h_S as $h_S = \sigma_S \cdot h_{S\_opt}$.

As Silverman suggested (1986), σ_S is a single scale parameter that equals to the root of the average marginal variance in S and h_{S_opt} is the optimal bandwidth value which equals to $A(K)|O|^{\frac{-1}{(|S|+4)}}$ where $A(K) = \{8_{c_{|S|}}{}^{-1}(|S| + 4)(2\sqrt{\pi})^{|S|}\}^{\frac{1}{|S|+4}}$ for the Epanechnikov kernel. For $\mathrm{CSMiner}$, given a subspace S, let $\mathcal{S}$ be the set of descendants of S in the subspace set enumeration tree in the descending order of standard deviation. Then, $L_S(q|O_+)$ and $L_S(q|O_-)$ can be computed by Equation (5.1.6) and $L_S^*(q|O_+) = \frac{1}{|O|(\sigma'_{min} h'_{opt\_min})^{\tau}} \sum_{o \in O \wedge \frac{dist_S(q,o)^2}{(\sigma S h'_{opt\_max})^2} < 1} \left(\frac{1}{2} c_{\mathcal{S}}^{min^{-1}} (d_{\mathcal{S}}^{max} + 2)(1 - \frac{dist_S(q,o)^2}{\sigma S h'_{opt\_max}}) \right)$ where $d_{\mathcal{S}}^{max} = \max\{|S'| \,|\, S' \in S\}$, $c_S^{min} = \min\{c_d |\, |S| < d \leq d_S^{max}\}$. Using the Epanechnikov kernel, $\hat{f}_S(q, O_-) = 0$ if for any object $o \in O_-$, $\frac{dist_S(q,o)^2}{h_S^2} < 1$. Accordingly, $LC_S(q) = \frac{\hat{f}_S(q,O_+)}{\hat{f}_S(q,O_-)} = +\infty$. Given a data set O (consisting of O_+ and O_-), the set of objects whose maximum likelihood contrast computed using the Epanechnikov kernel is infinity is $O_E^{+\infty} = \{o \in O | \exists S \;\; s.t. \;\; LC_S(o) = +\infty\}$.

Let ℓ_G be the top-k contrast subspaces computed using the Gaussian kernel and ℓ_E be the top-k contrast subspaces computed using Epanechnikov kernel. For each query object $q \in O$, we discover top-10 inlying contrast subspaces and top-10 outlying contrast subspaces of q using the Gaussian and Epanechnikov kernels and compute $Sim(\ell_G, \ell_E)$ in each data set. For subspaces whose likelihood contrast values are infinity ($LC_S(q) = +\infty$), we sort them by $\hat{f}_S(q, O_+)$ in the descending order. Table 19 and 20 list the minimum, the maximum, and the average values of $Sim(\ell_G, \ell_E)$ as well as the ratio of $|O_E^{+\infty}|$ to $|O|$.

Table 19 Similarity between top-10 inlying contrast subspaces using different kernel functions in data set $\boldsymbol{O}$ ($\boldsymbol{\delta = 0.001}$)

Data Set $\boldsymbol{O}$	$\boldsymbol{Sim(\ell_G, \ell_E)}$			$\boldsymbol{\frac{\lvert O_E^{+\infty}\rvert}{\lvert O\rvert}}$
	Min	Max	Avg	
BCW	0.168	0.980	0.539	590/683=0.864
CMSC	0.066	0.826	0.391	540/540=1.000
Glass	0.242	0.984	0.814	76/214=0.355
PID	0.620	1.000	0.924	1/768=0.001
Waveform	0.189	0.981	0.690	2532/5000=0.506
Wine	0.159	0.993	0.670	145/178=0.815

Table 20 Similarity between top-10 outlying contrast subspaces using different kernel functions in data set $\boldsymbol{O}$ ($\boldsymbol{\delta = 0.001}$)

Data Set $\boldsymbol{O}$	$\boldsymbol{Sim(\ell_G, \ell_E)}$			$\boldsymbol{\frac{\lvert O_E^{+\infty}\rvert}{\lvert O\rvert}}$
	Min	Max	Avg	
BCW	0.239	1.000	0.916	67/683=0.098
CMSC	0.174	0.926	0.614	540/540=1.000
Glass	0.358	1.000	0.906	16/214=0.075
PID	0.655	1.000	0.938	1/768=0.001
Waveform	0.364	0.998	0.820	894/5000=0.179
Wine	0.209	1.000	0.804	40/178=0.225

From the results in tables 19 and 20, we can see that $Sim(\ell_G, \ell_E)$ is related to $\frac{|O_E^{+\infty}|}{|O|}$. Specifically, the smaller the value of $\frac{|O_E^{+\infty}|}{|O|}$, the more similar ℓ_G are ℓ_E. For example, when mining inlying contrast subspaces, the values of $\frac{|O_E^{+\infty}|}{|O|}$ for BCW, CMSC, Waveform and Wine are larger than 0.5 which is larger than the values of $\frac{|O_E^{+\infty}|}{|O|}$ for PID and Glass while the values of $Sim(\ell_G, \ell_E)$ are smaller for BCW, CMSC, Waveform and Wine than those for PID and Glass. When mining outlying contrast subspaces, the values of $\frac{|O_E^{+\infty}|}{|O|}$ are less than 0.1 for BCW, Glass and PID while the values of $Sim(\ell_G, \ell_E)$ for these data sets are over 0.9.

Finally, we compute $Sim(\ell_G, \ell_E)$ in $O \backslash O_E^{+\infty}$ for each data set except CMSC since for CMSC, $O \backslash O_E^{+\infty} = \emptyset$. From the results shown in Table 21 and 22, we can see that ℓ_G is more similar to ℓ_E when we do not consider the objects whose maximum likelihood contrast is infinity.

Table 21 Similarity between top-10 inlying contrast subspaces using different kernel functions in data set $\boldsymbol{O}\backslash\boldsymbol{O}_{\boldsymbol{E}}^{+\infty}$ ($\boldsymbol{\delta = 0.001}$)

Data Set $\boldsymbol{O}\backslash$ $\boldsymbol{O}_{\boldsymbol{E}}^{+\infty}$	$\boldsymbol{Sim(\ \ell_G, \ell_E)}$			$\lvert O\backslash O_E^{+\infty}\ \rvert$
	Min	Max	Avg	
BCW	0.643	0.980	0.922	93
Glass	0.720	0.984	0.929	138
PID	0.620	1.000	0.924	767
Waveform	0.324	0.981	0.754	2468
Wine	0.527	0.988	0.904	33

Table 22 Similarity between top-10 outlying contrast subspaces using different kernel functions in data set $\boldsymbol{O}\backslash\boldsymbol{O}_{\boldsymbol{E}}^{+\infty}$ ($\boldsymbol{\delta = 0.001}$)

Data Set $O\backslash$ $O_E^{+\infty}$	$\boldsymbol{Sim(\ \ell_G, \ell_E)}$			$\lvert O\backslash O_E^{+\infty}\ \rvert$
	Min	Max	Avg	
BCW	0.561	1.000	0.934	616
Glass	0.629	1.000	0.925	198
PID	0.655	1.000	0.938	767
Waveform	0.437	0.998	0.836	4106
Wine	0.482	1.000	0.863	138

5.2. Outlying Aspects

In a multidimensional dataset, given a query object q, we want to find a subspace in which q is most unusual or outlying. By mining outlying aspects, we seek an answer for "In what context does the organization stand out most?", for example. Using the Figure 1 example, Figure 35 illustrates two example subspaces.

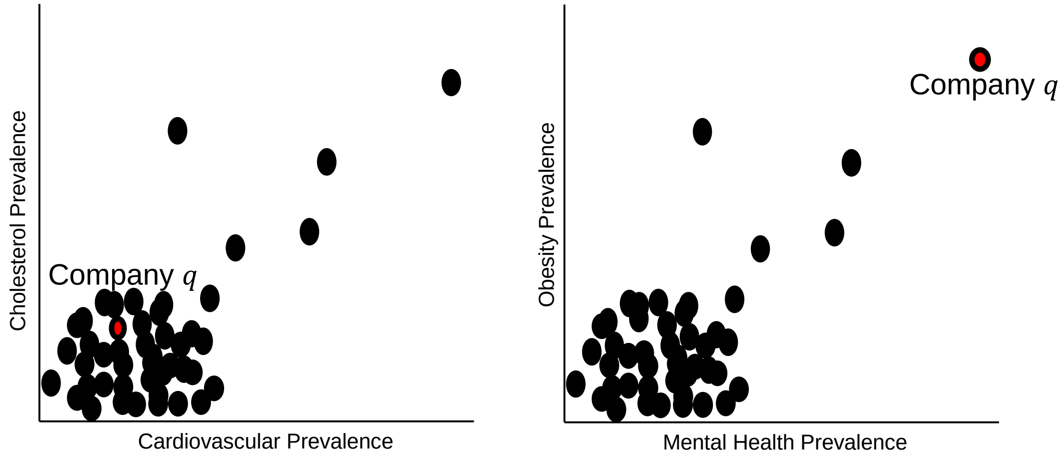


Figure 35 Example Subspaces

In the cholesterol-cardiovascular subspace (on the left), the company q (red dot) is part of the "normal" population, meaning this company's performance is similar to many others' for cholesterol and cardiovascular and the gap is marginal. On the other hand, in the obesity-mental health subspace (on the right), q is significantly outlying, meaning that its rate of obesity and mental health is unusual and the performance gap is material. The insight this example brings may be that the company q needs to develop a program to reduce the rate of obesity and/or mental health across its workforce. While this example only shows 2 dimensions for visual simplicity, mining outlying aspects detects top-k (where k can be any number of dimensions) most prominent dimensions in which q is an outlier.

5.2.1. Rank Statistics

In order to identify top-k subspaces in which a query object q is outlying most, the ability to compare the outlyingness degree of q in different subspaces is required. We use rank statistics as a vehicle for comparison.

Problem Definition

Let $D = \{D_1, \dots, D_d\}$ be a d-dimensional space where the domain of D_i is $\mathbb{R}$, a set of real numbers. A subspace $S \subseteq D$ $(S \neq 0)$ is a subset of D and D is also referred to as the **full space**. The value of an object o in dimension D_i $(1 \leq i \leq d)$ is denoted as $o.D_i$.

For a subspace $S = \{D_{i_1}, \ldots, D_{i_l}\} \subseteq D$, the projection of o in S is $o^S = \{o.D_{i_1}, \ldots, o.D_{i_l}\}$. The dimensionality of S, denoted by $|S|$, is the number of dimensions in S.

In a subspace $S \subseteq D$, we assume that we can define a measure of **outlyingness degree**, $OutDeg(\cdot)$, such that for each objet $o \in O$, $OutDeg(o)$ measures the outlyingness of o. Without loss of generality we assume that the lower the $OutDeg(o)$, the more outlying the object o. In this thesis, we consider a generative model; that is, the set of objects O is generated (i.e. sampled) from an often unknown probability distribution. Accordingly, we can use the probability density of an object o, denoted by $f(o)$, as the equivalent to $OutDeg(o)$. The smaller the value of $f(o)$, the more outlying the object o.

How can we compare the outlying degree of an object in different subspaces? We unfortunately cannot compare them directly since the probability density values depend on the properties of specific subspaces, such as, their scales. For example, it is well known that probability density tends to be low in subspaces of higher dimensionality since such subspaces often have a larger volume and thus sparser.

To address this issue, we consider the use of rank statistics. In a subspace S, we rank all objects in O in their outlyingness degree ascending order. For an object $o \in O$, we denote by:

$$rank_S(o) = |\{o' | o' \in O, OutDeg(o') < OutDeg(o)\}| + 1 \qquad (5.2.1)$$

the **outlyingness rank** of o in subspace S. The smaller the rank value, the more outlying the object is compared to the other objects in O in subspace S. We can compare the outlyingness of an object o in two subspaces S_1 and S_2 using $rank_{S_1}(o)$ and $rank_{S_2}(o)$. Object o is more outlying in the subspace where it has the smaller rank. In Equation (5.2.1), for objects with the same outlyingness degree (probability density value), their outlyingness ranks are the same.

Suppose for object o, there are two subspaces S and S' such that $S \subset S'$ and $rank_S(o) = rank_{S'}(o)$. Since S is more general than S', S is more significant in manifesting the outlyingness of o at the rank of $rank_S(o)$ relative to the other objects in the data set. Consequently, S' is redundant given S, in terms of outlying aspects.

In high dimensional subspaces where the values of probability densities of objects are very small, comparing the ranks may not be reliable since the subtle differences in values may be due to noise or sensitivity to parameter settings in the density estimation. Further to note that high dimensional subspaces may not be interesting since the results are hard to understand. Thus, we assume a maximum dimensionality threshold, $\ell > 0$ and consider only the subspaces whose dimensionalities are not greater than ℓ.

Definition 1 (Problem definition). Given a set of objects O in a multi-dimensional space D, a query object $q \in O$ and a maximum dimensionality threshold $0 < \ell \leq |D|$, a subspace $S \subseteq D (0 < |S| \leq \ell)$ is called a **minimal outlying subspace** of q if

1. (Rank minimality) there does not exist another subspace $S' \subseteq D (S' \neq \emptyset)$ such that $rank_{S'}(q) = rank_S(q)$; and
2. (Subspace minimality) there does not exist another subspace $S'' \subseteq S$ such that $rank_{S''}(q) = rank_S(q)$.

The problem of minimal outlying subspace is to find the minimal outlying subspaces of q. Given a query object q, there exists at least one and may be more than one minimal outlying subspace.

We use KDE (Breiman et al., 1977) to estimate the density given a set of objects O. To reduce sensitivity to outliers, we employ Härdle's rule of thumb (1990) instead of Silverman's and set the bandwidth:

$$h = 1.06 \min\{\sigma, \frac{R}{1.34}\} n^{-\frac{1}{5}} \quad (5.5.2)$$

where $R = X_{[0.75n]} - X_{[0.25n]}$, and $X_{[0.25n]}$ and $X_{[0.75n]}$ respectively are the first and the third quartiles.

For d-dimensional $(d \geq 2)$, $o = (o.D_1, \ldots, o.D_d)^T$ and $o_i = (o_i.D_1, \ldots, o_i.D_d)^T$ $(1 \leq i \leq n)$. Then the probability density of f at point $o \in \mathbb{R}^d$ can be estimated by:

$$\hat{f}_H(o) = \frac{1}{n} \sum_{i=1}^{n} K_H(o - o_i)$$

where H is a bandwidth matrix. The product kernel, consisting of the product of one-dimensional kernels, is a good choice for a multivariate kernel density estimator (Scott, 1992; Härdle et al., 2004). Hence, we have:

$$\hat{f}_H(o) = \frac{1}{n\prod_{j=1}^{d} h_{D_j}} \sum_{i=1}^{n} \left\{ \prod_{j=1}^{d} K\left(\frac{o.D_j - o_i.D_j}{h_{D_j}}\right) \right\} \tag{5.2.3}$$

where h_{D_i} is the bandwidth of dimension $D_i (1 \leq i \leq d)$. We use Gaussian kernel and the distance between two objects is measured by Euclidean distance. Thus, the kernel function is:

$$K\left(\frac{o-o_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(o-o_j)^2}{2h^2}} \tag{5.2.4}$$

Plugging Equation (5.2.4) into (5.2.3), the density of a query object $q \in O$ in subspace S can be estimated as:

$$\hat{f}_S(q) = \hat{f}_S(q^S) = \frac{1}{n(2\pi)^{\frac{|S|}{2}} \prod_{D_i \in S} h_{D_i}} \sum_{o \in O} e^{-\sum_{D_i \in S} \frac{(q.D_i - o.D_i)^2}{2h_{D_i}^2}} \tag{5.2.5}$$

Since we are only interested in the rank of q (i.e. $rank_S(q)$) and:

$$c = \frac{1}{n(2\pi)^{\frac{|S|}{2}} \prod_{D_i \in S} h_{D_i}} \tag{5.2.6}$$

is a factor common to every object in subspace S and does not affect the ranking at all, we can rewrite Equation (5.2.5) as:

$$\hat{f}_S(q) \sim \tilde{f}_S(q) = \sum_{o \in O} e^{-\sum_{D_i \in S} \frac{(q.D_i - o.D_i)^2}{2h_{D_i}^2}} \tag{5.2.7}$$

where "~" means equivalence for ranking. For clarity, we refer to $\tilde{f}_S(q)$ as **quasi-density** of q in S. Using $\tilde{f}_S(q)$ instead of $\hat{f}_S(q)$ not only simplifies the description but also saves the computational cost for calculating $rank_S(q)$.

Proposition 1 (Invariance). Given a set of objects O in space $S = \{D_1, \dots, D_d\}$, define a linear transformation $g(o) = (a_1 o.D_1 + b_1, \dots, a_d o.D_d + b_d)$ for any $o \in O$ where $a_1, \dots, a_d$ and $b_1, \dots, b_d$ are real numbers. Let $O' = \{g(o) | o \in O\}$ be the transformed data set. For any objects $o_1, o_2 \in O$ such that $\tilde{f}_S(o_1) > \tilde{f}_S(o_2)$ in O, $\tilde{f}_S(g(o_1)) > \tilde{f}_S(g(o_2))$ if the product kernel is used and the bandwidths are set using Equation (5.2.2).

Proof. For any dimension $D_i \in S (1 \le i \le d)$, the mean value of $\{o.D_i | o \in O\}$, denoted by μ_i, is $\frac{1}{|O|}\sum_{o \in O} o.D_i$, the standard deviation of $\{o.D_i | o \in O\}$, denoted by σ_i, is $\sqrt{\frac{1}{|O|}\sum_{o \in O}(o.D_i - \mu_i)^2}$, and the bandwidth of $D_i (h_i)$ is $1.06\min\{\sigma_i, \frac{R}{1.34}\}|O|^{-\frac{1}{5}}$ where R is the difference between the first and the third quartiles of O in D_i. We perform the linear transformation $g(o).D_i = a_i o.D_i + b_i$ for any $o \in O$. Then, the mean value of $\{g(o).D_i | o \in O\}$ is $\frac{1}{|O|}\sum_{o \in O}(a_i o.D_i + b_i) = a_i \mu_i + b_i$ and the standard deviation of $\{g(o).D_i | o \in O\}$ is $\sqrt{\frac{1}{|O|}\sum_{o \in O}(a_i o.D_i + b_i - a_i \mu_i - b_i)^2} = a_i \sqrt{\frac{1}{|O|}\sum_{o \in O}(o.D_i - \mu_i)^2} = a_i \sigma_i$. Thus, the bandwidth of D_i is $1.06\min\{a_i \sigma_i, \frac{a_i R}{1.34}\}|O|^{-\frac{1}{5}}$ after the linear transformation. As the distance between two objects in D_i is also enlarged by a_i, the quasi-density calculated by Equation (5.2.7) remains unchanged. Thus, the ranking is invariant under linear transformation.

5.2.2. Mining Methods

Baseline Method

Using quasi-density estimation (Equation (5.2.7)), we can develop a baseline algorithm for computing the outlyingness rank in a subspace S (Algorithm 1). The baseline method estimates the quasi-density of each object in a data set and ranks them. For the total number of objects n, the baseline method has to compute the distance between every pair of objects in every dimension of S. Thus, the time complexity is $O(n^2|S|)$ in each subspace S.

Algorithm 1 The baseline algorithm

```
Input: $O$: a set of objects, $q$: query object $\in O$, $S$: subspace
Output: $rank_S(q)$
for each object $o \in O$ do
    compute $\tilde{f}_S(o)$ using Equation (5.2.7);
end for
return $rank_S(q) = |\{o | o \in O, \tilde{f}_S(o) < \tilde{f}_S(q)\}|+1$;
```

OAMiner *Framework*

To reduce computational cost, we propose OAMiner (for Outlying Aspect Miner) in Algorithm 2.

Algorithm 2 OAMiner

```
Input: $O$: a set of objects, $q$: query object $\in O$
Output: a set of minimal outlying subspaces for $q$
initialize $r_{best} \leftarrow |O|$ and $Ans \leftarrow \emptyset$;
remove $D_i$ from $D$ if the values of all objects in $D_i$ are identical;
compute $rank_{D_i}(q)$ in each dimension $D_i \in D$;
sort all dimensions in $rank_{D_i}(q)$ ascending order;
for each subspace $S$ searched by traversing the set enumeration tree in a depth-first manner do
    compute $rank_S(q)$;
    if $rank_S(q) < r_{best}$ then
        $r_{best} \leftarrow rank_S(q), Ans \leftarrow \{S\}$;
    end if
    if $rank_S(q) = r_{best}$ and $S$ is minimal then
        $Ans \leftarrow Ans \cup \{S\}$;
    end if
    if a subspace pruning conditions is true then
        prune all descendants of $S$
    end if
end for
return $Ans$;
```

As a first step, OAMiner removes the dimensions where all values of objects are identical since no object is outlying in such dimensions. As a result, the standard deviation of all dimensions involved in outlying aspect mining are greater than 0.

To ensure that $\mathrm{OAMiner}$ finds the most outlying subspaces, we need to enumerate all possible subspaces in a systematic way. We again use the set enumeration tree introduced in section 5.1.3 (Figure 25).

$\mathrm{OAMiner}$ searches subspaces by traversing the subspace enumeration tree in a depth-first manner. Given a set of objects O, a query object $q \in O$ and a subspace S, if $rank_S(q) = 1$, then every descendant of S cannot be a minimal outlying subspace and thus can be pruned.

Pruning Rule 1. If $rank_S(q) = 1$, according to the dimensionality minimality condition (Definition 1), all descendants of S can be pruned.

In the case of $rank_S(q) > 1$, $\mathrm{OAMiner}$ prunes subspaces according to the current best rank of q in the search process. Heuristically, we want to find subspaces where the query object q has a low rank early on so that the pruning technique is more effective. Consequently, we compute the outlyingness rank of q in each dimension D_i and order dimensions in the ascending order of $rank_{D_i}(q)$.

In general, the outlyingness rank does not have any monotonicity with respect to subspaces; that is, for subspaces $S_1 \subseteq S_2$, neither $rank_{S_1}(q) \leq rank_{S_2}(q)$ nor $rank_{S_1}(q) \geq rank_{S_2}(q)$ holds.

Example 1. Given a set of objects $O = \{o_1, o_2, o_3, o_4\}$ with 2 numeric attributes D_1 and D_2, the values of each object in O are listed in Table 23. Using Equation (5.2.7), we estimate the quasi-density values of each object for different subspaces (Table 24). We can see that $\tilde{f}_{\{D_1\}}(o_2) > \tilde{f}_{\{D_1\}}(o_4)$ and $\tilde{f}_{\{D_2\}}(o_2) > \tilde{f}_{\{D_2\}}(o_4)$ which indicate $rank_{\{D_1\}}(o_2) > rank_{\{D_1\}}(o_4)$ and $rank_{\{D_2\}}(o_2) > rank_{\{D_2\}}(o_4)$. However, for subspaces $\{D_1, D_2\}$, since $\tilde{f}_{\{D_1,D_2\}}(o_2) < \tilde{f}_{\{D_1,D_2\}}(o_4)$, $rank_{\{D_1,D_2\}}(o_2) < rank_{\{D_1,D_2\}}(o_4)$.

Table 23 A numeric data set example

$object$	$o_i.D_1$	$o_i.D_2$
o_1	14.23	1.50
o_2	13.20	1.78
o_3	13.16	2.31
o_4	14.37	1.97

Table 24 quasi-density values of objects in Table 23

$object$	$\tilde{f}_{\{D_1\}}(o_i)$	$\tilde{f}_{\{D_2\}}(o_i)$	$\tilde{f}_{\{D_1,D_2\}}(o_i)$
o_1	2.229	1.832	1.305
o_2	2.220	2.529	1.300
o_3	2.187	1.626	1.185
o_4	2.113	2.474	1.314

A further challenge is that the probability density itself does not have any monotonicity with respect to subspaces either. For subspaces $S_1 \subseteq S_2$, according to Equation (5.2.5), we have:

$$\frac{\hat{f}_{S_1}(q)}{\hat{f}_{S_2}(q)} = \frac{\sum_{o\in O} e^{-\sum_{D_i\in S_1}\frac{(q.D_i-o.D_i)^2}{2h_{D_i}^2}}}{n(2\pi)^{\frac{|S_1|}{2}}\prod_{D_i\in S_1} h_{D_i}} / \frac{\sum_{o\in O} e^{-\sum_{D_i\in S_2}\frac{(q.D_i-o.D_i)^2}{2h_{D_i}^2}}}{n(2\pi)^{\frac{|S_2|}{2}}\prod_{D_i\in S_2} h_{D_i}}$$

$$= (2\pi)^{\frac{|S_1|-|S_2|}{2}} \prod_{D_i\in S_2\setminus S_1} h_{D_i} \frac{\sum_{o\in O} e^{-\sum_{D_i\in S_1}\frac{(q.D_i-o.D_i)^2}{2h_{D_i}^2}}}{\sum_{o\in O} e^{-\sum_{D_i\in S_2}\frac{(q.D_i-o.D_i)^2}{2h_{D_i}^2}}}$$

Since $S_1 \subseteq S_2$, $\frac{\sum_{o\in O} e^{-\sum_{D_i\in S_1}\frac{(q.D_i-o.D_i)^2}{2h_{D_i}^2}}}{\sum_{o\in O} e^{-\sum_{D_i\in S_2}\frac{(q.D_i-o.D_i)^2}{2h_{D_i}^2}}} \geq 1$ and $(2\pi)^{\frac{|S_1|-|S_2|}{2}} \geq 1$. However, in the case $\prod_{D_i\in S_2\setminus S_1} h_{D_i} < 1$, there is no guarantee that $\frac{\hat{f}_{S_1}(q)}{\hat{f}_{S_2}(q)} > 1$ always holds. Consequently, neither $\hat{f}_{S_1}(q) \leq \hat{f}_{S_2}(q)$ nor $\hat{f}_{S_1}(q) \geq \hat{f}_{S_2}(q)$ holds in general.

A Bounding-Pruning-Refining Method

Bounding Probability Density

To obtain rank statistics, $\mathrm{OAMiner}$ needs to compare the density of the query object with the densities of other objects. In order to speed up density estimation, we exploit the observation that the contributions to the density of an object from remote objects are small; thus, the density of an object can be bounded. Similar to the concept of ϵ-$neighbourhood$ with $\mathrm{CSMiner}$, we can derive upper and lower bounds of the density of an object using a neighbourhood.

Given objects $o, o' \in O$, subspace S and a subset $O' \subseteq O$, we denote by $dc_S(o, o')$ the quasi-density contribution of o' to o in S and $\tilde{f}_S^{O'}(o)$ the sum of quasi-density contributions of objects in O' to o. That is:

$$dc_S(o, o') = e^{-\sum_{D_i \in S} \frac{(o.D_i - o\prime.D_i)^2}{2h_{D_i}^2}}$$

$$\tilde{f}_S^{O'}(o) = \sum_{o\prime \in O\prime} e^{-\sum_{D_i \in S} \frac{(o.D_i - o\prime.D_i)^2}{2h_{D_i}^2}}$$

To estimate the bounds of $\tilde{f}_S(o)$ efficiently, we define two kinds of neighbourhood. For an object $o \in O$, a subspace S and $\{\epsilon_{D_i} | \epsilon_{D_i} > 0, D_i \in S\}$, the ϵ-$tight\ neighbourhood$ of o in S, denoted by $TN_S^{\epsilon,o}$, is $\{o' \in O | \forall D_i \in S, |o.D_i - o'.D_i| \leq \epsilon_{D_i}\}$, the ϵ-$loose\ neighbourhood$ of o in S, denoted by $LN_S^{\epsilon,o}$, is $\{o' \in O | \exists D_i \in S, |o.D_i - o'.D_i| \leq \epsilon_{D_i}\}$. Based on the definitions of $TN_S^{\epsilon,o}$ and $LN_S^{\epsilon,o}$, we depict the following properties.

Property 1. $TN_S^{\epsilon,o} \subseteq LN_S^{\epsilon,o}$.

Property 2. $TN_S^{\epsilon,o} = LN_S^{\epsilon,o} \text{if } |S| = 1$.

Using $TN_S^{\epsilon,o}$ and $LN_S^{\epsilon,o}$, O can be divided into three disjoint subsets: $TN_S^{\epsilon,o}$, $LN_S^{\epsilon,o} \setminus TN_S^{\epsilon,o}, O \setminus LN_S^{\epsilon,o}$. For any object $o' \in O$, we obtain a lower bound and an upper bound of $dc_S(o, o')$ as follows:

Theorem 1 (Single quasi-density contribution bounds). Given an object $o \in O$, a subspace S and $\{\epsilon_{D_i} | \epsilon_{D_i} > 0, D_i \in S\}$, then:

for any object $o' \in TN_S^{\epsilon,o}, dc_S^{\epsilon} \leq dc_S(o, o') \leq dc_S^{max}(o)$;

for any object $o' \in LN_S^{\epsilon,o} \setminus TN_S^{\epsilon,o}, dc_S^{min}(o) \leq dc_S(o, o') \leq dc_S^{max}(o)$;

for any object $o' \in O \setminus LN_S^{\epsilon,o}, dc_S^{min}(o) \leq dc_S(o, o') \leq dc_S^{\epsilon}$

where:

$$dc_S^{\epsilon} = e^{-\sum_{D_i \in S} \frac{\epsilon_{D_i}^2}{2h_{D_i}^2}}$$

$$dc_S^{max}(o) = e^{-\sum_{D_i \in S} \frac{\min_{o' \in O}\{|o.D_i - o'.D_i|\}^2}{2h_{D_i}^2}}$$

$$dc_S^{min}(o) = e^{-\sum_{D_i \in S} \frac{\max_{o' \in O}\{|o.D_i - o'.D_i|\}^2}{2h_{D_i}^2}}.$$

Proof.

1) Given an object $o' \in TN_S^{\epsilon,o}$, for any dimension $D_i \in S$, $\min_{o'' \in O}\{|o.D_i - o''.D_i|\} \leq |o.D_i - o'.D_i| \leq \epsilon_{D_i}$. Thus, $e^{-\sum_{D_i \in S} \frac{\epsilon_{D_i}^2}{2h_{D_i}^2}} \leq e^{-\sum_{D_i \in S} \frac{|o.D_i - o'.D_i|^2}{2h_{D_i}^2}} \leq e^{-\sum_{D_i \in S} \frac{\min_{o'' \in O}\{|o.D_i - o''.D_i|\}^2}{2h_{D_i}^2}}$. That is, $dc_S^{\epsilon} \leq dc_S(o, o') \leq dc_S^{max}(o)$.
2) Given an object $o' \in LN_S^{\epsilon,o} \setminus TN_S^{\epsilon,o}$, for any dimension $D_i \in S$, $\min_{o'' \in O}\{|o.D_i - o''.D_i|\} \leq |o.D_i - o'.D_i| \leq \max_{o'' \in O}\{|o.D_i - o''.D_i|\}$. Thus, $e^{-\sum_{D_i \in S} \frac{\max_{o'' \in O}\{|o.D_i - o''.D_i|\}^2}{2h_{D_i}^2}} \leq e^{-\sum_{D_i \in S} \frac{|o.D_i - o'.D_i|^2}{2h_{D_i}^2}} \leq e^{-\sum_{D_i \in S} \frac{\min_{o'' \in O}\{|o.D_i - o''.D_i|\}^2}{2h_{D_i}^2}}$. That is, $dc_S^{min}(o) \leq dc_S(o, o') \leq dc_S^{max}(o)$.
3) Given an object $o' \in O \setminus LN_S^{\epsilon,o}$, for any dimension $D_i \in S$, $\epsilon_{D_i} \leq |o.D_i - o'.D_i| \leq \max_{o'' \in O}\{|o.D_i - o''.D_i|\}$. Thus, $e^{-\sum_{D_i \in S} \frac{\max_{o'' \in O}\{|o.D_i - o''.D_i|\}^2}{2h_{D_i}^2}} \leq e^{-\sum_{D_i \in S} \frac{|o.D_i - o'.D_i|^2}{2h_{D_i}^2}} \leq e^{-\sum_{D_i \in S} \frac{\epsilon_{D_i}^2}{2h_{D_i}^2}}$. That is, $dc_S^{min}(o) \leq dc_S(o, o') \leq dc_S^{\epsilon}$.

Using the size of $TN_S^{\epsilon,o}$and $LN_S^{\epsilon,o}$, we obtain a lower and an upper bound of $\tilde{f}_S(o)$ as follows.

Corollary 1 (Bounds by neighbourhood size). For any object $o \in O$,

$$\left|TN_S^{\epsilon,o}\right| dc_S^{\epsilon} + (|O| - \left|TN_S^{\epsilon,o}\right|) dc_S^{min}(o) \leq \tilde{f}_S(o)$$

$$\tilde{f}_S(o) \leq \left|LN_S^{\epsilon,o}\right| dc_S^{max}(o) + (|O| - \left|LN_S^{\epsilon,o}\right|) dc_S^{\epsilon}$$

Corollary 1 allows us to compute the quasi-density bounds of an object without computing the quasi-density contributions of the other objects to it.

Proof. We divide O into disjoint subsets $TN_S^{\epsilon,o}$, $LN_S^{\epsilon,o}\setminus TN_S^{\epsilon,o}, O\setminus LN_S^{\epsilon,o}$. By Theorem 1, for objects belonging to $TN_S^{\epsilon,o}$, we have:

$$|TN_S^{\epsilon,o}|dc_S^{\epsilon} \leq \textstyle\sum_{o'\in TN_S^{\epsilon,o}} dc_S(o,o') \leq |TN_S^{\epsilon,o}|dc_S^{max}(o),$$

for objects belonging to $LN_S^{\epsilon,o}\setminus TN_S^{\epsilon,o}$, we have:

$$\begin{aligned}\left(\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|\right)dc_S^{min}(o) &\leq \textstyle\sum_{o'\in\ LN_S^{\epsilon,o}\setminus TN_S^{\epsilon,o}} dc_S(o,o') \\ &\leq (\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|)dc_S^{max}(o),\end{aligned}$$

for objects belonging to $O\setminus LN_S^{\epsilon,o}$, we have:

$$\left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{min}(o) \leq \textstyle\sum_{o'\in\ O\setminus LN_S^{\epsilon,o}} dc_S(o,o') < (|O| - \left|LN_S^{\epsilon,o}\right|)dc_S^{\epsilon} \text{ as:}$$

$$\begin{aligned}\tilde{f}_S(o) \leq \textstyle\sum_{o'\in O} dc_S(o,o') = \textstyle\sum_{o'\in TN_S^{\epsilon,o}} dc_S(o,o') &+ \\ \textstyle\sum_{o'\in\ LN_S^{\epsilon,o}\setminus TN_S^{\epsilon,o}} dc_S(o,o') &+ \\ \textstyle\sum_{o'\in\ O\setminus LN_S^{\epsilon,o}} dc_S(o,o').&\end{aligned}$$

Thus:

$$\begin{aligned}\tilde{f}_S(o) &\geq \left|TN_S^{\epsilon,o}\right|dc_S^{\epsilon} + \left(\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|\right)dc_S^{min}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{min}(o) \\ &= \left|TN_S^{\epsilon,o}\right|dc_S^{\epsilon} + \left(|O| - \left|TN_S^{\epsilon,o}\right|\right)dc_S^{min}(o)\end{aligned}$$

$$\begin{aligned}\tilde{f}_S(o) &\leq \left|TN_S^{\epsilon,o}\right|dc_S^{max}(o) + \left(\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|\right)dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{\epsilon} \\ &= \left|LN_S^{\epsilon,o}\right|dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{\epsilon}.\end{aligned}$$

Finally, if $LN_S^{\epsilon,o} \subset O$; that is, $O\setminus LN_S^{\epsilon,o} \neq \emptyset$, then:

$$\tilde{f}_S^{O'}(o) < \left|LN_S^{\epsilon,o}\right|dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{\epsilon}.$$

Corollary 2 (Bounds by *ϵ-tight neighbours*). For any object $o \in O$ and $O' \subseteq TN_S^{\epsilon,o}$,

$$\tilde{f}_S^{O'}(o) + \left(\left|TN_S^{\epsilon,o}\right| - |O'|\right)dc_S^{\epsilon} + (|O| - \left|TN_S^{\epsilon,o}\right|)dc_S^{min}(o) \leq \tilde{f}_S(o)$$

$$\tilde{f}_S(o) \leq \tilde{f}_S^{O'}(o) + \left(\left|LN_S^{\epsilon,o}\right| - |O'|\right)dc_S^{max}(o) + (|O| - \left|LN_S^{\epsilon,o}\right|)dc_S^{\epsilon}$$

Proof. Since $O' \subseteq TN_S^{\epsilon,o}$, for objects belonging to $O\setminus O'$, we divide them into $TN_S^{\epsilon,o}\setminus 0'$, $LN_S^{\epsilon,o}\setminus TN_S^{\epsilon,o}, O\setminus LN_S^{\epsilon,o}$. Then:

$$\begin{aligned}\tilde{f}_S(o) = \tilde{f}_S^{O'}(o) + \textstyle\sum_{o'\in TN_S^{\epsilon,o}\setminus O'} dc_S(o,o') &+ \\ \textstyle\sum_{o'\in\ LN_S^{\epsilon,o}\setminus TN_S^{\epsilon,o}} dc_S(o,o') &+\end{aligned}$$

$$\textstyle\sum_{o' \in\ O \setminus LN_S^{\epsilon,o}} dc_S(o, o').$$

By Theorem 1, for objects belonging to $TN_S^{\epsilon,o} \setminus O'$, we have:

$$\left(\left|TN_S^{\epsilon,o}\right| - |O'|\right) dc_S^{\epsilon} \leq \textstyle\sum_{o' \in\ TN_S^{\epsilon,o} \setminus O'} dc_S(o, o') \leq (\left|TN_S^{\epsilon,o}\right| - |O'|) dc_S^{max}(o),$$

for objects belonging to $LN_S^{\epsilon,o} \setminus TN_S^{\epsilon,o}$, we have:

$$\begin{aligned}\left(\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|\right) dc_S^{min}(o) &\leq \textstyle\sum_{o' \in\ LN_S^{\epsilon,o} \setminus TN_S^{\epsilon,o}} dc_S(o, o') \\ &\leq (\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|) dc_S^{max}(o),\end{aligned}$$

for objects belonging to $O \setminus LN_S^{\epsilon,o}$, we have:

$$\left(|O| - \left|LN_S^{\epsilon,o}\right|\right) dc_S^{min}(o) \leq \textstyle\sum_{o' \in\ O \setminus LN_S^{\epsilon,o}} dc_S(o, o') \leq (|O| - \left|LN_S^{\epsilon,o}\right|) dc_S^{\epsilon}.$$

Thus:

$$\begin{aligned}\tilde{f}_S(o) \geq{}& \tilde{f}_S^{O'}(o) + \\ & (\left|TN_S^{\epsilon,o}\right| - |O'|) dc_S^{\epsilon} + \left(\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|\right) dc_S^{min}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right) dc_S^{min}(o) \\ ={}& \tilde{f}_S^{O'}(o) + (\left|TN_S^{\epsilon,o}\right| - |O'|) dc_S^{\epsilon} + \left(|O| - \left|TN_S^{\epsilon,o}\right|\right) dc_S^{min}(o)\end{aligned}$$

$$\begin{aligned}\tilde{f}_S(o) \leq{}& \tilde{f}_S^{O'}(o) + \\ & (\left|TN_S^{\epsilon,o}\right| - |O'|) dc_S^{max}(o) + \left(\left|LN_S^{\epsilon,o}\right| - \left|TN_S^{\epsilon,o}\right|\right) dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right) dc_S^{\epsilon} \\ ={}& \tilde{f}_S^{O'}(o) + (\left|LN_S^{\epsilon,o}\right| - |O'|) dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right) dc_S^{\epsilon}.\end{aligned}$$

Finally, if $LN_S^{\epsilon,o} \subset O$; that is, $O \setminus LN_S^{\epsilon,o} \neq \emptyset$, then:

$$\tilde{f}_S(o) < \widetilde{f}_S^{O'}(o) < (\left|LN_S^{\epsilon,o}\right| - |O'|) dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right) dc_S^{\epsilon}.$$

Corollary 3 (Bounds by $\boldsymbol{\epsilon\text{-}loose\ neighbours}$). For any object $o \in O$ and $TN_S^{\epsilon,o} \subset O' \subseteq LN_S^{\epsilon,o}$,

$$\tilde{f}_S^{O'}(o) + (|O| - |O'|) dc_S^{min}(o) \leq \tilde{f}_S(o)$$

$$\tilde{f}_S(o) \leq \tilde{f}_S^{O'}(o) + \left(\left|LN_S^{\epsilon,o}\right| - |O'|\right) dc_S^{max}(o) + (|O| - \left|LN_S^{\epsilon,o}\right|) dc_S^{\epsilon}$$

Proof. Since $TN_S^{\epsilon,o} \subset O' \subseteq LN_S^{\epsilon,o}$, for objects belonging to $O \setminus O'$, we divide them into $LN_S^{\epsilon,o} \setminus \mathrm{0}'$ and $O \setminus LN_S^{\epsilon,o}$. Then:

$$\tilde{f}_S(o) = \tilde{f}_S^{O'}(o) + \textstyle\sum_{o' \in LN_S^{\epsilon,o} \setminus O'} dc_S(o, o') + \sum_{o' \in\ O \setminus LN_S^{\epsilon,o}} dc_S(o, o').$$

By Theorem 1, for objects belonging to $LN_S^{\epsilon,o} \setminus O'$, we have:

$$\left(\left|LN_S^{\epsilon,o}\right| - |O'|\right)dc_S^{min}(o) \leq \textstyle\sum_{o' \in \ LN_S^{\epsilon,o} \setminus \ O'} dc_S(o,o') \leq (\left|LN_S^{\epsilon,o}\right| - |O'|)dc_S^{max}(o),$$

for objects belonging to $O \setminus LN_S^{\epsilon,o}$, we have:

$$\left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{min}(o) \leq \textstyle\sum_{o' \in \ O \setminus LN_S^{\epsilon,o}} dc_S(o,o') \leq (|O| - \left|LN_S^{\epsilon,o}\right|)dc_S^{\epsilon}.$$

Thus:

$$\begin{aligned}\tilde{f}_S(o) \geq \tilde{f}_S^{O'}(o) + &\\ &(\left|LN_S^{\epsilon,o}\right| - |O'|)dc_S^{min}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{min}(o) \\ = \tilde{f}_S^{O'}(o) + (|O| - |O'|)dc_S^{min}(o)\end{aligned}$$

$$\tilde{f}_S(o) \leq \tilde{f}_S^{O'}(o) + (\left|LN_S^{\epsilon,o}\right| - |O'|)dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{\epsilon}.$$

Finally, if $LN_S^{\epsilon,o} \subset O$; that is, $O \setminus LN_S^{\epsilon,o} \neq \emptyset$, then:

$$\tilde{f}_S(o) < \ \tilde{f}_S^{O'}(o) < (\left|LN_S^{\epsilon,o}\right| - |O'|)dc_S^{max}(o) + \left(|O| - \left|LN_S^{\epsilon,o}\right|\right)dc_S^{\epsilon}.$$

Corollary 4 (Bounds by a superset of $\boldsymbol{\epsilon}$-*loose neighbours*). For any object $o \in O$ and $LN_S^{\epsilon,o} \subset O' \subseteq O$,

$\tilde{f}_S^{O'}(o) + (|O| - |O'|)dc_S^{min}(o) \leq \tilde{f}_S(o)$

$\tilde{f}_S(o) \leq \tilde{f}_S^{O'}(o) + (|O| - |O'|)dc_S^{\epsilon}$

Proof. $LN_S^{\epsilon,o} \subset O' \subseteq O$. Then:

$$\tilde{f}_S(o) = \tilde{f}_S^{O'}(o) + \textstyle\sum_{o' \in O \setminus O'} dc_S(o,o').$$

By Theorem 1, for objects belonging to $O \setminus O'$, we have:

$$\left(\left|LN_S^{\epsilon,o}\right| - |O'|\right)dc_S^{min}(o) \leq \textstyle\sum_{o' \in \ O \setminus \ O'} dc_S(o,o') \leq (|O| - |O'|)dc_S^{\epsilon}.$$

Thus:

$$\tilde{f}_S(o) \geq \tilde{f}_S^{O'}(o) + \ (|O| - |O'|)dc_S^{min}(o)$$

$$\tilde{f}_S(o) \leq \tilde{f}_S^{O'}(o) + (|O| - |O'|)dc_S^{\epsilon}$$

Since the density of o is the sum of the density contributions of all objects in O and the density contribution decreases with the distance, OAMiner first computes the quasi-density contributions from the objects in $TN_S^{\epsilon,o}$, then from the objects in $LN_S^{\epsilon,o} \setminus TN_S^{\epsilon,o}$, and finally from the objects in $O \setminus LN_S^{\epsilon,o}$.

By computing the bounds of $\tilde{f}_S(o)$, $\mathrm{OAMiner}$ takes a bounding-pruning-refining method, shown in Algorithm 3, to perform density comparison efficiently in Subspace S. Initially $\mathrm{OAMiner}$ estimates the quasi-density of query object q denoted by $\tilde{f}_S(q)$. Then, for an object o, it first computes the bounds of $\tilde{f}_S(o)$ by the sizes of $TN_S^{\epsilon,o}$ and $LN_S^{\epsilon,o}$(Corollary 1) and compares the bounds with $\tilde{f}_S(q)$ (Steps 1 to 8). If $\tilde{f}_S(q)$ is smaller than the lower bound or greater than the upper bound, then we have $\tilde{f}_S(q) < \tilde{f}_S(o)$ or $\tilde{f}_S(q) > \tilde{f}_S(o)$. That is, the relationship between $\tilde{f}_S(q)$ and $\tilde{f}_S(o)$ is determined; thus, the Algorithm 3 stops. Otherwise, $\mathrm{OAMiner}$ updates the lower and upper bounds of $\tilde{f}_S(o)$ by involving the quasi-density contributions of objects in $TN_S^{\epsilon,o}$ (Steps 10 to 20), in $LN_S^{\epsilon,o}\backslash TN_S^{\epsilon,o}$ (Steps 21 to 31) and in $O\backslash LN_S^{\epsilon,o}$ (Steps 32 to 42) and repeatedly compares the updated bounds with $\tilde{f}_S(q)$ until the relationship between $\tilde{f}_S(q)$ and $\tilde{f}_S(o)$ is fully determined.

Algorithm 3 Density Comparison

```
Input: quasi-density of the query object: $\tilde{f}_S(o)$, object: $o \in O$, subspace: $S$, the $\epsilon$-$tight\ neighbourhood$ of $o$:
  $TN_S^{\epsilon,o}$, and the $\epsilon$-$loose\ neighbourhood$ of $o$: $LN_S^{\epsilon,o}$
Output: a Boolean value indicating $\tilde{f}_S(o) < \tilde{f}_S(q)$ is true or not
 1: $L \leftarrow$ the lower bound of $\tilde{f}_S(o)$ computed by Corollary 1; //bounding
 2: if $L > \tilde{f}_S(q)$ then
 3:   return false; //pruning
 4: end if
 5: $U \leftarrow$ the upper bound of $\tilde{f}_S(o)$ computed by Corollary 1; //bounding
 6: if $U < \tilde{f}_S(q)$ then
 7:   return true; //pruning
 8: end if
 9: $O' \leftarrow \emptyset$; $\tilde{f}_S^{O'}(o) \leftarrow 0$;
10: for each $o' \in TN_S^{\epsilon,o}$ do
11:   $\tilde{f}_S^{O'}(o) \leftarrow \tilde{f}_S^{O'}(o) + dc_S(o,o'); O' \leftarrow O' \cup \{o'\}$; //refining
12:   $L \leftarrow$ the lower bound of $\tilde{f}_S(o)$ computed by Corollary 2; //bounding
13:   if $L > \tilde{f}_S(q)$ then
14:     return false; //pruning
15:   end if
16:   $U \leftarrow$ the upper bound of $\tilde{f}_S(o)$ computed by Corollary 2; //bounding
17:   if $U < \tilde{f}_S(q)$ then
18:     return true; //pruning
19:   end if
20: end for
```

```
for each $o' \in LN_S^{\epsilon,o} \setminus TN_S^{\epsilon,o}$ do
  $\tilde{f}_S^{O'}(o) \leftarrow \tilde{f}_S^{O'}(o) + dc_S(o,o'); O' \leftarrow O' \cup \{o'\}$; //refining
  $L \leftarrow$ the lower bound of $\tilde{f}_S(o)$ computed by Corollary 3; //bounding
  if $L > \tilde{f}_S(q)$ then
    return false; //pruning
  end if
  $U \leftarrow$ the upper bound of $\tilde{f}_S(o)$ computed by Corollary 3; //bounding
  if $U < \tilde{f}_S(q)$ then
    return true; //pruning
  end if
end for
for each $o' \in O \setminus LN_S^{\epsilon,o}$ do
  $\tilde{f}_S^{O'}(o) \leftarrow \tilde{f}_S^{O'}(o) + dc_S(o,o'); O' \leftarrow O' \cup \{o'\}$; //refining
  $L \leftarrow$ the lower bound of $\tilde{f}_S(o)$ computed by Corollary 4; //bounding
  if $L > \tilde{f}_S(q)$ then
    return false; //pruning
  end if
  $U \leftarrow$ the upper bound of $\tilde{f}_S(o)$ computed by Corollary 4; //bounding
  if $U < \tilde{f}_S(q)$ then
    return true; //pruning
  end if
end for
return false;
```

In $\mathrm{OAMiner}$, the neighbourhood distance in dimension D_i, denoted by ϵ_{D_i}, is defined as $\alpha\sigma_{D_i}$ where σ_{D_i} is the standard deviation in dimension D_i and α is a parameter. Our experiments show that α is not sensitive and can be set in the range of 0.8 to1.2. $\mathrm{OAMiner}$ runs efficiently with this range. Theorem 2 guarantees that regardless of the neighbourhood distance, the ranking results remain unchanged.

Theorem 2. Given an object $o \in O$ and a subspace S, for any neighbourhood distances ϵ_1 and ϵ_2, $rank_S^{\epsilon_1}(o) = rank_S^{\epsilon_2}(o)$ where $rank_S^{\epsilon_1}(o)$ $\left(rank_S^{\epsilon_2}(o)\right)$ is the outlyingness rank of o in S computed using ϵ_1 (ϵ_2).

Proof by contradiction. Let O be a set of objects, S be a subspace, ϵ_1 and ϵ_2 be neighbourhood distances and q be the query object. For any object $o \in O$, denote by L_{ϵ_1} the lower bound of $\tilde{f}_S(o)$ estimated by ϵ_1, $U_{\epsilon_2}$the upper bound of $\tilde{f}_S(o)$ estimated by ϵ_2. Assume that $\tilde{f}_S(q) < L_{\epsilon_1}$ and $\tilde{f}_S(q) > U_{\epsilon_2}$. Since $L_{\epsilon_1}$is a lower bound of $\tilde{f}_S(o)$ and U_{ϵ_2} is an upper bound of $\tilde{f}_S(o)$, $L_{\epsilon_1} < \tilde{f}_S(o) < U_{\epsilon_2}$. Then, we have $\tilde{f}_S(q) < L_{\epsilon_1} < \tilde{f}_S(o)$ and $\tilde{f}_S(o) < U_{\epsilon_2} < \tilde{f}_S(q)$. Consequently, $\tilde{f}_S(o) < \tilde{f}_S(q) < \tilde{f}_S(o)$. A contradiction. Thus, $rank_S^{\epsilon_1}(q) = \left|\{o \in O \middle| \tilde{f}_S(o) < \tilde{f}_S(q)\}\right| + 1 = rank_S^{\epsilon_2}(q)$.

Efficiently Estimating Density Bounds

Given a candidate subspace $S \subseteq D$ and an object $o \in O$, to estimate lower and upper bounds of $\tilde{f}_S(o)$, OAMiner has to compute $TN_S^{\epsilon,o}, LN_S^{\epsilon,o}, dc_S^{\epsilon}, dc_S^{min}(o), dc_S^{max}(o)$ and $dc_S(o, o')$ where $o' \in O$. For $|S| = 1$, we compute $TN_S^{\epsilon,o}$, dc_S^{ϵ}, $dc_S^{min}(o), dc_S^{max}(o)$ and $dc_S(o, o')$ according to their definitions. $TN_S^{\epsilon,o} = LN_S^{\epsilon,o}$ in this case. Further, the density contribution is symmetrical such that the computational cost for $dc_S(o', o)$ can be saved if $dc_S(o, o')$ is available. Since OAMiner searches subspaces by traversing the subspace enumeration tree in a depth-first manner, for a subspace satisfying $|S| \geq 2$, we denote by $par(S)$ the parent subspace of S. Suppose $S \backslash par(S) = D'(|D'| = 1)$. Then we have:

$$TN_S^{\epsilon,o} = TN_{par(S)}^{\epsilon,o} \cap TN_{D'}^{\epsilon,o}$$

$$LN_S^{\epsilon,o} = LN_{par(S)}^{\epsilon,o} \cup LN_{D'}^{\epsilon,o}$$

$$dc_S^{\epsilon} = e^{-\sum_{D_i \in S} \frac{\epsilon_{D_i}^2}{2h_{D_i}^2}} = e^{-(\sum_{D_i \in par(S)} \frac{\epsilon_{D_i}^2}{2h_{D_i}^2} + \frac{\epsilon_{D'i}^2}{2h_{D'i}^2})} = dc_{par(S)}^{\epsilon} \cdot dc_{SD'}^{\epsilon}$$

$$dc_S^{min}(o) = e^{-(\sum_{D_i \in par(S)} \frac{\max\limits_{o' \in O}\{|o.D_i - o'.D_i|\}^2}{2h_{D_i}^2} + \frac{\max\limits_{o' \in O}\{|o.D' - o'.D'|\}^2}{2h_{D'}^2})}$$

$$= dc_{S \backslash par(S)}^{min}(o) \cdot dc_{D'}^{min}(o)$$

$$dc_S^{max}(o) = e^{-(\sum_{D_i \in par(S)} \frac{\min\limits_{o' \in O}\{|o.D_i - o'.D_i|\}^2}{2h_{D_i}^2} + \frac{\min\limits_{o' \in O}\{|o.D' - o'.D'|\}^2}{2h_{D'}^2})}$$

$$= dc^{max}_{S\backslash par(S)}(o) \cdot dc^{max}_{D'}(o)$$

$$dc_S(o,o') = e^{-\sum_{D_i \in S} \frac{(o.D_i - o'.D_i)^2}{2h^2_{D_i}}} = e^{-(\sum_{D_i \in par(S)} \frac{(o.D_i - o'.D_i)^2}{2h^2_{D_i}} + \frac{(o.D_i - o'.D_i)^2}{2h^2_{D_i}})}$$

$$= dc_{par(S)}(o,o') \cdot dc_{D'}(o,o')$$

Accordingly, $\mathrm{OAMiner}$ can efficiently estimate the bounds of $\tilde{f}_S(o)$ using $par(S)$ and $S\backslash par(S)$.

Subspace Pruning

While $\mathrm{OAMiner}$ is traversing the subspace enumeration tree, let S_1 be the set of subspaces it has searched so far and S_2 be the set of subspaces it has not searched yet. $|S_1 \cup S_2| = 2^{|D|} - 1$. Given a query object q, let $r_{best} = \min_{S \in S_1}\{rank_S(q)\}$ be the best rank q has achieved so far. We can then use r_{best} to prune some subspaces not searched yet; that is, for a subspace $S \in S_2$, once we determine that $rank_S(q) > r_{best}$, then S cannot be an outlying aspect and thus can be pruned.

Observation 1. When subspace S is met in a depth-first search of the subspace enumeration tree, let r_{best} be the best rank of q in all subspaces searched so far. Given object q with $rank_S(q) > 1$, if for every proper superspace $S' \supset S$, $rank_{S'}(q) > r_{best}$ then all proper superspace of S can be pruned.

For $rank_S(q) = 1$, all superspaces of S can be pruned due to dimensionality minimality condition as per Pruning Rule 1. As a result, we only consider the case where $rank_S(q) > 1$. To implement Observation 1 in a subspace S where $rank_S(q) > 1$, we check if there are at least r_{best} objects that are ranked better than q in every superspace of S. If true, all superspaces of S can be pruned. The common factor c as per Equation (5.2.6) does not affect the outlyingness rank. For simplicity, $\mathrm{OAMiner}$ computes quasi-density $\tilde{f}_S(o)$ (Equation (5.2.7)) instead of probability density $\hat{f}_S(o)$ (Equation (5.2.5)) for ranking. Thus, we have the following monotonicity of $\tilde{f}_S(o)$ with respect to subspaces.

Lemma 1. For a set of objects O and two subspaces S and S' satisfying $S' \supset S$, let $D_i \in S' \setminus S$. If the standard deviation of O in D_i is greater than 0, then for any object $o \in O, \tilde{f}_S(o) > \tilde{f}_{S'}(o)$.

Proof. Given $D_i \in S' \setminus S$, for any object $o' \in O$, we have $\frac{(o.D_i - o'.D_i)^2}{2h_{D_i}^2} \geq 0$. Since the standard deviation of O in D_i is greater than 0, there exists at least one object $o'' \in O$ such that $\frac{(o.D_i - o''.D_i)^2}{2h_{D_i}^2} > 0$; that is, $e^{-\frac{(o.D_i - o''.D_i)^2}{2h_{D_i}^2}} < 1$.

Thus:

$$\tilde{f}_S(o) = \sum_{o' \in O} e^{-\sum_{D_i \in S} \frac{(o.D_i - o'.D_i)^2}{2h_{D_i}^2}}$$

$$> \sum_{o' \in O} e^{-\left(\sum_{D_i \in S} \frac{\left(o.D_i - o'.D_i\right)^2}{2h_{D_i}^2} + \sum_{D_i \in S' \setminus S} \frac{\left(o.D_i - o'.D_i\right)^2}{2h_{D_i}^2}\right)}$$

$$= \tilde{f}_{S'}(o)$$

As preprocessing, per Step 2 of Algorithm 2, $\mathrm{OAMiner}$ removes dimensions with standard deviation 0. Consequently, the standard deviation of any dimension $D_i \in S' \setminus S$ is greater than 0. $\mathrm{OAMiner}$ sorts all dimensions in in the ascending order of $rank_{D_i}(q)(D_i \in D)$ and traverse the subspace set enumeration tree in the depth-first manner. Given R the ascending order of $rank_{D_i}(q)$, for a subspace $S = \{D_{i_1}, \dots, D_{i_m}\}$, let $R(S) = \{D_j | D_j \text{ is behind } D_{i_m} \text{ in } R\}$. By Lemma 1, for any subspace S' such that $S \subset S' \subseteq S \cup R(S)$, the minimum quasi-density of q, denoted by $\tilde{f}_{sup(S)}^{min}(q)$, is $\tilde{f}_{S \cup R(S)}(q)$. An object $o \in O$ is called a **competitor** of q in S if $\tilde{f}_S(o) < \tilde{f}_{sup(S)}^{min}(q)$ and the set of q's competitors is denoted by $Comp_S(q)$. For any $o \in Comp_S(q)$ by Lemma 1, we have $\tilde{f}_{S'}(o) < \tilde{f}_S(o) < \tilde{f}_{\sup(S)}^{min}(q) \leq \tilde{f}_{S'}(q)$. Thus, $rank_{S'}(o) < rank_{S'}(q)$. Further we have the following property of $Comp_S(q)$.

Property 3. Given a query object q and a subspace S, for any subspace S' such that $S \subset S'$, $Comp_S(q) \subseteq Comp_{S'}(q)$.

Proof. Since $S \subset S'$, by Lemma 1, for any $o \in Comp_S(q)$, $\tilde{f}_{S'}(o) < \tilde{f}_S(o) < \tilde{f}^{min}_{\sup(S)}(q)$. Since $\tilde{f}^{min}_{\sup(S)}(q) \leq \tilde{f}^{min}_{\sup(S')}(q)$, we have $\tilde{f}_{S'}(o) < \tilde{f}_S(o) < \tilde{f}^{min}_{\sup(S)}(q) \leq \tilde{f}^{min}_{\sup(S')}(q)$. Thus, $o \in Comp_{S'}(q)$; that is, $Comp_S(q) \subseteq Comp_{S'}(q)$.

Accordingly, $\mathrm{OAMiner}$ performs subspace pruning based on the number of competitors.

Pruning Rule 2. When subspace S is met in a depth-first search of the subspace enumeration tree, let r_{best} be the best rank of q in all subspaces searched so far. If there are at least r_{best} competitors of q in S; that is, $|Comp_S(q)| \geq r_{best}$, then all proper superspaces of S can be pruned.

When maximum dimensionality threshold of an outlying aspect ℓ is less than $|S| + |R(S)|$, $|S| < |S'| \leq \ell < |S| + |R(S)|$. It is unsuitable to use $\tilde{f}_{S \cup R(S)}(q)$ as $\tilde{f}^{min}_{\sup(S)}(q)$. Intuitively, we can set $\tilde{f}^{min}_{\sup(S)}(q)$ to $\min\{\tilde{f}_{S'}(q) \big| \, |S'| = \ell, S \subset S' \subset S \cup R(S)\}$. However, the computational cost may be high since the number of candidates is $\binom{|R(S)|}{\ell - |S|}$. Alternatively, we consider a method that uses a lower bound of $\tilde{f}_{S'}(q)$ to compute $\tilde{f}^{min}_{\sup(S)}(q)$ efficiently.

For object o', the quasi-density contribution of o' to q in S, denoted by $\tilde{f}_S(q, o')$, is $e^{-\sum_{D_i \in S} \frac{(o.D_i - o'.D_i)^2}{2h^2_{D_i}}}$. Let $R(S, o')$ be the set of $(\ell - |S|)$ dimensions in $R(S)$ with the largest values of $\frac{|q.D_j - o'.D_j|}{h_{D_j}}$ $(D_j \in R(S))$. Then, the minimum quasi-density contribution of o' to q in S' $(S \subset S')$ is $\tilde{f}_{S \cup R(S,o')}(q, o')$. Since $\tilde{f}_{S'}(q) = \sum_{o' \in O} \tilde{f}_{S'}(q, o')$, we have $\tilde{f}^{min}_{\sup(S)}(q)$. If we compared $\tilde{f}^{min}_{\sup(S)}(q)$ with the quasi-density values of all objects in O, the computational cost for density estimation would be considerably high especially when the size of O is large. For efficiency, a trade-off needs to be made between subspace pruning and object pruning. That is, when a subspace S is searched, once $rank_S(q) > r_{best}$ is determined, the search of S can be terminated immediately.

Algorithm 4 gives the pseudo code of computing outlyingness rank and pruning subspaces in $\mathrm{OAMiner}$.

Algorithm 4 $rank_S(q)$

```
Input: query object: $q \in O$, subspace: $S$, the set of competitors of $q$ discovered in the parent subspaces of $S$:
  $Comp$ ($Comp$ is empty if $|S| = 1$), the best rank of $q$ in the subspaces searched so far: $r_{best}$
Output: $rank_S(q)$
compute $\tilde{f}_S(q)$ using Equation (5.2.7);
$rank_S(q) \leftarrow |Comp| + 1$;
for each object $o \in O \backslash Comp$ do
  if $\tilde{f}_S(o) < \tilde{f}_S(q)$ then
    $rank_S(q) \leftarrow rank_S(q) + 1$;
    if $\tilde{f}_S(o) < \tilde{f}^{min}_{\sup(S)}(q)$ then
      $Comp \leftarrow Comp \cup \{o\}$;
    if $|Comp| = r_{best}$ then
      prune descendants of $S$ and return; //Pruning Rule 2
     end if
   end if
  if $rank_S(q) > r_{best}$ then
    return;
  end if
 end if
end for
return $rank_S(q)$;
```

Theorem 3 guarantees that Algorithm 4 can find all minimal outlying subspaces.

Theorem 3 (Completeness of OAMiner). Given a set of objects O in a multi-dimensional space D, a query object $q \in O$ and a maximum dimensional threshold $0 < \ell \leq |D|$, $\mathrm{OAMiner}$ finds all minimal outlying subspaces of q.

Proof by contradiction. Let Ans be the set of minimal outlying subspaces of q found by $\mathrm{OAMiner}$ and r_{best} be the best rank. Assume that subspaces $S \notin Ans$ satisfying $S \subseteq D$ and $0 < |S| \leq \ell$ is a minimal outlying subspace of q. Since $\mathrm{OAMiner}$ searches subspaces by traversing the subspace enumeration tree in a depth-first manner and $S \notin Ans$, S is pruned by Pruning Rule 1 or Pruning Rule 2. If S is pruned by Pruning Rule 1, S is not minimal. A contradiction. If S is pruned by Pruning Rule 2, then there exists a subspace S' such that is S' a parent of S in the subspaces enumeration tree and $Comp_{S'}(q) \geq r_{best}$. Per the property of competitors, we have $Comp_{S'}(q) \subseteq$

$Comp_S(q)$. Accordingly, $rank_S(q) \geq |Comp_S(q)| \geq |Comp_{S'}(q)| \geq r_{best}$. A contradiction.

5.2.3. Empirical Evaluation

In this section, we present a systematic empirical study using several real data sets as well as synthetic data sets to verify the effectiveness and efficiency of $\mathrm{OAMiner}$. All experiments were conducted on a PC with Intel Core i7-3770 3.40 GHz CPU and 8GB RAM, running Windows 7 operating system. All algorithms were implemented in Java and compiled with JDK 7. Since it is hard to comprehend the meaning of subspaces with dimensionality over 5, defaults were set to $\ell = 5, \alpha = 1.0$.

Effectiveness

Mining Outlying Aspects with Synthetic Data Sets

Keller et al. (2012) provided a collection of synthetic data sets, each set consisting of 1,000 data objects. Each data set contains some subspace outliers which deviate from all clusters in at least one 2-5 dimensional subspaces. As observed by Keller et al. (2012), an object can be an outlier in multiple subspaces independently. We performed our tests on the data sets of 10, 20, 30, 40, and 50 dimensions denoted by $Synth\_10D$, $Synth\_20D$, $Synth\_30D$, $Synth\_40D$, and $Synth\_50D$ respectively.

For an outlier q in a data set, let S be the ground truth outlying subspace of q. Note that S may not be an outlying aspect of q if there exists another outlier more outlying than q in S since $\mathrm{OAMiner}$ finds the subspaces in which the query object is most outlying. To verify the effectiveness of $\mathrm{OAMiner}$, using the known ground truth outlying subspaces with multiple implanted outliers in S, we select one outlier as a query object q at a time and remove the other outliers, and repeat this process for each implanted outlier. Since q is the only implanted outlier in subspace S, $\mathrm{OAMiner}$ is expected find the ground truth outlying subspace S where q takes rank 1 in outlyingness; that is, $rank_S(q) = 1$. We divide the mining results of $\mathrm{OAMiner}$ into the following 3 cases:

Case 1: only the ground truth outlying subspace is discovered by $\mathrm{OAMiner}$ with outlyingness rank 1.

Case 2: in addition to the ground truth outlying subspace, $\mathrm{OAMiner}$ finds other outlying aspects with outlyingness rank 1.

Case 3: instead of the ground truth outlying subspace, $\mathrm{OAMiner}$ finds a subset of the ground truth as an outlying aspect with outlyingness rank 1.

Mining results for $Synth\_10D$ are shown in Table 25. Results use the same object ID's and dimension ID's as in the original data set in the work of Keller et al. (2012).

Table 25 Outlying Aspects with $\boldsymbol{Synth\_10D}$

Query Object	Ground Truth Outlying Subspace	Outlying Aspect with Outlyingness Rank 1	Case
172	{8,9}	{8,9}	1
183	{0,1}	{0,1}, {0,6,8}	2
184	{6,7}	{6,7}	1
207	{0,1}	{0,1}	1
220	{2,3,4,5}	{2,3,4,5}	1
245	{2,3,4,5}	{2,5}	3
315	{0,1},{6,7}	{0,1}, {6,7}, {3,4}, {3,5,9}, {4,6,9}	2
323	{8,9}	{8,9}	1
477	{0,1}	{0,1}	1
510	{0,1}	{0,1}	1
577	{2,3,4,5}	{2,3,4,5}, {0,3,7}	2
654	{2,3,4,5}	{2,3,4,5}	1
704	{8,9}	{8,9}, {0,2,3,4}	2
723	{2,3,4,5}	{2,3,4,5}	1
754	{6,7}	{6,7}, {2,4,8}, {2,6,8}, {4,6,8}	2
765	{6,7}	{6,7}, {1,4,6}, {3,4,5,6}	2
781	{6,7}	{6,7}	1
824	{8,9}	{8,9}	1
975	{8,9}	{8,9}, {2,5,9}, {5,6,8}, {2,3,5,8}	2

For all outliers used as query objects, outlying aspects with outlyingness rank 1 were identified. Also, for objects 183, 315, 577, 704, 754, 765 and 975, $\mathrm{OAMiner}$ found not only the ground truth outlying subspaces but also other outlying subspaces (Case 2). For object 245, the outlying aspect discovered by $\mathrm{OAMiner}$ is a subset of the ground truth outlying subspace (Case 3). For the other 11 objects, the outlying aspects identified by $\mathrm{OAMiner}$ are identical to the ground truth subspaces (Case 1).

Table 26 summarizes the mining results for all of $Synth\_10D$, $Synth\_20D$, $Synth\_30D$, $Synth\_40D$, and $Synth\_50D$.

Table 26 Statistics on the mining results of $\mathrm{OAMiner}$

Data Set	# of Outliers	# of Case 1	# of Case 2	# of Case 3
$Synth\_10D$	19	11	7	1
$Synth\_20D$	25	1	23	1
$Synth\_30D$	44	0	40	4
$Synth\_40D$	53	0	52	1
$Synth\_50D$	68	0	65	3

The number of Case 2 instances increases with higher dimensionality which indicates that more outlying aspects can be found when more attributes of data are in scope for search, which is consistent with the experimental observations with the real data sets shown later.

To further illustrate the effectiveness of $\mathrm{OAMiner}$, Figure 36 shows how visually object 245 (Case 2) stands out, likewise, Figure 37 object 315 (Case 3).

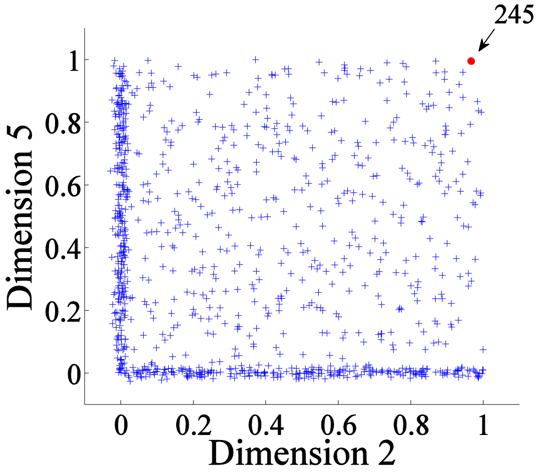


Figure 36 Outlying aspect of object 245 ($\boldsymbol{Synth\_10D}$)

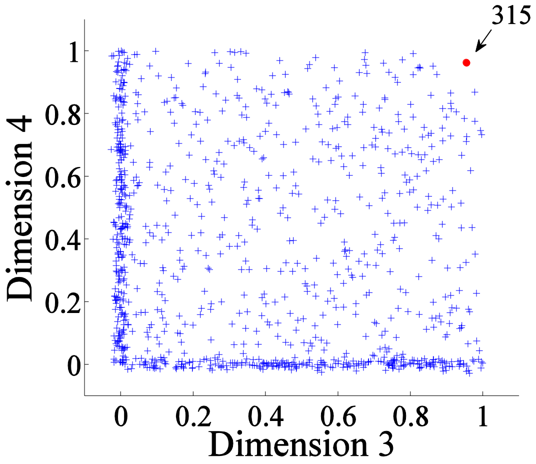


Figure 37 Outlying aspect of object 315 ($Synth\_10D$)

Mining Outlying Aspects with Real Data Sets

We use the same UCI data set (Bache et al., 2013) in section 5.1.4 for measuring the effectiveness of $\mathrm{OAMiner}$. Again non-numerical attributes and all records that are missing values were removed from the data sets. Data set statistics is shown in Table 27.

Table 27 UCI data set characteristics

Data Set	# of Objects	# of Attributes
Breast Cancer	194	33
Climate Model	540	18
Concrete Slump	103	10
Parkinson’s	195	22
Wine	178	13

Figure 38 shows the distributions of the best outlyingness ranks for the UCI data set. The best rank values are small for most objects, meaning most objects are ranked well in some subspaces.

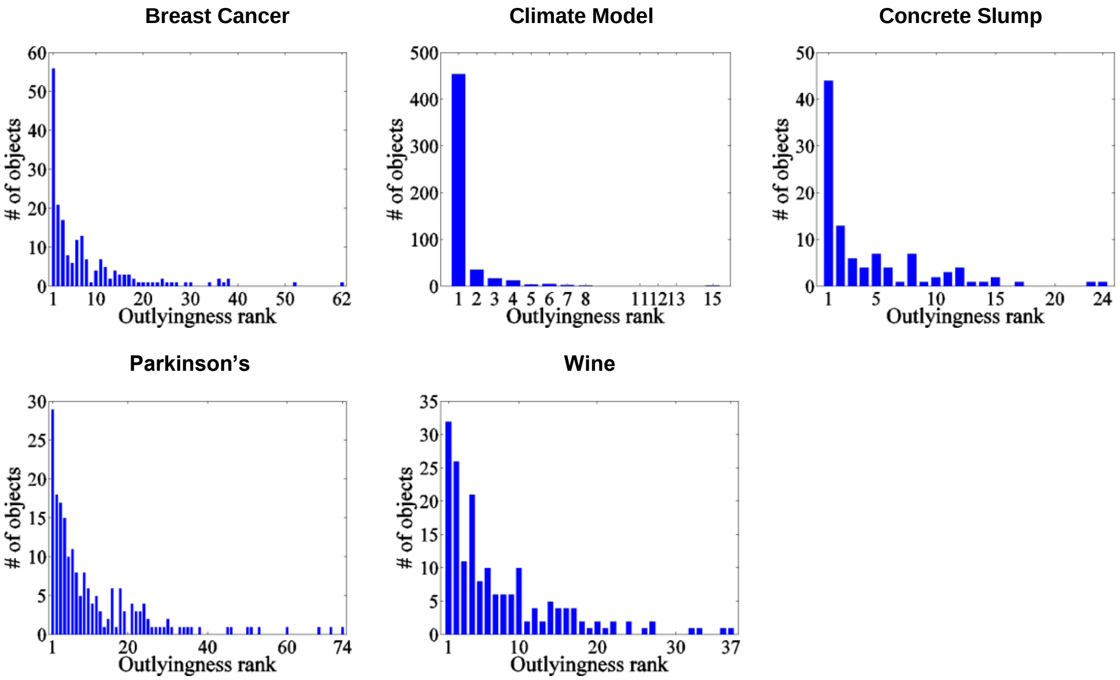


Figure 38 Distribution of outlyingness ranks: UCI ($\ell = 5$)

Figure 39 shows the distributions of the number of minimal outlying subspaces where the objects achieve the best outlyingness ranks (i.e. outlying aspects). For most objects, the number of outlying aspects is small, indicating that most objects can be distinguished from others by a small number of factors.

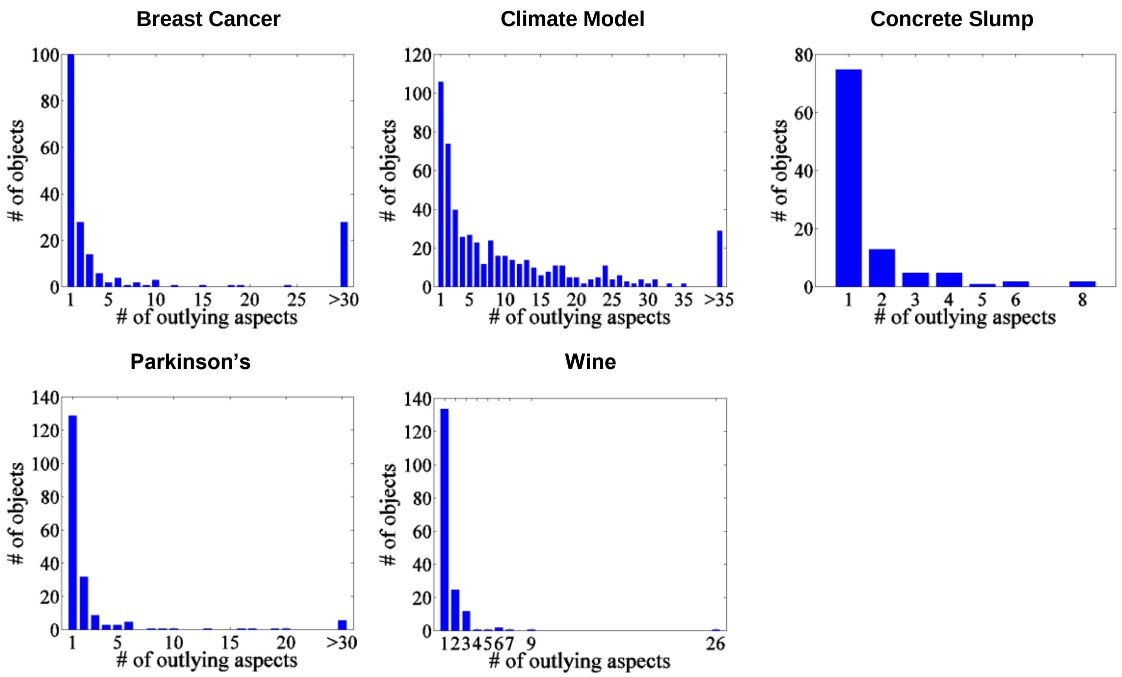


Figure 39 Distribution of # of outlying aspects: UCI ($\ell = 5$)

Table 28 exhibits the mining results of $\mathrm{OAMiner}$ when $\ell = 4, 5, 6$. As the value of ℓ increases, the average outlyingness rank decreases while the average number of outlying aspects as well as the average dimensionality increases. It can be observed that more outlying aspects can be found as the number of attributes and the objects increases; for example, the average number of outlying aspects found for the breast cancer data is the largest.

Table 28 Sensitivity of $\mathrm{OAMiner}$ effectiveness with respect to ℓ (UCI)

Data set	ℓ	Outlyingness Rank			# of Outlying Aspects			Dimensionality		
		Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
Breast Cancer	4	1	70	8.04	1	232	9.57	1	4	3.47
	5	1	62	7.74	1	2,478	43.37	1	5	4.67
	6	1	56	7.57	1	11,681	243.10	1	6	5.77
Climate Model	4	1	33	1.97	1	30	4.57	1	4	3.65
	5	1	15	1.45	1	78	10.18	1	5	4.43
	6	1	15	1.28	1	149	16.97	1	6	5.07
Concrete Slump	4	1	27	4.67	1	8	1.56	1	4	2.38
	5	1	24	4.44	1	8	1.64	1	5	2.59
	6	1	24	4.41	1	8	1.65	1	6	2.66
Parkinson's	4	1	74	12.13	1	156	4.20	1	4	3.25
	5	1	74	11.51	1	400	7.63	1	5	4.09
	6	1	74	11.33	1	889	14.30	1	6	5.01
Wine	4	1	37	7.65	1	26	1.49	1	4	2.66
	5	1	37	7.47	1	26	1.59	1	5	2.96
	6	1	37	7.46	1	26	1.66	1	6	3.09

Mining Outlying Aspects with NBA Data Sets

To evaluate the usefulness of outlying aspect mining, we analyze outlying aspects of some NBA players in detail. We first investigate the outlying aspects of all NBA guards, forwards and centres in the 2012-2013 Season. The technical statistics for 20 numerical attributes were collected from http://sports.yahoo.com/nba/stats/. Table 29 lists the names of dimensions and Table 30 the data set characteristics. The data for centres for 3-Points are removed since the statistics for most centres are 0.

Table 29 NBA 20 data dimensions

1: Game played	6: 3-Points (M)	11: Free throw (Pct)	16: Turnover
2: Minutes	7: 3-Points (A)	12: Rebounds (Off)	17: Steal
3: Field goal (M)	8: 3-Points (Pct)	13: Rebounds (Def)	18: Block
4: Field goal (A)	9: Free throw (M)	14: Rebounds (Tot)	19: Personal foul
5: Field goal (Pct)	10: Free throw (A)	15: Assist	20: Points/game

Table 30 NBA data set characteristics

Data set	# of Objects	# of Attributes
Guards	220	20
Forwards	160	20
Centres	46	17

Figure 40 shows the distributions of the best outlyingness ranks for the NBA data set. Consistent with the UCI data set findings, the best rank values are small for most objects; for example, 90 guards (40.9%), 81 forwards (50.6%) and 32 centres (69.6%) have an outlying rank of 5 or better (i.e. smaller rank values). This means most players have some subspaces where they are substantially different from the others; justifying the need for outlying aspect mining.

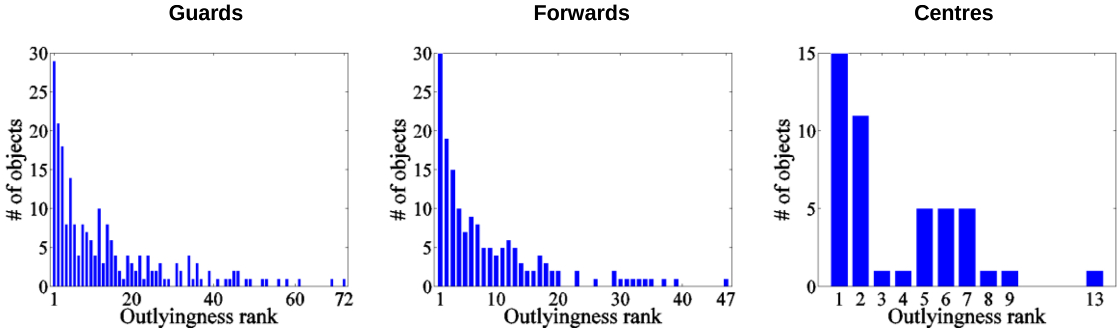


Figure 40 Distributions of outlyingness ranks: NBA ($\ell = 5$)

Figure 41 shows the distributions of the number of outlying aspects. Again, consistent with the UCI data set results, for most objects, the number of outlying aspects is small. For example, 150 (68.2%) guards have only 1 outlying aspect.

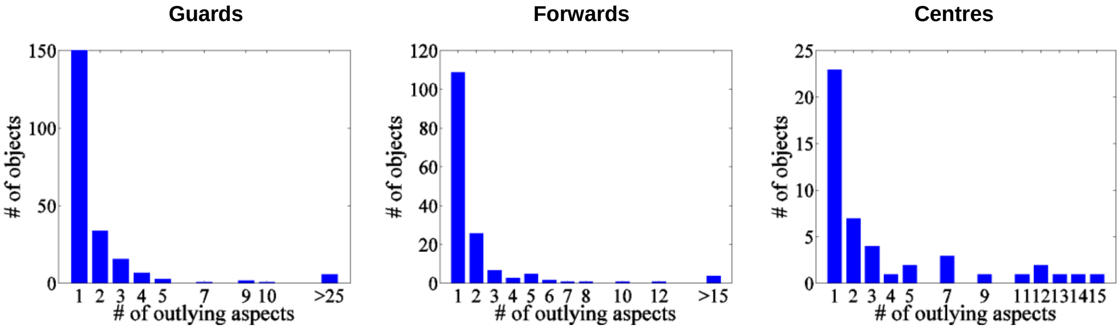


Figure 41 Distribution of # of outlying aspects: NBA ($\ell = 5$)

Table 31 exhibits the mining results of $\mathrm{OAMiner}$ when $\ell = 4, 5, 6$. Similar to the observations made with the UCI data set, the average number of outlying aspects and the average dimensionality increases along with the value of ℓ, whereas the average outlyingness rank decreases. This observation is coherent with the well-known fact that probability density tends to be low in subspaces of higher dimensionality since such subspaces often have a larger volume and thus sparser.

Table 31 Sensitivity of OAMiner effectiveness with respect to ℓ (NBA)

Data set	ℓ	Outlyingness Rank			# of Outlying Aspects			Dimensionality		
		Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
Guards	4	1	72	13.94	1	49	2.02	1	4	2.79
	5	1	72	13.70	1	111	3.05	1	5	3.68
	6	1	72	13.50	1	359	5.67	1	6	4.83
Forwards	4	1	48	8.79	1	40	2.24	1	4	2.77
	5	1	47	8.54	1	41	2.37	1	5	3.13
	6	1	46	8.43	1	71	2.98	1	6	3.77
Centres	4	1	13	3.70	1	15	3.28	1	4	2.74
	5	1	13	3.57	1	15	3.65	1	5	3.08
	6	1	13	3.54	1	18	3.61	1	6	3.23

A player receives a good outlyingness rank (i.e. a small rank value) in a subspace if very few other players are close to him. Table 32 lists 10 guards who have the largest number of rank 1 outlying aspects where $\ell = 3$. Dimensions in Table 32 corresponds to the serial numbers in Table 29. Two types of reasoning can be made as to why certain objects stand out in some subspaces. One is there are not enough comparable statistics for the objects in any subspace; another is there are enough statistics and these objects are truly unique by the characteristics these subspaces represent.

Table 32 Guards with most rank 1 outlying aspects

Name	Outlying Aspect ($\ell = 3$)
Quentin Richardson	{1}, {12}, {14}, {2,17}, {3,4}, {3,13}, {4,17}, {5,8}, {5,11}, {5,13}, {13,17}, {13,20}, {2,3,16}, {2,4,5}, {2,5,6}, {2,5,7}, {2,5,9}, {4,5,7}
Will Conroy	{2,5}, {5,8}, {5,11}, {5,12}, {5,13}, {5,14}, {5,16}, {4,5,6}, {4,5,9}, {4,5,10}, {4,5,7}, {4,5,19}, {5,6,7}, {5,7,9}
Brandon Rush	{5}, {1,19}, {2,19}, {17,19}
Ricky Rubio	{3,17}, {7,17}, {16,17}, {17,20}
Rajon Rondo	{15}, {16}, {1,17}, {1,2,20}
Scott Machado	{19}, {2,16}, {5,8,18}
Kobe Bryant	{3}, {4}, {20}
Jamal Crawford	{19,20}, {4,19}, {2,3,19}
James Harden	{9}, {10}
Stephen Curry	{6}, {7}

The first several players in Table 32 are not well known and their outlyingness ranks are due to the fact that no other players have similar statistics. For example:

- Quentin Richardson played only one game during which he did well at rebounds but poorly at field goal;
- Will Conroy played four games and his performance for shooting was poor;
- Brandon Rush played two games and his number of personal fouls is large;
- Ricky Rubio performed well at stealing;
- Rajon Rondo assisted well but his statistics for turnover is large;
- Scott Machado played 6 games and did not make any personal fouls.

The remaining 4 players are famous and their overall performance in every aspect is much better than most other guards. For example:

- Kobe Bryant excels at scoring;
- Jamal Crawford has very low personal fouls;
- James Harden excels at free throw;
- Stephen Curry leads in 3-points scoring.

Table 33 lists guards who were not ranked well in any subspace; in other words, they do not stand out in any particular subspace.

Table 33 Guards with poor ranks in outlying aspects

Outlyingness Rank	Nam	Outlying Aspect
72	Terence Rose	{11}
70	E’Twaun Moore	{18}
69	C.J. Watson	{8,12,13,14,18}
61	Jerryd Bayless	{2,3,4,19,20}
58	Nando De Colo	{1,2}, {3,4,5,11,20}
56	Alec Burks	{2,9,10,11}
55	Rodrigue Beaubois	{1,2,8,11,15}
52	Marco Belinelli	{9,10,12}
49	Aaron Brooks	{2,3,5,7,16}
48	Nick Young	{1,3,16,18,20}

Although subspace outlier detection is fundamentally different from outlying aspect mining, the results of subspace outlier ranking can be utilized to verify the discovered outlying aspects. Specifically, we take the objects that are ranked the best by either HiCS (Keller et al., 2012) or SOD (Kriegel et al., 200) and determine their outlyingness ranks for comparison.

Since HiCS randomly selects subspace slices, we ran it 3 times independently on each data set with the default parameters. The parameter for the number of nearest neighbours in both HiCS and SOD was varied across 5, 10 and 20, and the best ranks were reported. In SOD, the parameter l which specifies the size of the reference sets cannot be larger than the number of nearest neighbours, as such, we set it to the number of nearest neighbours for our experimentation.

Table 34 shows the results. $rank_{HL}$ means the ranks computed by HiCS, $rank_{SOD}$ the ranks computed by SOD and $rank_S$ the outlyingness rank computed by OAMiner.

Table 34 Comparison of $rank_{HL}, rank_{SOD}, rank_S$

Position	Name	$rank_{HL}$	$rank_{SOD}$	$rank_S$ (# of Outlying Aspects)
Guard	Quentin	1	1	1(54)
	Richardson	1	9	1(3)
	Kobe Bryant	32	1	1(4)
	Brandon Ray			
Forward	Carmelo Anthony	1	5	1(26)
	Kevin Love	3	1	1(41)
Centre	Dwight Howard	1	2	1(15)
	Andrew Bogut	10	1	1(9)

The results show that every player ranked top with either HiCS or SOD has some outlying aspects where he is ranked number 1. The rankings produced by OAMiner match those either by HiCS or SOD, although the results by HiCS and SOD are not consistent with each other except for Quentin Richardson.

Efficiency

Once again, to the best of our knowledge, there is no previous method addressing the efficiency of the same mining problem. As such, we will evaluate the efficiency of OAMiner and its variations; that is, comparisons amongst baseline (Algorithm 1 with Pruning Rule 1), OAMiner-$part$ (the version that does not use bounds), and OAMiner-$full$ (the version that uses all techniques).

The same synthetic data set provided by Keller et al. (2012) was used. The set consists of 1,000 data objects and the dimensionality is 50. 10 data points (non-outliers) were randomly chosen as query objects and the average runtime was reported. For all 3 variations, $\ell = 5$ and for $\mathrm{OAMiner}\text{-}full$, $\alpha = 1.0$.

Figure 42 shows the runtime (on logarithm scale) with respect to data set size. As expected, the baseline method is time consuming. The pruning techniques can achieve a roughly linear runtime; both $part$ and $full$ versions of $\mathrm{OAMiner}$ are substantially faster than the baseline and $full$ is more efficient than $part$.

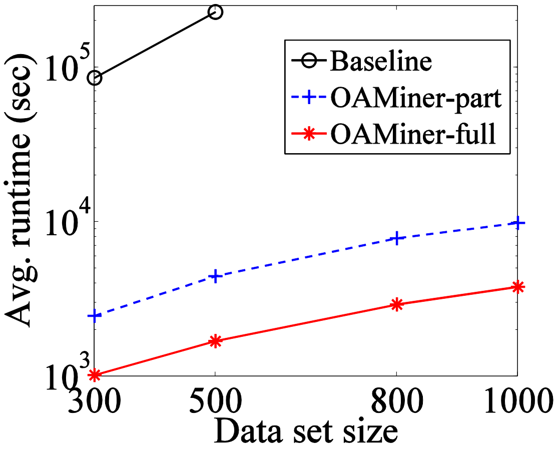


Figure 42 Runtime with respect to Data Set Size

Figure 43 shows the runtime (on logarithm scale) with respect to dimensionality. As expected, as the dimensionality increases, the runtime increases exponentially. The pruning techniques can achieve a roughly linear runtime; both $part$ and $full$ versions of $\mathrm{OAMiner}$ are substantially faster than the baseline and $full$ is more efficient than $part$.

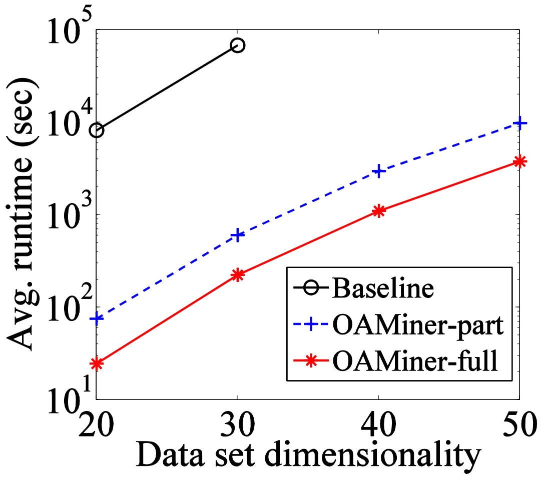


Figure 43 Runtime with respect to Dimensionality

Figure 44 shows the runtime (on logarithm scale) with respect to maximum dimensionality threshold (ℓ). As the value of ℓ increases, the more subspaces are enumerated and thus the runtime increases. The pruning techniques can achieve a roughly linear runtime in practice; both $part$ and $full$ versions of OAMiner are substantially faster than the baseline and $full$ is more efficient than $part$.

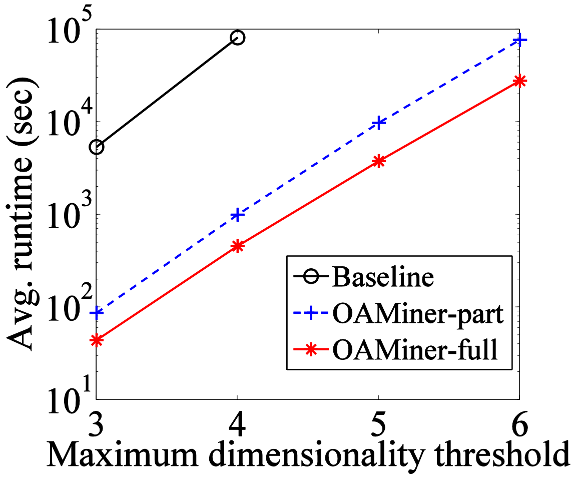
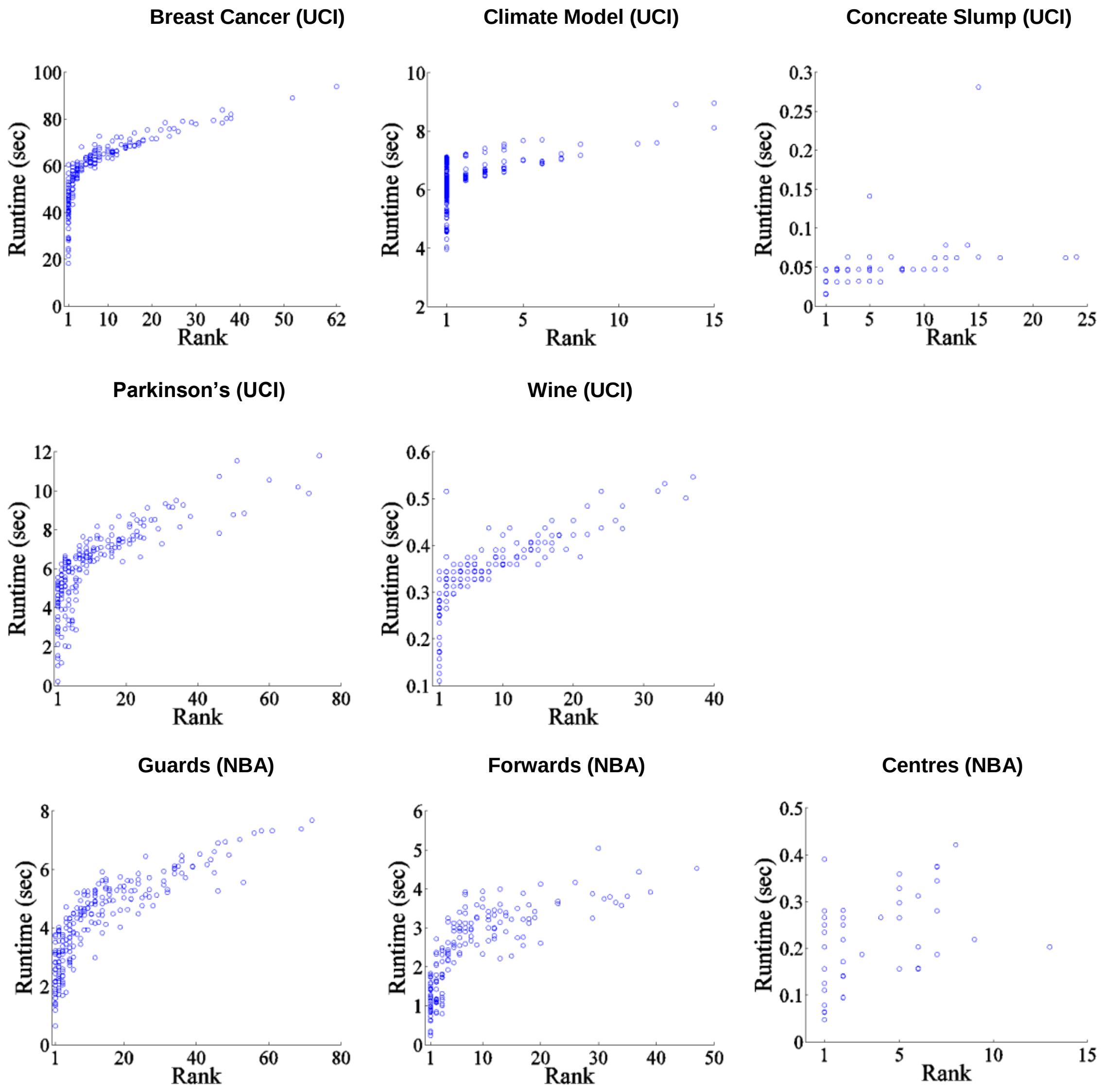


Figure 44 Runtime with respect to ℓ

Using the real data sets (both UCI and NBA), the efficiency of OAMiner has been tested against the outlyingness of the query object. Figure 45 shows the runtime with respect to outlyingness rank of the query object. The runtime is proportional to the outlyingness rank of the query object.



Breast Cancer (UCI)

Climate Model (UCI)

Concreate Slump (UCI)

Parkinson’s (UCI)

Wine (UCI)

Guards (NBA)

Forwards (NBA)

Centres (NBA)

Figure 45 Runtime with respect to Outlyingness Rank

Not surprisingly, the objects with large outlyingness rank cost more runtime since OAMiner prunes subspaces based on the rank of the query object by means of either Pruning Rule 1 or Pruning Rule 2.

Finally, the sensitivity of the parameter α for bounding quasi-density has been tested with varying values of α. The value of α sets the ϵ-$neighbourhood$ distance. Table 35 lists the average runtime of $\mathrm{OAMiner}$ for each data set.

Table 35 Average Runtime of OAMiner with respect to $\boldsymbol{\alpha}$

Data Set	**Average Runtime (second)**				
	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$	$\alpha = 1.2$	$\alpha = 1.4$
Guards	4.459	4.234	**4.213**	4.303	4.315
Forwards	2.810	2.519	2.424	2.418	**2.413**
Centres	0.260	0.234	0.216	**0.212**	0.220
Breast Cancer	58.476	58.228	57.927	**57.613**	57.982
Climate Model	6.334	6.268	6.339	**6.253**	6.410
Concrete Slump	0.047	**0.044**	0.044	0.045	0.045
Parkinson’s	6.164	6.154	**6.083**	6.218	6.243
Wine	0.351	0.341	**0.339**	0.344	0.350

The runtime of $\mathrm{OAMiner}$ is not sensitive to α in general. Experimentally, the shortest runtime of $\mathrm{OAMiner}$ is achieved when α is in $[0.8,1.2]$.

Chapter 6.

Conclusion

In this thesis, we explored a multitude of techniques applicable to multidimensional benchmarking.

Benchmarking is an important business practice which sets organizations' performance improvement targets by comparing them to others and identifying areas where the performance gaps exist. While in practice many organizations limit their benchmarking scope to the numerical quantification of performance gaps (e.g. company A's service level is 15% below the benchmark), it is well recognized that quantitative benchmarking alone does not help organizations actually achieve performance improvement. In order to improve performance, organizations need to understand the key drivers for the gaps. Why then do organizations not take more qualitative approach? It is because, in the author's opinion, qualitative measures are difficult to model. In this thesis, we claimed that multidimensional analysis approach can be used as a step towards more qualitative benchmarking.

Existing multidimensional benchmarking methods build upon economic efficiency analysis, such as, frontier models which estimate the ability of a producer to produce maximum output from a given set of inputs. Chapter 2 presented two representative methods of the model; stochastic frontier production function and data envelopment analysis. The chapter also briefly touched on non-systematic proprietary approach including Gartner Magic Quadrant and Forrester Wave.

Despite that the main concern of multidimensional benchmarking is to consider multiple dimensions simultaneously, to the best of the author's knowledge, there are no notable techniques in the industry which take advantage of data warehouses (i.e. multidimensional databases) and associated computational algorithms. The key ideas presented in this thesis aim to expand the scope of multidimensional benchmarking by leveraging data warehousing and business intelligence, outlier detection in data warehouses, and subspace analysis:

- Identifying significant benchmarks efficiently in data warehouses (Chapter 3).

  Rather than comparing organizations to “any” population, it is more meaningful to compare them to the population that renders the largest performance gap (i.e. a significant benchmark). Finding significant benchmarks only without looking at everything in the data warehouse that constitutes a population requires an efficient computational approach. We developed 2 efficient methods: SIIC/SIICP and DAM. DAM outperforms SIIC/SIICP because it only stores and searches the dominant answers in the quotient groups.

- Detecting outliers in data warehouses as a basis for multidimensional benchmarking (Chapter 4).

  When organizations conduct benchmarking, they are mainly concerned with identifying areas for performance improvement; that is, areas where their performance is out of the norm. To this end, we claim that finding outliers in a data warehouse lends itself to viable multidimensional benchmarking. By employing outlier detection techniques, we find what drives organizations to deviate from the norm (i.e. benchmarks). We defined two types of outliers: type-I (organization is an outlier because a small number of underlying units are outliers) and type-II (a majority of them are outliers). Since this it is to look into the makeup of self, we referred to the technique developed in this chapter as “reflective benchmarking”.

- Identifying contexts in which organizations are significant outliers (Chapter 5).

  In this last chapter, we draw our attention to defining the contexts (or subspaces) in which organizations perform most exceptionally (positively or negatively), the primary benefit of multidimensional benchmarking for presenting key drivers for performance gaps. We defined two types of subspaces: contrast subspace and outlying aspect. A contrast subspace is a subspace in which an organization is most similar to a group of organizations while it is most different from another group. This is essentially a model selection problem where one of the two models must be selected on the basis of observed data. Outlying aspect is a subspace where an organization outlies most. To identify outlying aspects, we used rank statistics to compare different subspaces.

While multidimensional benchmarking for computational performance efficiency (i.e. CPU, I/O bandwidth, etc.) is well defined and research materials abundantly available, the counterpart in business performance management somewhat lacks in rigor in definitions and systematic methods. This thesis attempted to provide technical definitions along with more objective and methodical approach to multidimensional benchmarking in the business setting. The thesis focused on establishing technical foundation for multidimensional benchmarking and demonstrating the effectiveness and the efficiency of the techniques devised. These techniques can be applied to a variety of benchmarking scenarios to supplement quantitative benchmarking. Once numerical performance gaps are shown through quantitative benchmarking, the techniques proposed in this thesis can be employed as a next step to identify factors that are driving the gaps or contexts in which the gaps are most significant. These factors or contexts will then become the focal areas in which improvement programs can be created to boost performance. Primary contributions made in the thesis are:

1. Modeling

   We have claimed that organizations do not conduct qualitative benchmarking because qualitative measures are difficult to model because measures cannot be single numerical values. To this end, we technically modelled qualitative measures including significant benchmarks, reflection, contrast subspace, and outlying aspect.

2. Computational Efficiency

   When multiple dimensions are incorporated into analysis, computational efficiency needs to be a consideration due to the well known curse of dimensionality. The time complexity tends to be np-hard. We devised practical heuristics using bounding, pruning, and refining methods. Through experimental results, we showed that our methods are effective and efficient.

3. Application Impact

   If some of the ideas proposed in this thesis are commercialized in a form of a business application, organizations can conduct qualitative benchmarking in a systematic and objective manner. To the best of the author’s knowledge, there is no such application in business today. In view of the fact that benchmarking is so common, such an application can make a significant impact on business.

6.1. Future Directions

Business intelligence applications, such as, multidimensional benchmarking analysis should be interactive by providing summary, drill-down and what-if scenario capabilities. Such applications should take user feedback intelligently, allowing ad-hoc inputs, and enable users to navigate and analyze data, memorizing insights to allow informed decision making. In chapter 3, we defined a benchmark query whereby the performance of a query object with selected properties (i.e. UID attributes) can be compared to the performance of others with the same properties in certain aspects (i.e. combinations of DIM attributes). If the application is to become interactive, the ability to change query properties and aspects on the fly needs to be considered. This type of user interaction can be enabled by allowing benchmark queries at different levels of aggregation hierarchy and reusing the assets already built for techniques, such as, DAM and SIIC. For example, a query object's properties may be age-group and gender (e.g. young males) initially but the user later on wishes to see what the performance gap may look like if he/she removed age-group from the properties such that the query object is now all males. Since the aggregate group, all males, is an ancestor of the aggregate group, young males, DAM can still answer this revised benchmark query efficiently without re-materializing or re-indexing. This applies to other scenarios where attributes are removed from or added to properties (UID) and/or aspects (DIM).

To support what-if scenarios, data mining techniques should be incorporated such that queries, such as, "If we enhanced the education levels of my young male staff, how much sales volume increase can we expect?" can be answered.

Finally, in the author's opinion, among the most significant challenges for multidimensional benchmarking is the intuitive representation of query results. If the techniques developed in this thesis were to be adopted as a common practice in business, the results must be easily understandable. To this end, an effective data visualization approach should be considered with the goal to analyze vast amounts of multidimensional data to visually distill the most valuable and relevant information content. The visual representation should reveal relevant data properties for easy perception by the analyst. An appropriate user interface should be developed such that analysts can focus on tasks at hand, as such, the interfaces should not be overly technical or complex. Visually representing multidimensional data on a 2-dimensional plane is a challenge. Existing multidimensional data representations tend to be somewhat technical requiring users to have some statistics background. To address this, interaction techniques which support seamless and intuitive visual communication between the users and the system should be developed.

References

I. A. Ajibefun

"An Evaluation of Parametric and Non-Parametric Methods of Technical Efficiency Measurement: Application to Small Scale Food Crop Producton in Nigeria," Journal of Agriculture & Social Sciences, ISSN Print: 1813-2235, 2008.

C. Aggarwal

Outlier Analysis, Springer, 2013.

C. Aggarwal, N. Ta, J. Wang, J. Feng and M. J. Zaki

"Xproj: A Framework for Projected Structural Clustering of XML Documents," in *ACM KDD*, 2007.

C. Aggarwal and P. S. Yu

"Outlier detection for high dimensional data," in *ACM Sigmod Record*, ACM, vol 30, pp 37-46, 2001.

C. Aggarwal and P. S. Yu

"Online Analysis of Community Evolution in Data Streams," in *SDM*, 2005.

C. Aggarwal, Y. Zhao and P. S. Yu

"Outlier Detection in Graph Streams," in *ICDE Conference*, 2011.

R. Aggarwal and R. Srikant

"Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB, pp 487-499, 1994.

L. Akoglu, M. McGlohon and C. Faloutsos

"OddBall: Spotting Anomalies in Weighted Graphs," PAKDD’10 Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining, vol. Part II, pp. 410-421, 2010.

S. Albrecht, J. Busch, M. Kloppenburg, F. Metze and P. Tavan

"Generalized radial basis function networks for classification and novelty detection: self-organization of optimal Bayesian decision," *Neural Networks,* vol. 13, pp. 1075-1093, 2003.

American Productivity and Quality Center

"What is best practice?," 1999. [Online]. Available: *http://www.apqc.org*.

F. Anguilli, F. Fassetti and L. Palopoli

"Detecting outlying properties of exceptional objects,” ACM Trans Database Syst 34(1):7:1-7:62, 2009.

F. Anguilli, F. Fassetti, L. Palopoli and G. Manco

"Outlying property detection with numerical attributes,” CoRR abs/1306.3558, 2013.

M. Augusteijn and B. Folkert

"Neural network classification and novelty detection," International Journal on Remote Sensing, 2002.

K. Bache and M. Lichman

UCI machine learning repository, 2013.

D. Baldry, V. Koss and D. Wyatt

"Building Global Card-Issuing Capabilities: A Multi-Dimensional Benchmarking Approach," Booz & Company, 2009.

S.D. Bay and M. J. Pazzani

"Detecting group differences: Mining contrast sets," Data Mining and Knowledge Discovery 5(3):213-246, 2001.

I. Ben-Gal

Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers, Kluwer Academic Publishers, 2005.

K. Beyer and R. Ramakrishnan

"Bottom-up computation of sparse and iceberg cubes" in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pp 359-370, 1999.

K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft

"When is the “nearest neighbor” meaningful?" in *Proceedings of the 7th International Conference on Database Theory*, pp 217-235, 1999.

K. Bhaduri, B. L. Matthews and C. R. Giannella

"Algorithms for speeding up distance-based outlier detection," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pp 859-867, 2011.

C. M. Bishop

"Novelty detection and Neural Network validation," in *IEE Conference on Vision, Image and Signal Processing*, 1994.

C. E. Bogan and M. J. English

"Benchmarking for best practices: winning through innovative adaptation," McGraw-Hill, New York, 1994.

K. Bohm, F. Keller, E. Muller, H.V. Nguyen and J. Vreeken

"CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection," in *Proceedings of the 13th SIAM International Conference on Data Mining*, SDM, pp 198-206, 2013.

D. Breiter and S. Kline

"Benchmarking quality management in hotels," FIU Hospitality Review, 13(2), 45 52, 1995.

M. Breunig, H. P. Kriegel, R. Ng and J. Sander

"LOF: Identifying Density-based Local Outliers," in *ACM SIGMOD*, 2000.

L. Breiman, W. Meisel and E. Purcell

"Variable kernel estimates of multivariate densities," Technometrics 19(2), 135-144, 1977.

C. E. Brodley and M. A. Friedl

"Identifying and Eliminating Mislabeled Training Instances," *Journal of Artificial Intelligence Research,* vol. 11, pp. 131-167, 1996.

Y. Cai, H. K. Zhao, H. Han, R. Y. K. Lau, H. F. Leung and H. Min

"Answering typicality query based on automatically prototype construction," in *Proceedings of the 2012 IEEE/WIC/ACM International Join Conference on Web Intelligence and Intelligent Agent Technology*, Volume 01, pp 362-366, 2012.

R. Camp

“Benchmarking: the Search for Industry Best Practices that Leads to Superior Performance,” ASQC Quality Pres, Milwaukee, Wisconsin, 1989.

A. Campbell

“Outlier Detection: A Survey of Methods and Applications to Healthcare,” PhD Depth Examination, School of Computing Science, Simon Fraser University, 2014.

Canadian Institute for Health Information

"National Health Expenditure Trends, 1975 to 2013," 2013.

Carbon Dioxide Information Analysis Centre (CDIAC)

2015 [Online]. Availble: http://cdiac.ornl.gov/ftp/ndp026b/.

G. A. Carpenter and S. Grossberg

"The ART of adaptive pattern recognition by a self–organising neural network," *IEEE Computer,* vol. 21, pp. 77-88, 1988.

D. Chakrabarti

"AutoPart: Parameter-Free Graph Partitioning," in *PKDD*, 2004.

V. Chandola, A. Banerjee and V. Kumar

"Anomaly detection: A survey," *ACM Comput Surv,* 41(3):15:1-15:58, 2009.

P. Chebyshev

"Sur les valeurs limites des integrales," *Imprimerie de Gauthier-Villars*, 1874.

Y. Chen, F. Dehne, T. Eavis, and A. Rau-Chaplin

"Pnp: Sequential, external memory, and parallel iceberg cube computation," in *Distributed Parallel Databases*, 23(2):99-126, 2008.

L. Chen and G. Dong

"Masquerader detection using OCLEP: One class classification using length statistics of emerging patters," in *Proceedings of International Workshop on INformation Processing over Evoving Networks (WINPEN)*, p 5, 2006.

C. Chen, X. Yan, F. Zhu, J. Han, and P.S. Yu

"Graph OLAP: A multi-dimensional framework for graph data analysis," Knowledge Information Systems, 21(1):41-63, 2009.

W. Cohen

"Fast Effective Rule Induction," in *Machine Learning*, Lake Tahoe, 1995.

S. Cook

"Practical Benchmarking: a Manager's Guide to Creating a Competitive Advantage," Kogan Page, London, 1995.

A. Cox and I. Thompson

"On the appropriateness of benchmarking," Journal of General Management, 23(3), 1 20, 1998.

P. Crook and G. Hayes

"A robot implementation of a biologically inspired method for novelty detection," in *Towards Intelligent Mobile Robots*, 2001.

R. Cross and P. Leonard

"Benchmarking: a strategic and tactical prespective," in *Date, B.G (ed.)*, Managing Quality, 2nd edn, Prentice Hall, New Jersey, 497 513, 1994.

K. Das and J. Schneider

"Detecting Anomalous Records in Categorical Datasets," in *KDD*, San Jose, 2007.

P. Davies

"Benchmarking," Total Quality Management, 309 10, 1990.

G. Dong, J. Han, J.M.W. Lam, J. Pei, and K. Wang

"Mining multi-dimensional constrained gradients in data cubes," in *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pp 321-330, 2001.

G. Dong, J. Han, J.M.W. Lam, J. Pei, K. Wang and W. Zou

"Mining Constrained Gradients in Large Databases," IEEE Transactions on Knowledge and Data Engineering, Volume 16, Number 8, pages 922-938, 2005.

L. Dong and J. Bailey

"Contrast Data Mining: Concepts, Algorithms, and Applications," CRC Press, 2013.

L. Dong and J. Li

"Efficient mining of emerging patterns: discovering trends and differences," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pp 43-52, 1999.

L. Duan, G. Tang, J. Pei, J. Bailey, G. Dong, A. Campbell and C. Tang

"Mining Contrast Subspaces," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, Taiwan, 2014.

L. Duan, G. Tang, J. Pei, J. Bailey, A. Campbell and C. Tang

"Mining Outlying Aspects on Numeric Data," in *ECML/PKDD*, 2014.

N. Duforet-Frebourg and M. G. B. Blum

"Bayesian Matrix Factorization for Outlier Detection: An Application in Population Genetics," *Springer Proceedings in Mathematics & Statistics,* vol. 63, pp. 143-147, 2014.

W. Eberle and L. B. Holder

"Mining for Structural Anomalies in Graph-based Data," in *DMIN*, 2007.

E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo

"A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Aplications of Data Mining in Computer Security*, 6:77-102, 2002.

M. Ester, H. Kriegel, J. Sander and X. Xu

"A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.

R. Fagin, R. Kumar and D. Sivakumar

"Comparing top k lists," In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, pp 28-36, 2003.

C. Faloutsos, F. Korn, A. Labrinidis, Y. Kotidis, A. Kaplunovich and D. Perkovic

"Quantifiable Data Mining Using Principal Component Analysis," Institute for Systems Research, University of Maryland, College Park, MD, 1996.

Forrester

"THE FORRESTER WEVE METHODOLOGY GUIDE ," 2015. [Online]. Available: *https://www.forrester.com/marketing/policies/forrester-wave-methodology.html*.

J. Gao, H. Cheng and P. N. Tan

"A Novel Framework for Incorporating Labeled Examples into Anomaly Detection," in *SIAM International Conference on Data Mining*, 2006.

J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun and J. Han

"On Community Outliers and their Efficient Detection in Information Networks," in *KDD*, Washington, 2010.

Gartner, Inc.

"Gartner Magic Quadrant," 2015. [Online]. Availble: http://www.gartner.com/technology/research/methodologies/research_mq.jsp.

B. Gerber

"Benchmarking: measuring yourself against the best," Training, November, 36 44, 1990.

J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow and H. Pirahesh

"Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," in *Data Mining Knowledge Discovery*, 1(1):29-53, 1997.

H. P. Grünwald

"MDL Tutorial," 2010.

M. Gupta, J. Gao, Y. Sun and J. Han

"Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers," in *KDD*, 2012.

J. Han, M. Kamber and J. Pei

Data Mining: Concepts and Techniques, Morgan Kaufmann, 2011.

J. Han, J. Pei, G. Dong, and K. Wang

"Efficient computation of iceberg cubes with complex measures," in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD'01, pages 1-12, 2001.

W. Hardle

Smoothing Techniques: With Implementation in S. Springer-Verlang, New York, 1990.

W. Hardle, A. Werwatz, M. Muller and S. Sperlich

Nonparametric and Semi-parametric Modelss, Springer Series in Statistics, Springer, 2004.

V. Harinarayan, A. Rajaraman, and J. D. Ullman

"Implementing data cubes efficiently," in *SIGMOD*, 1996.

M. Hauskrecht, M. Valko, I. Batal, G. Clermont, S. Visweswaran and G. F. Cooper

"Conditional Outlier Detection for Clinical Alerting," in *AMIA Annual Symposium*, 2010.

D. Hawkins

Identification of Outliers, Chapman and Hall, 1980.

S. Hawkins, H. He, G. Williams and R. Baxter

"Outlier Detection Using Replicator Neural Networks," *Data Warehousing and Knowledge Discovery,* vol. 2454, pp. 170-180, 2002.

Z. He, P. Wong, B. Kao, E. Lo, and R. Cheng

"Fast evaluation of iceberg pattern-based aggregate queries," In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pp. 2219-2224, 2013.

Z. He, X. Xu and S. Deng

"Discovering cluster-based outliers," Pattern Recognition Letters, vol. 24, no. 9-10, 2003.

Z. He, X. Xu, J. Z. Huang and S. Deng

"A Frequent Pattern Discovery Method for Outlier Detection," *Springer-Verlag Berlin Heidelberg,* no. 3129, p. 26–732, 2004.

V. Hodge and J. Austin

"A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review,* vol. 22, no. 2, pp. 85-126, 2004.

S. Holder, B. Veronese, P. Metcalfe, F. Mini, S. Carter and B. Basalisco

"Cost Benchmarking of Air Navigation Service Providers: A Stochastic Frontier Analysis ," *NERA Economic Consulting*, London, UK, 2006.

J. Hu, F. Wang, J. Sun, R. Sorrentino and S. Ebadollahi

"A Healthcare Utilization Analysis Framework for Hot Spotting and Contextual Anomaly Detection," in *American Medical Informatics Association Annual Symposium*, Chicago, 2012.

M. Hua, J. Pei and A. W. Fu

"Top-k typicality queries and efficient query answering methods on large databases," The VLDB Journal 18(3) 809-835, 2009.

IBNET

"BENCHMARKING METHODOLOGIES," [Online]. Availble: http://www.ib-net.org/en/Benchmarking-Methodologies/PerformanceBenchmarking-DataEnvelopAnalysis.php?L=6&S=2&ss=3, 2015.

IBS Centre for Management Research

"Xerox – The Benchmarking Story," [Online]. Availble: http://www.icmrindia.org/free%20resources/casestudies/xerox-benchmarking-1.htm, Case Code: OPER012, 2006.

T. Imielinski, L. Khachiyan and A. Abdulghani

"Cubegrades: Generalizing association rules," *Data Mining Knowledge Discovery*, 6(3): 219-257 2002.

R. Jacobs

"Alternative Methods to Examine Hospital Efficiency: Data Envelopment Analysis and Stochastic Frontier Analysis," in *Discussion Paper 177*, The University of York Centre for Health Economics, 2000.

A. Jagota

"Novelty detection on a very large number of memories stored in a Hopfield-style network," in *International Joint Conference on Neural Networks*, 1991.

H. Jeffreys

"The Theory of Probability," 3rd Edition, Oxford, 1961.

T. Ji, D. Yang and J. Gao

"Incremental Local Evolutionary Outlier Detection for Dynamic Social Networks," *Lecture Notes in Computer Science, Springer,* vol. 8189, pp. 1-15, 2013.

Q. Jian, A. Campbell, G. Tang and J. Pei

"Multi-level Relationship Outlier Detection," *International Journal of Business Intelligence and Data Mining (IJBIDM),* vol. 7, no. 4, pp. 253-273, 2012.

W. Jin, A. Tung and J. Han

"Mining Top-n Local Outliers in Large Databases," in *ACM KDD*, 2001.

W. Jin, A. Tung, J. Han and W. Wang

"Ranking outliers using symmetric neighborhood relationship," in *PAKDD*, 2006.

M. V. Joshi, R. 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, 2001.

M. V. Joshi, R. Agarwal and V. Kumar

"Predicting rare classes: Can boosting make any weak learner strong?" in *Proceedings of the 8th ACM SIGMOD International Conference on Knowledge Discovery and Data Mining*, pages 297-306, 2002.

M. V. Joshi and V. Kumar

"CREDOS: Classification Using Ripple Down Structure," in *SDM*, 2004.

J. FL. Kay

"Health Care Benchmarking," Medical Bulletin, VOL.12 NO.2, February 2007.

B. Karlof and S. Ostblom

"Benchmarking: a Signpost of Excellence in Quality and Productivity," John Wiley & Sons, Chichester, 1993.

F. Keller, E. Muller and K. Bohm

"HiCS: High contrast subspaces for density-based outlier ranking," ICDE 1037-1048, 2012.

B. Kleine

"Benchmarking for continuous performance improvement: tactics for suces," Total Quality Environmental Management, Spring, 283 95, 1994.

E. Knorr and R. T. Ng

"Algorithms for Mining Distance-based Outliers in Large Datasets," in *VLDB*, 1998.

E. Knorr and R. T. Ng

"Finding intentional knowledge of distance-based outliers," in *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB, pp 211-222, 1999.

R. M. Konijn, W. Duivesteijn, W. Kowalczyk and A. J. Knobbe

"Discovering Local Subgroups, with an Application to Fraud Detection," *Lecture Notes in Computer Science, Springer,* vol. 7818, pp. 1-12, 2013.

P. Kriegel, P. Kroger, E. Schubert and A. Zimek

"Outlier detection in axis-parallel subspace of high dimensional data," PAKDD '09, pp 831-838, 2009.

P. Kriegel, M. Schubert and A. Zimek

"Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pp 444-452, 2008.

T. S. Kuhn

The Structure of Scientific Revolutions, Chicago: University of Chicago Press, 1962.

J. Kulmala

"APPROACHES TO BENCHMARKING," Finnish Employers' Management Development Institute, FEMDI, http://www15.uta.fi/yksikot/entrenet/hankerekisteri/hanke5_benchmarking.htm, [accessed] 2015.

S. Kullback and R. Leibler

"On information and sifficiency," *The Annals of Mathematical Statistics,* 1951.

L.V.S. Lakshmanan, J. Pei and J. Han

"Quotient cube: How to summarize the semantics of a data cube," *In Proceedings of the 28th Internaltional Conference on Very Large Data Bases, VLDB '02, pages 778-789, VLDB Endowment,* 2002.

J. Li , K.-Y. Huang, J. Jin and J. Shi

"A survey on statistical methods for health care fraud detection," *Science + Business Media, Springer,* 2007.

E. Lo, B, Kao, W.S. Ho, S.D. Lee, C.K. Chui, and D.W. Cheung

"OLAP on sequence data," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD 08, pages 649-660, 2008.

Z. Long, A. Bakar and A. Raz

"Frequent Pattern using Multiple Attribute Value for Itemset Generation," in *Conference on Data Mining and Optimization*, Selangor, 2011.

Z. Long, A. R. Hamdan and A. Bakar

"Anomaly Based On Frequent-Outlier for Outbreak Detection in Public Health Surveillance," *World Academy of Science, Engineering and Technology,* vol. 7, 2013.

A. Lourenco, H. Silva, C. Carreiras and A. Fred

"Outlier Detection in Non-intrusive ECG Biometric System," *Lecture Notes in Computer Science, Springer,* vol. 7950, pp. 43-52, 2013.

C. A. K. Lovell

"Frontier Analysis in Health Care," *Department of Economics, University of Georgia*, Athens, GA 30602, USA, 2003

C. McNair and K. Leibfried

"Benchmarking; a Tool for Continuous Improvement," *Harper Business*, New York, 1992.

Microsoft

"Data Warehousing and OLAP," *Technet Library, 2015* [Online]. Availble: https://technet.microsoft.com/en-us/library/aa197903(v=sql.80).aspx.

J. Mourão-Miranda, D. R. Hardoon, T. Hahn, A. F. Marquand, S. C. R. Williams, J. Shawe-Taylor and M. Brammer

"Patient classification as an outlier detection problem: An application of the One-Class Support Vector Machine," *NeuroImage,* no. 58, pp. 793-804, 2011.

E. Muller, I. Assent, P. Iglesias, Y. Mulle and K. Bohm

"Outlier ranking via subspace analysis in multiple view of the data," ICDM '12, pp529-538, 2012a.

E. Muller, F. Keller, S. Blanc and K. Bohm

"OutRules: A framework for outlier descriptions in multiple context spaces," ECML/PKDD (2), pp 828-832, 2012b.

E. Muller, M. Schiffer and T. Seidl

"Statistical selection of relevant subspace projections for outlier ranking," in *Proceedings of the 27th IEEE International Conference on Data Engineering*, ICDE, pp 434-455, 2011.

A. Nairac, N. Townsend , R. Carr, S. King, P. Cowley and L. Tarassenko

"A system for the analysis of jet system vibration data," *Integrated ComputerAided Engineering,* vol. 6, no. 1, p. 53 – 65, 1999.

R. T. Ng, A. Wagner, and Y. Yin

"Iceberg-cube computation with pc clusters," In *Proceedings of the 2001 AM SIGMOD International Conference on Management of Data,* SIGMOD -01, pages 25 – 36, 2001.

M. Nguyen, E. Muller and J. Vreeken

"CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection," SDM 198-206, 2013.

C. Noble and D. Cook

"Graph-Based Anomaly Detection," in *SGKDD*, Washington, 2003.

P. K. Novak, N. Lavrac and G. I. Webb

"Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining," Journal of Machine Learning Research, 10:377-403, 2009.

Y. A. Ozcan

"Health Care Benchmarking and Performance Evaluation: An Assessment using Data Envelopment Analysis (DEA)," International Series in Operational Research & Management Science, Volume 210, Springer, 2008.

C.H. Papadimitriou and M. Yannakakis

"Optimization, approximation, and complexity classes," Journal of Computer and System Sciences, 3(3):425-440, 1991.

R. Paravastu, H. Kumar and V. Pudi

"Uniqueness mining," in *Proceedings of the 13th International Conference on Database Systems for Advanced Applications*, DASFAA, pp 84-94, 2008.

C. Phua, D. Alahakoon and V. Lee

"Minority report in fraud detection: classification of skewed data," in *ACM SIGKDD Explorations Newsletter*, 6(1):50-59, 2004.

S. Ramaswamy, R. Rastogi and K. Shim

"Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD, pp 427-438, 2000.

R. Rymon

"Search through systematic set enumeration," in *Principle of Knowledge Representation and Reasoning*, 1992.

S. Sarawagi, R. Agrawal and N. Megiddo

"Discovery-driven exploration of OLAP data cubes," In *Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology,* EDBT '98, pages 168-182, 1998.

D. W. Scott

"Multivariate Density Estimation: Theory, Practice, and Visualization," *Wiley Series in Probability and Statistics,* Wiley, New York, 1992.

Y. Shan, D. W. Murray and A. Sutinen

"Discovering inappropriate billings with local density based outlier detection method," in *The 8th Australasian Data Mining Conference*, Melbourne, 2009.

H. D. Sherman and J. Zhu

"DATA ENVELOPMENT ANALYSIS EXPLAINED," Service Productivity Management, Springer, XXII, 328p, 2006.

K. Singh and and S. Upadhyaya

"Outlier Detection: Applications And Techniques," *International Journal of Computer Science Issues,* vol. 9, no. 1, 2012.

B. W. Silverman

"Density Estimation for Statistics and Data Analysis," Chapman and Hall/CRC, 1986.

M. C. Sokol, K. A. McGuigan, R. R. Verbrugge and R. S. Epstein

"Impact of Medication Adherence on Hospitalization Risk and Healthcare Cost," *Medical Care,* vol. 43, 2005.

I. Steinwart, D. Hush and C. Scovel

"A Classification Framework for Anomaly Detection," *Journal of Machine Learning Research,* vol. 6, p. 211–232, 2005.

J. Sun, S. Papadimitriou, P. Yu and C. Faloutsos

"Graphscope:Parameter-free Mining of Large Time-Evolving Graphs," in *KDD*, San Jose, 2007.

J. Sun, Y. Xie, H. Zhang and C. Faloutsos

"Less is More: Compact Matrix Representation of Large Sparse Graphs," in *SIAM Conference on Data Mining*, 2007.

G. Tang, J. Bailey, J. Pei and G. Dong

"Mining multidimensional contextual outliers from categorical relational data," in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, SSDBM, pp 43:1-43:4, 2013.

D. M. Tax and R. P. Duin

"Support Vector Data Description," *Machine Learning,* vol. 54, no. 1, p. 4566, 2004.

O. Taylor and D. Addison

"Novelty Detection Using Neural Network Technology," in *COMADEN*, 2000.

J. Thongkam, G. Xu, Y. Zhang and F. Huang

"Support Vector Machine for Outlier Detection in Breast Cancer Survivability Prediction," *Lecture Notes in Computer Science, Springer,* vol. 4977, pp. 99-109, 2008.

H. Tong and C. Y. Lin

"Non-Negative Residual Matrix Factorization with Application to Graph Anomaly Detection," in *SDM Conference*, 2011.

TPC-H

Available: http://www.tpc.org/tpch/

R. Vilalta and S. Ma

"Predicting rare events in temporal domains," In *Proceedings of 2002 IEEE International Conference on Data Mining*, pages 474-481, 2002.

K. Vaziri

"Using competitive benchmarking to set goals," Quality Progress, October, 81 5, 1992.

N. Wale, X. Ning and G. Karypis

"Trends in Chemical Data Mining," in *Managing and Mining Graph Data*, 2010.

F. Wang, S. Chawla and D. Surian

"Latent Outlier Detection and the Low Precision Problem," in *ODD'13*, Chicago, 2013.

J. Wang, J. Han and J. Pei

"Closed Constrained-Gradient Mining in Retail Databases," IEEE Transactions on Knowledge and Data Engineering, Volume 18, Number 6, pages 764-769, IEEE Computer Society, 2006

L. Wang, H. Zhao, H. Dong and G. Li

"On the complexity of finding emerging patterns," Theoretical Computer Science, 335(1) 15-27, 2005.

G. Watson

"Strategic Benchmarking: How to Rate Your Company's Performance Against the World's Best," John Wiley & Sons, Canada, 1993.

W. Webber, A. Moffat and J. Zobel

"A similarity measure for indefinite rankings," ACM Trans Inf Syst 28(4):20:1-20:38, 2010.

G. Weiss and H. Hirsh

"Learning to predict rare events in event sequences," In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 359-363, 1998.

Wikipedia

"Inverted Index," March 2016. [Online]. Available: https://en.wikipedia.org/wiki/Inverted_index

Wikipedia

"Magic Quadrant," March 2015. [Online]. Available: http://en.wikipedia.org/wiki/Magic_Quadrant

W. K. Wong, A. Moore, G. Cooper and M. Wagner

"Rule-Based Anomaly Pattern Detection for Detecting Disease Outbreaks," in *AAAI*, 2002.

S. Wrobel

"An algorithm for multi-relational discovery of subgroups," in *Proceedings of the 1st European Symposium on Priciples of Data Mining and Knowledge Discovery*, , pp 78-87, 1997.

S. Wu and F. Crestani

"Methods for ranking information retrieval systems without relevance judgements," in *Proceedings of the 2003 ACM Symposium on Applied Computing*, ACM, New York, pp811-816, 2003.

World Health Organization

"Adherence to Long-term Therapies: Evidence for Action," January 2003. [Online]. Available: http://www.who.int/chp/knowledge/publications/adherence_report/en/.

A. Yeung

"Matrix Factorization: A Simple Tutorial and Implementation in Python," 16 September 2010. [Online]. Available: http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/.

M. Yousaf, M. Welzl and B. Yener

"Accurate Shared Bottleneck Detection Based On SVD and Outliers Detection," 2008.

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 Berkely Laboratory, 2005.

D. Zhang, D. Gatica-Perez, S. Bengio and I. McCowan

"Semi-supervised Adapted HMMs for Unusual Event Detection," in *Computer Vision and Pattern Recognition*, 2005.

Y. Zhang, N. Meratnia and P. Havinga

"Adaptive and Online One-Class Support Vector Machine-based Outlier Detection Techniques for Wireless Sensor Networks," in *Advanced Information Networking and Applications Workshops*, 2009.

K. Zhang, S. Shi, H. Gao and J. Li

"Unsupervised outlier detection in sensor networks using aggregation tree," in *Advanced Data Mining and Applications*, pages 158-169, 2007.

Y. Zhao, P.M. Deshpande and J.F. Naughton

"An array-based algorithm for simultaneous multidimensional aggregates," in *Proc. 1997 ACM-SIGMOD International Conference on Management of Data*, pages 159-170, 1997.

S. Zhu, Y. Wang and Y. Wu

"Health Care Fraud Detection Using Nonnegative Matrix Factorization," in *The 6th International Conference on Computer Science & Education*, SuperStar Virgo, Singapore, 2011.

A. Zimek, E. Schubert and H. P. Kriegel

"A survey on unsupervised outlier detection in high-dimentional numerical data," Stat Anal Data Min, 5(5):363-387, 2012.