2018

# Knowledge discovery techniques for transactional data model

Piyush Lakhawat
*Iowa State University*

# Knowledge discovery techniques for transactional data model

by

**Piyush Lakhawat**

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Arun K. Somani, Major Professor
Jack H Lutz
Joseph Zambreno
Phillip H Jones
Yong Guan
Zhengdao Wang

Iowa State University

Ames, Iowa

2018

## DEDICATION

I would like to dedicate this thesis to my Family and my advisor Dr. Arun K. Somani. I will be forever grateful for their support, guidance, encouragement and love.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

## ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to all my professors at ISU. Starting with Dr. Arun K. Somani, who provided immeasurable guidance and support throughout my time in Ames. I would also like to express my sincerest gratitude to all my POS committee members who provided crucial suggestions and directions at important times. I have to acknowledge the support from all my research group members, including Dr. Mayank Misha, Dr. Pratik Mishra, Dr. Parijat Shukla and every one else. All my friends in Ames and outside, made my time at ISU very special and it would not have been the same without them.

## ABSTRACT

In this work we give solutions to two key knowledge discovery problems for the Transactional Data model: Cluster analysis and Itemset mining. By knowledge discovery in context of these two problems, we specifically mean novel and useful ways of extracting clusters and itemsets from transactional data. Transactional Data model is widely used in a variety of applications. In cluster analysis the goal is to find clusters of similar transactions in the data with the collective properties of each cluster being unique. We propose the first clustering algorithm for transactional data which uses the latest model definition. All previously proposed algorithms did not use the important utility information in the data. Our novel technique effectively solves this problem. We also propose two new cluster validation metrics based on the criterion of high utility patterns. When comparing our technique with competing algorithms, we miss much fewer high utility patterns of importance than them.

Itemset mining is the problem of searching for repeating patterns of high importance in the data. We show that the current model for itemset mining leads to information loss. It ignores the presence of clusters in the data. We propose a new itemset mining model which incorporates the cluster structure information. This allows the model to make predictions for future itemsets. We show that our model makes accurate predictions successfully, by discovering 30-40% future itemsets in most experiments on two benchmark datasets with negligible inaccuracies. There are no other present itemset prediction models, so accurate prediction is an accomplishment of ours.

We provide further theoretical improvements in our model by making it capable of giving predictions for specific future windows by using time series forecasting. We also perform a detailed analysis of various clustering algorithms and study the effect of the Big Data phenomenon on them. This inspired us to further refine our model based on a classification problem design. This addition allows the mining of itemsets based on maximizing a customizable objective function made of different prediction metrics. The final framework design proposed by us is the first of its kind to make itemset predictions by using the cluster structure. It is capable of adapting the predictions to a specific future window and customizes the mining process to any specified prediction criterion. We create an implementation of the framework on a Web analytics data set, and notice that it successfully makes optimal prediction configuration choices with a high accuracy of “0.895”.

# CHAPTER 1. INTRODUCTION

The goal of our research is to solve knowledge discovery problems for the Transactional data model. In this work, specifically by knowledge discovery we mean extracting useful abstractions from the data by using following two data mining techniques: cluster analysis and itemset mining. We first introduce the broader research domain pertaining to this work, and the specific direction chosen by us. We then highlight our contributions made towards the set research goal. Recent trends from the perspective of hardware and software development, as well as analytics and data mining techniques are discussed. We also discuss the key motivations behind our choice of research direction. We present these motivations in a progressive fashion which also acts as a window into the flow of following chapters of the thesis and provides an outline for it.

## 1.1 Research Problem

All business and scientific applications are increasing their use of data for better forecasting, planning and decision making in their respective domains. The continuous progress in hardware and software technologies in data collection, storage, communication and processing has greatly accelerated this process. The data mining and analytics techniques continue to evolve the knowledge discovery process and scale to increasing data levels in terms of volume, velocity and variety. One of the most important focus areas is developing new techniques which will allow rich modeling capabilities and search over multiple dimensions of data for valuable insights. While the advancement in computing capabilities has largely been regarded as the trigger for the Big Data phenomenon, modern data models and analytics have critically helped in keeping its promise. We believe this evolution of knowledge discovery techniques still has long way to go, and will need to continuously adapt to the changing nature of data and analytics.

Two core problems of knowledge discovery discussed in this work are cluster analysis and itemset mining. These two techniques find widespread use in a variety of applications because of their simple and intuitive abstractions which apply to many data sources. Our focus is specifically on the Transactional data format. Varied application areas like market basket analysis, web analytics and bio informatics use the transactional data format, while applying clustering and itemset mining techniques to address their analytics problems. Often both these sets of techniques are applied on the same data set to study its different aspects but independently. Therefore, establishing synergy between these two techniques can be very beneficial. It can lead to richer information extraction and provide new knowledge discovery avenues.

### 1.1.1 Research Goal

The primary research goal is to improve knowledge discovery (i.e. extracting useful abstractions) in the use of current itemset mining and clustering techniques for transactional data model. Specifically we want to:

1. Improve the present solutions to the problem of cluster analysis for transactional data.
2. Evolve the itemset mining model to overcome an information loss problem.
3. Identify the future challenges in analytics, and address them for the itemset mining problem.

### 1.1.2 Research Contribution

Our specific research contributions are the following:

1. We develop and design a novel clustering algorithm to capture utility information in transactional data. It facilitates a more realistic clustering analysis. We introduce two new cluster validation measures to quantify and compare cluster qualityy.
2. We extend the current utility itemset mining model by implicitly using the cluster structure in the mining process. This adds a useful predictive aspect to the model.

3. We add an adaptive aspect to the above predictive model by anticipating cluster contributions via time series forecasting techniques.

4. We perform an examination of the effect of Big Data phenomenon on the broad domain of cluster analysis.

5. We propose a framework to address some of the future challenges of itemset mining

## 1.2 Recent trends

The advancement in sensing, networking, memory and computing technologies allows more data to be collected, transmitted, stored and processed. A recent article (WSDaily (2012)) states that Wal-Mart manages more than one million transactions every hour, while their databases store more than 2.5 petabytes of data. The scientists at CERN now process up to 30 petabytes of data annually (CERN (2015)) to understand particle physics better. The increase in use of data to look for intelligence is a phenomenon which is common in almost all range of business and scientific applications. Huge amounts of data are being used to either improve existing predictive models or discover new knowledge.

### 1.2.1 Hardware and software

Kambatla et al. (2014) discuss the recent trends in big data analytics in detail regarding hardware, software and applications domain. The need for faster memory access times (along with performance support for random accesses) have resulted in increased interest in non-volatile memories like Flash based (like SSD) or Phase change memory (PCRAM). The traditional hard drives are still the most cost effective, but due to the performance needs of new data intensive applications, memory architectures are being designed with these faster memories in them. They are placed either at a memory hierarchy level same as the traditional hard drives or one above them.

With the variety of computing platforms available ranging from accelerators like Intel MIC, GPGPUs to various multi-core architectures, there is lot of scope for customization at the hardware and software level. Based on their type the recent data centered applications are designed to make maximum use of the available computing hardware. Network technologies like the recent DCTCP (Alizadeh et al. (2010)) have started to adapt to the needs of Data centers by designing custom protocols for map-reduce type applications. Data centers also bring another challenge, i.e. their energy consumption. Technologies like dynamic voltage and frequency scaling (DVFS) have been developed to address this challenge.

There has been a significant development in the software domain as well. Brewer (2000) propose an interesting theorem that- You can only have two of the following properties in a shared data system: Consistency, Availability and Tolerance towards network partition. With some reflection, this theorem is quite intuitive. For example, if we most desire consistency then we either do not finely partition the data so that book keeping is fast (hence availability) or be willing to spend time in maintaining consistency among many partitions (hence tolerance for partitions). This observation has led to the development of different types of software which give more importance to one of these properties over the other.

### 1.2.2 Data mining and analytics techniques

The progress in hardware and software domain has accelerated progress in data mining techniques research. Tsai (2012) give a summary data mining works with respect to authors, institutes and place of publication from 1989 to 2009. While Liao et al. (2012) do a survey of how data mining techniques and their applications have evolved from 2000 to 2011. For data mining applications they found increasing trends in: bioinformatics and customer relationship management. Both bioinformatics and customer relationship management are very data centered applications with volumes of data being created continuously. Thus it is understandable that there is increasing interest for using data mining techniques in these application areas. While among the data mining and analytics techniques they discovered increasing trends in: neural networks, decision trees, clustering and association rule mining. To briefly overview these:

1. Neural networks are a class of models in machine learning which include interconnected neurons (inspired from biology) where the connections are characterized by weights. These weights keep tuning based on the inputs fed to network. This simple yet effective design of neural networks makes them very powerful because they can learn from a vast variety of data and have shown surprisingly good prediction accuracies. With increasingly more computational power available neural networks with ascending levels of granularity are stacked in many layers to model and predict/train with higher levels of abstractions in the data. Each layer feeds data to the next one until it reaches the top while learning in the process. With the ability to model many abstraction types, this technique is hugely popular now and is referred to as deep learning due to the many layers in the model.

2. Decision trees use a tree type model where the root node is the starting point and the leaf nodes are possible outcomes. The internal nodes are various possible events in between. Going over the various event traces the tree can be trained. They are good for visualizations and can be modeled for different scenario types.

3. Clustering algorithms are a class of techniques which enable creating groups of similar objects from a data set of similar objects. The metrics used for defining clustering and the criterion for similarity varies by the goal of clustering and the object types. Since their no training labels in the clustering problem they fall in category of unsupervised learning.

4. Association rule mining is used to discover frequently occurring patterns of interest in the data and then develop association rules out them. For example an association rule can be of type (object A, object B) $\rightarrow$ (object C) can be developed from a frequently occurring pattern of (object A, B and C), meaning occurrence of object A and object B in an instance imply object C with some confidence.

Kriegel et al. (2007) discuss the future trends in development of data mining techniques and identify the following as key points:

1. Ability to mine complex objects of arbitrary type.
2. Model and derive useful information from the temporal aspects of data.
3. Perform fast, transparent and structured data preprocessing.
4. Increased usability of the techniques.

Shah et al. (2010) identify response time, access pattern, working set, data type, read vs write, processing complexity as the classifying dimension of various data center workloads. Based on these they identify five data engines which can model (and process) most type of workloads: distributed sort, in-memory index search, recommendation systems, data deduplication and video uploading/streaming.

After reviewing the research trends and key aspects in various related domains, in the following section we make the case for motivation behind this work.

## 1.3 Research motivation

From the study of current trends in the realm of data mining and analytics we observe the following four important directions in the research:

1. Hardware (computing, memory and network) needs to continuously evolve to support needs of increasing data processing.
2. Software should adapt to efficiently utilize the new hardware.
3. Techniques should evolve to scale to the increasing data levels along with more emphasis on security and energy consumption issues.

4. Focus on techniques which will allow rich modeling capabilities and search over multiple dimensions of data for valuable insights. These will solve the future data challenges outside of sheer speed and size aspects.

Our interest is in the fourth direction in the above list. We believe the huge resurgence in popularity of neural network over the last few years has been due to their ability to learn from and model many abstract types of inputs. They are also able to model/learn from multiple dimensions of data by using many layered (deep learning) neural networks. This is a common goal for all knowledge discovery techniques and they will continue to progress in this direction.

Our focus being on the Transactional data format because of its widespread applications, we realized pressing need for the application of the above idea in its analysis. As we mentioned before two key analysis techniques used on it are cluster analysis and itemset mining. The data model itself had evolved in order to be more realistic but this transition had not occurred for discovery of clusters, so our first motivation was to make that transition. Following that we realized that the way itemset mining was performed was leading to information loss. This was due to ignoring the knowledge of clusters present in the data. Correcting this was our second motivation. Following that we noticed that there is a further temporal aspect to the cluster structure which can aid in making adaptive predictions about itemsets, which was our third motivation. To make our knowledge discovery solutions adept to the future data challenges, we wanted to perform a study of how the broad field of cluster analysis was affected by the Big Data phenomenon. To understand the associated challenges and adaptations was our fourth motivation. Finally we wanted to propose a framework which addressed some of the future challenges for our itemset prediction model, which was our fifth motivation. There following six chapters in the thesis show our work associated with each of these progressive motivations. In the final chapter we provide our conclusive remarks and future work directions.

# CHAPTER 2. THE TRANSACTIONAL DATA MODEL, ANALYTICS AND APPLICATIONS

In this chapter, we first introduce and then formally describe the Transactional data model. We start with the Frequency based model and then discuss the shortcomings of it before defining the evolved utility based model. We also discus the semantics of performing cluster analysis on Transactional data. Finally we discus three key application areas where the Transactional data model plays an important role, and the significance of analytics based on it. Illustrations and discussions included in this chapter are derived from Lakhawat et al. (2017) and a selected for publication work with the Springer group in a special journal edition for selected and revised proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2017, held in Funchal, Madeira-Portugal, in November 2017.

## 2.1 Illustrative example

As the name suggests, the building block of this model is a transaction. While the term transaction here defines an abstract concept, the meaning is very similar to the everyday understanding of the word. When a person goes to a store and buys various objects in one specific visit, he creates a transaction constituting those objects. The transaction also usually has an identifier (an ordering of some type) associated with it. The collection of transactions over a particular day/week/season represents the purchase activity for that period. The transactional data model aims to capture that activity/phenomenon with the goal of deriving (mostly) generalizable insights from it. For example, customers who buy a particular brand of coffee, very often also buy a particular brand of cereal. Another type of insight can be of the form, that among all transactions there are (say) 5 broad groups which have similar objects within them and fairly unique objects among them. These might be attributed to the different customer types at the store.

Similar structures of data are present varied applications domains like web analytics (Ahmed et al. (2009); Li et al. (2008)), bio-informatics (Alves et al. (2009); Naulaerts et al. (2015)) etc, which therefore all employ the Transactional data model. The constituting objects of transactions are called items. The set of all unique items is called the set of Itemtypes. An important concept of analysis here is something called an Itemset. Itemset is essentially any non-null subset of the set of Itemtypes. An itemset is said to have "support" from an transaction if the transaction contains all the items present in the itemset. The corresponding analysis performed based on this concept is identifying all itemsets (called the set of frequent itemsets) which have support from more than a fixed number (say $\Phi$) of transactions.

For illustration, consider a small example of FIM as presented in Figure 2.1 (extracted from Lakhawat et al. (2017)). Dataset D represents a set of transactions from a retail store. Set I represents various item types. The set of Frequent Itemsets contains all itemsets which are present in two or more (as $\Phi = 2$) transactions in the dataset D. In real world scenarios when the threshold values ($\Phi$) are large, a frequent itemset of type (A, B) leads to an example association rule of type A $\rightarrow$ B. The practical implication of such an association rule depends on the application domain. In market basket analysis it can imply a customer buying item A is likely to buy item B as well, so A and B should be advertised together.

## 2.2 Formal definition of frequency based model

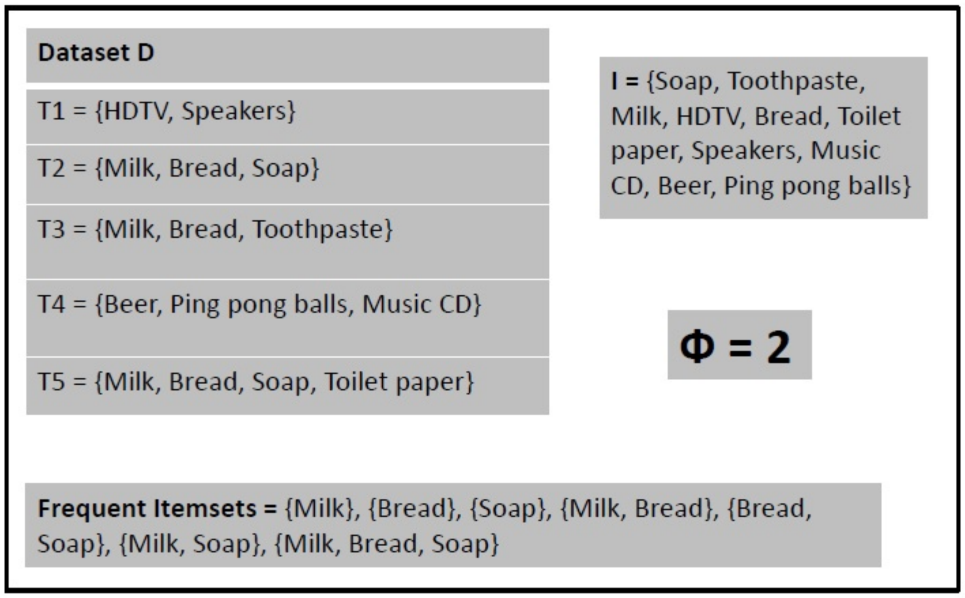
Let us now formally define the problem of Frequent itemset mining (FIM). Over the following space we will expand this model, but will only add incremental model features to it.

$$I = \{a_1, a_2, \ldots, a_M\} = \text{Set of distinct item types} \tag{2.1}$$

$$D = \{T_1, T_2, \ldots, T_N\} = \text{Transaction dataset} \tag{2.2}$$

$$\text{where each } T_i = \{x_1, x_2, \ldots\}, x_k \in I$$

$$\text{itemset}(X) \text{ of size k} = \{x_1, x_2, \ldots, x_k\} \tag{2.3}$$



**Dataset D**

T1 = {HDTV, Speakers}

T2 = {Milk, Bread, Soap}

T3 = {Milk, Bread, Toothpaste}

T4 = {Beer, Ping pong balls, Music CD}

T5 = {Milk, Bread, Soap, Toilet paper}

**I =** {Soap, Toothpaste, Milk, HDTV, Bread, Toilet paper, Speakers, Music CD, Beer, Ping pong balls}

**Φ = 2**

**Frequent Itemsets =** {Milk}, {Bread}, {Soap}, {Milk, Bread}, {Bread, Soap}, {Milk, Soap}, {Milk, Bread, Soap}

Figure 2.1 Illustration of Frequent Itemset Mining

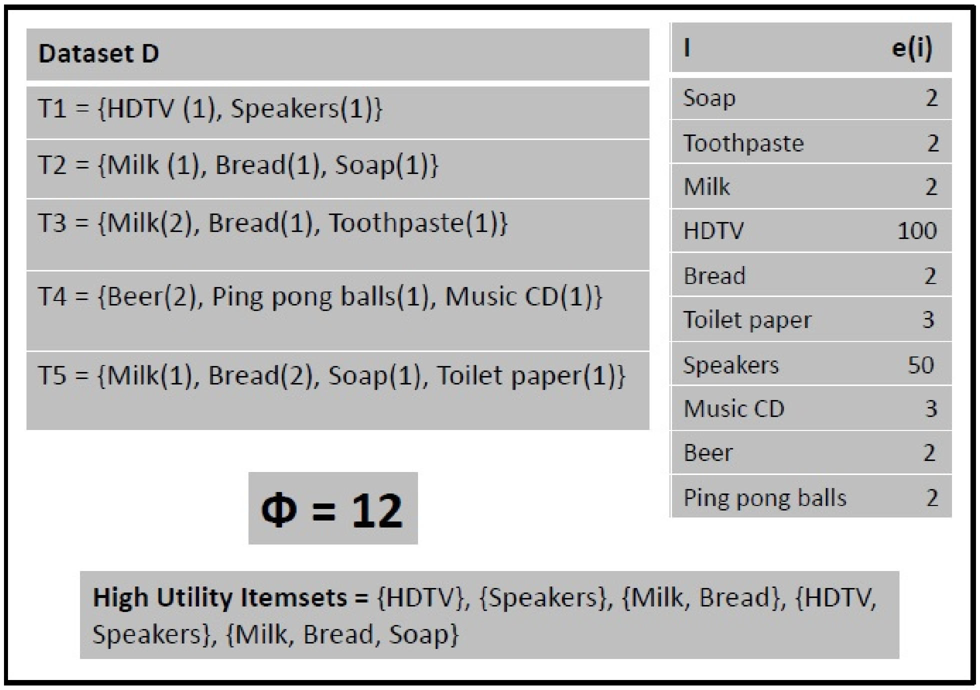
$$SC(X) = |\{T_i \text{ such that } X \in T_i \wedge T_i \in D\}| \tag{2.4}$$

$$\text{where } SC(X) = \text{support count for } X \text{ in } D$$

$$\text{Frequent itemsets} = \{X \text{ such that } SC(X) \geq \Phi\} \tag{2.5}$$

## 2.3 Evolution to utility based model

FIM problem lacks the ability model the relative importance of various item types. For example in Figure 2.1, above a HDTV and a pack of soap have the same unit of importance. While in reality the profit yield of a unit sale of a HDTV is expected to be much more than that of soap. For a frequency threshold of two, the HDTV and Speakers could not make it to the list of frequent itemsets. Another limitation of FIM is the inability to model the occurrence frequency of items in a particular transaction. For example, it is possible that in transaction T2 from Figure 2.1 the customer bought one pack of bread, while in transaction T3 customer bought two packs of bread. The FIM problem model is unable to differentiate between these. To overcome these limitations Utility Itemset Mining (UIM) emerged as an evolved version of FIM.



| Dataset D |
|---|
| T1 = {HDTV (1), Speakers(1)} |
| T2 = {Milk (1), Bread(1), Soap(1)} |
| T3 = {Milk(2), Bread(1), Toothpaste(1)} |
| T4 = {Beer(2), Ping pong balls(1), Music CD(1)} |
| T5 = {Milk(1), Bread(2), Soap(1), Toilet paper(1)} |

| I | e(i) |
|---|---|
| Soap | 2 |
| Toothpaste | 2 |
| Milk | 2 |
| HDTV | 100 |
| Bread | 2 |
| Toilet paper | 3 |
| Speakers | 50 |
| Music CD | 3 |
| Beer | 2 |
| Ping pong balls | 2 |

**Φ = 12**

**High Utility Itemsets =** {HDTV}, {Speakers}, {Milk, Bread}, {HDTV, Speakers}, {Milk, Bread, Soap}

Figure 2.2 Illustration of Utility Itemset Mining

The UIM version of the problem from Figure 2.1 is presented in Figure 2.2 (extracted from Lakhawat et al. (2017)). In Figure 2.2, next to items in the transactions, the parenthesis contain occurrence frequency for the item in that particular transaction. The right side of Figure 2.2 lists various item types. e(i) represents the relative importance of each item type i. In this case they can be interpreted as the profit associated with the unit sale of that item. The profit associated with an itemset (referred to as absolute utility of the itemset) is the calculated as the sum of profit made by that itemset in all transactions it occurs in. For example, the itemset (milk, Bread) occurs in T2 (profit made = 1 x Milk + 1 x Bread = 4), T3 (profit made = 2 x Milk + 1 x Bread = 6) and T5 (profit made = 1 x Milk + 2 x Bread = 6). Therefore the absolute utility of itemset (Milk, Bread) will be 4 + 6 + 6 = 16. The threshold (Φ) for UIM is a combination criterion of frequency and utility/importance. For the problem in Figure 2.2, the set of High Utility Itemsets contain all itemsets with absolute utility more than 12 (Φ).

## 2.4 Formal definition of utility based model

As mentioned earlier, FIM lacks two key modelling capabilities. It cannot model difference in relative importance of various item types and the frequency of an item type in a transaction. UIM overcomes these limitations. UIM problem builds up on the FIM problem with additional information of external and internal utilities for items. External utility is a measure of unit importance of an item type in the analysis. This is a transaction independent utility. Internal utility is a transaction specific utility. This is typically the frequency or some measure of quantity of an item in the transaction. It can formally be defined as follows.

$$eu(a_i) = \text{ external utility of item type } a_i \tag{2.6}$$

$$iu(a_i, T_j) = \text{ internal utility of } a_i \text{ in } T_j \tag{2.7}$$

The absolute utility of an item in a transaction is defined as the product of its internal and external utility.

$$au(a_i, T_j) = eu(a_i) * iu(a_i, T_j) \tag{2.8}$$

Absolute utility of an itemset in a transaction is the sum of absolute utilities of its constituent items.

$$au(X, T_j) = \sum_{x_i \in X} au(x_i, T_j) \tag{2.9}$$

Absolute utility of a transaction (also called transaction utility) is the sum of absolute utilities of all its constituent items.

$$TU(T_j) = \sum_{x_i \in T_j} au(x_i, T_j) \tag{2.10}$$

Absolute utility of an itemset in the dataset $D$ is the sum of absolute of that itemset in all transactions that it occurs in.

$$au(X, D) = \sum_{X \in T_j \wedge T_j \in D} au(X, T_j) \tag{2.11}$$

The set of HUI is the collection of all itemsets which have absolute utility more than or equal to $\Phi$ in the dataset D.

$$\text{set of HUI} = \{X \text{ s.t. } au(X, D) \geq \Phi\} \tag{2.12}$$

The following three concepts are used in the solution techniques of UIM to achieve a downward closure property for efficient candidate generation similar to the FIM problem: Transaction weighted utility(TWU) of itemset X in dataset $D$ is the sum of transaction utilities of transactions in which the itemset X occurs.

$$TWU(X, D) = \sum_{X \in T_j \wedge T_j \in D} TU(T_j) \tag{2.13}$$

Set of high transaction weighted utility itemsets (HTWUI) is a collection of all itemsets which have transaction weighted utility more than or equal to $\Phi$ in the dataset $D$.

$$\text{Set of HTWUI } = \{X \text{ s.t. } TWU(X) \geq \Phi\} \tag{2.14}$$

TWDC property (Tseng et al. (2015); Liu et al. (2005a)):"The transaction-weighted downward closure property states that for any itemset $X$ that is not a HTWUI, all its supersets are low utility itemsets."

The goal of UIM is to find the set of all high utility itemsets for a given $\Phi$. Here threshold $\Phi$ is a combination criterion of utility and frequency rather than a solely frequency based one in FIM.

## 2.5 Formal definition of cluster analysis for Transactional data

The key idea in performing cluster analysis on any type of data is to identify clusters (groups/collections) of the constituent data objects from the entire dataset, such that the members of each cluster are "similar" to each other and the collective attributes of clusters are unique among each other. The definition of "similarity" is usually subjective and depends of the application domain as well as the type of data. We can intuitively imagine that performing cluster analysis for transactional data would mean identifying groups of transactions following the above rationale.

Clustering is a very broad analysis discipline and comes under the umbrella of Unsupervised machine learning. It transcends far beyond the model and applications of transactional data. Transactional data is special type of data where each transaction is a non-uniform collection of categorical (non-numerical) items, along with other attributes defined associated with them (utilities). This restricts the scope of clustering techniques which can be applied to them. While most categorical clustering algorithms can be applied to them, there are also specific algorithms designed specifically for transactional data. We will hold off on formally defining certain technique specific concepts related to clustering here, but will define the more broader aspects.

A cluster $C_k$ of is a subset of transactions from $D$.

$$C_k = \{T_1, T_2 \ldots T_k | T_i \in D\} \tag{2.15}$$

$$I_{C_k} = \{a_i | a_i \in T_j \wedge T_j \in C_k\} = \text{item types in } C_k \tag{2.16}$$

$C$ is the set of all given clusters.

## 2.6 Key application areas

In this section we discuss three key application areas of Transactional data. While this data model is not restricted to only these domain, it has witnessed more application in these. Both the analytics techniques (itemset mining and clustering) are applied and have their respective usefulness.

### 2.6.1 Market basket analysis

With the advent of cloud computing and easy access to analytics solutions, almost all size of business have started to employ them. This has led to tremendous increase in all businesses using analytics to understand, strategize and engage with their customer base. Market basket analysis is a frequently used term to address this is phenomenon. These techniques are used to achieve both short term as well as long term goals, in terms of increasing profits, customer engagement, understanding the customer base model, streamlining advertising and engagement etc. Ngai et al. (2009) discuss in great detail the various data mining techniques and the classify their application to different problems in customer relationship management. They identify association and clustering as two of the main (along with classification, forecasting, regression, sequence discovery and visualization) functions which solve the core business problems.

Itemset mining is the foundation of identifying useful associations in a any business data. The knowledge of association directly benefits advertising, logistics, transportation and production strategy, and almost all other areas indirectly. Clustering analysis solves the key problem of customer segmentation and customer identification. These two processes are at very important in any long/short term customer engagement strategy and are important drivers of the business.

### 2.6.2 Bioinformatics

The big data computing and storage capabilities have benefited almost all physical sciences research. Among them Bioinformatics has been one of the fields which has seen significant progress owing to it. Topics ranging from gene sequencing to bio-molecule experiments, all have substantial data which follows the transactional format. Since a lot of decisions in bio-informatics research is based on knowledge of repeating patterns or implicit structures present in the data, both itemset mining and clustering techniques are widely applied.

### 2.6.3 Click stream analysis

Click stream analysis is key problem in Web Analytics, where the flow of user clicks is analyzed for each user visit. The dataset of all such click streams provide a clear picture of user interaction issues with the web service. Each stream is modeled as a transaction, with each click an item. The idea of utility is also very practically applicable because of each click having unequal value for the web service. Knowledge of various patterns in this dataset helps them understand, shape and interact with the customer traffic in both a dynamic and long term setting.

# CHAPTER 3. A NOVEL CLUSTERING ALGORITHM TO CAPTURE UTILITY INFORMATION IN TRANSACTIONAL DATA

We develop and design a novel clustering algorithm to capture utility information in transactional data. Transactional data is a special type of categorical data where transactions can be of varying length. A key objective for all categorical data analysis is pattern recognition. Therefore, transactional clustering algorithms focus on capturing the information on high frequency patterns from the data in the clusters. In recent times, utility information for category types in the data has been added to the transactional data model for a more realistic representation of data. As a result, the key information of interest has become high utility patterns instead of high frequency patterns. To the best our knowledge, no existing clustering algorithm for transactional data captures the utility information in the clusters found. Along with our new clustering rationale we also develop corresponding metrics for evaluating quality of clusters found. Experiments on real datasets show that the clusters found by our algorithm successfully capture the high utility patterns in the data. Comparative experiments with other clustering algorithms further illustrate the effectiveness of our algorithm. The work included in this chapter has been published as Lakhawat et al. (2016).

## 3.1 Introduction

Transactional data model is used in many important applications of data mining and analytics like market basket analysis, bioinformatics, click stream analysis etc. Clustering is one of key analysis techniques for these applications. For example, clustering of customer transactions data is performed in market basket analysis for segmentation and identification of various customer types (Ngai et al. (2009)). An important data type in bioinformatics experiments is gene co-expression data which can be modeled as transactional data (Andreopoulos et al. (2009)). Identifying clusters of genes exhibiting similar expression in controlled conditions can lead to critical discoveries in medicine.

A typical transaction of size k in a transactional data set can represented as $T_{ID}$ =($X_1, X_2 \ldots X_K$), where $X_i$ is a category type(also called item type) and ID is the transaction ID. For example, transactions from a retail store data can be like $T_1$ =(Candy, Pop, Chips), $T_2$ =(HDTV, Hi-Definition Speakers) and so on, where each transaction is a list of items bought by a customer in one trip to the store. If we work with only this level of information then we assume equal importance of each category type in the analysis. This would mean that if more number of customer purchase patterns include Pop than HDTVs then our analysis would implicitly assume Pop being a more important category type. However, due to significantly more profit per sale of a HDTV, it is possible that overall HDTV sales yield much more profit to the store than Pop sales. The general idea is that various category types in a transactional data have different level of relative importance (called utility) in the analysis. Therefore to make the transactional data model more realistic utility values are assigned to each category type in the data. An important data mining problem which also uses transactional data model is itemset mining. The aforementioned idea has led to the evolution of traditional frequent itemset mining problem into the current high utility itemset mining problem (Tseng et al. (2015)). Such a transition has not yet occurred for the transactional data clustering problem. To the best of our knowledge no existing clustering algorithm for transactional data captures the utility information.

Current transactional clustering algorithms capture the frequency information (number of occurrences) while finding clusters (Yan et al. (2010)) but do not use the utility values. This will lead to misleading clusters especially when various category types have significant difference in their utility. In order to capture high utility patterns (patterns based on combination of frequency and utility) in the data, we propose a concept of relative utility of category types in clusters. We use it to define a new similarity metric to guide the clustering process. Based on our similarity metric we develop a hierarchical agglomerative clustering algorithm. The algorithm design allows two tunable parameters to be set to achieve clusters with desired properties. Since we are introducing a new clustering rationale, we propose two corresponding cluster evaluation criterion as the existing ones dont take utility values into account.

### 3.1.1 Contributions

Our work makes the following contributions:

- A novel clustering algorithm to capture utility information in transactional data. The algorithm can be tuned to obtain clusters with desired properties of a strong core and a balanced structure.
- Two validation criterion for evaluating the obtained cluster structure based on how accurately it captures the high utility patterns in the data.
- Experimental results on real data sets showing that the clustering algorithm successfully captures the high utility patterns in the data. Comparative experiments with other algorithms illustrate the effectiveness of our algorithm.

## 3.2 Prior work on categorical and transactional clustering

Transactional data is a special type of categorical data where transactions can be of varying lengths. While it is possible to use categorical clustering algorithms for transactional data, most of them lead to a data explosion problem as described in Yan et al. (2010). Specific algorithms targeting transactional data clustering have been developed as well (Wang et al. (1999), Yan et al. (2005), Yan et al. (2010), Yang et al. (2002)). Use of similarity measures and cluster quality measures are two common approaches used in most of the categorical and transactional clustering algorithms. Based on our review of the published work we classify them into the following three categories:

- **Similarity measures:** In Huang (1998) an extension of the popular "k-means" algorithm (which is for numerical data) is presented called "k modes". It replaces the means with modes of clusters to minimize a cost function. An in-depth analysis of subsequent updates of the k-modes algorithm is done in Bai et al. (2013). In Guha et al. (1999) a link based similarity measure is proposed to guide the clustering. An entropy based similarity criterion is presented in Chen and Liu (2005). A transformation technique is presented in Qian et al. (2015) to map categorical data into a space structure followed by using similarity measures. A similarity measure based using information bottleneck theory is presented in Tishby and Slonim (2000) and Andritsos et al. (2004). Ganti et al. (1999), Gibson et al. (1998) and Wang et al. (1999) are some other works with the focus on similarity type measures for clustering. A mapping of categorical data into a tree structure is performed in Chen et al. (2016) before performing clustering using a tree similarity metric.

- **Quality measures:** Yang et al. (2002), Yan et al. (2005), Barbará et al. (2002) and Yan et al. (2010) present algorithms which iteratively improve overall cluster quality. These definitions are based on various entropy based concepts from information theory. An in-depth discussion of entropy based criterion used for the categorical clustering is presented in Li et al. (2004).

- **Miscellaneous:** Categorical clustering has been presented as an optimization problem in Xiong et al. (2012). A learning based multi-objective fuzzy approach is presented in Saha and Maulik (2014). An entropy based subspace clustering technique for categorical data is presented in Sun et al. (2015).

While the current body of work has numerous clustering algorithms, all lack of capturing the utility information in the data. We propose the following algorithm to fill the void.

## 3.3 Clustering algorithm

Any clustering algorithm is designed with a clustering criterion at its core. The clustering criterion behind our algorithm is to gather transactions which share common high utility category types in a cluster. At the convergence of clustering we expect to capture high utility patterns of the data. While the current algorithms for transactional clustering aim to capture high frequency patterns while clustering, we argue that this can lead to inaccurate clustering in cases where the utilities of category types have a steep distribution. A high utility cluster (containing transactions with common high utility category types) can be missed (i.e. not found) if we focus only on frequency information while clustering. Before further describing our clustering algorithm in detail we define the following new concepts in addition to the formal problem setup described Chapter 2 (equations 2.1 - 2.16).

$$CU(C_k) = \sum_{T_j \in C_k} TU(T_j) = \text{Cluster utility of } C_k \tag{3.1}$$

We define cluster utility as an overall measure of importance of a cluster, since it is the sum of utilities of all transactions in it.

$$\forall a_i \in I_{C_k}, ru(a_i, C_k) = \frac{\sum_{a_i \in I_{C_k} \wedge T_j \in C_k} au(a_i, T_j)}{CU(C_k)} \tag{3.2}$$

$$= \text{relative utility of } a_i \text{ in } C_k$$

We define relative utility as the relative importance (since utility is a unit of importance) given to $a_i$ among all $I_{C_k}$ in $C_k$. The set $\{ru(a_i, C_k) | a_i \in I_{C_k}\}$ is then a of signature of the cluster. It is a concise representation of the utility information we are interested in about the cluster. And the collection of all such sets gives us the concise global information of the cluster structure present in the dataset on the basis of high utility patterns in it.

For clusters $C_i$ and $C_j$:

$$affinity(C_i, C_j) = \sum_{a \in I_{C_k} \wedge a \in I_{C_j}} min(ru(a, C_i), ru(a, C_j)) \tag{3.3}$$

We define affinity as the sum of shared utility of common category types among two clusters. So it is a representative of the similar importance given by two clusters to their common category types. This aligns with our goal of clustering which is to gather transactions which share common high utility category types in a cluster.

### 3.3.1 Clustering Algorithm

Using our definition of affinity we perform clustering on the dataset using a bottom-up (also called agglomerative) hierarchical approach. Agglomerative hierarchical approach is well established for successful clustering of categorical data(Guha et al. (1999)). We initialize our algorithm by assigning each transaction as an individual cluster. We then calculate affinity of each cluster with every other cluster. Following that we incrementally merge the two clusters which have most affinity to each other. A merge implies union of the two clusters to form a single cluster. After each merge we calculate all the affinity values associated with the new cluster. This includes first computing the cluster utility and relative utilities for the new cluster.

The cluster structure represents a complete graph with each cluster as a node connected to every other node. The node weights represent the cluster utility and the edge weights represent the affinity values. At each merge we calculate the node weight for the new node and the edge weights for the new edges while keeping the graph complete. We terminate the algorithm when the maximum affinity found between any two clusters is below a certain predefined affinity threshold $min_{aff}$. The cluster distribution can have a long tail, meaning there can be many transactions which do not fall under any significant cluster structure. $min_{aff}$ ensures that clusters with high cluster utility do not end up merging just because all other remaining small clusters are very dissimilar among each other.

At this point we only select clusters with more than or equal to a minimum cluster utility value. We do this by predefining a parameter $min_{uty}$ and accepting only those clusters which have cluster utility more than or equal to ${min_{uty}}^{th}$ of the cluster with the maximum cluster utility. Figure 3.1 (extracted from Lakhawat et al. (2016)) shows a small synthetic example illustrating the clustering process. It shows affinities and cluster utilities as edge and node weights respectively. The change in cluster structure can be observed based on the choice of $min_{aff}$ and the $min_{uty}$ parameters. Algorithm 1 below (extracted from Lakhawat et al. (2016)) provides a pseudocode.

## 3.4 Properties of the cluster structure

The obtained cluster structure by our algorithm will have the following properties by design:

- A strong core: We terminate the clustering process when the maximum affinity found between any two clusters is less than $min_{aff}$. Therefore, each cluster will have a set of category types

```
Input: C ;
while $max_{aff} \geq min_{aff}$ do
    for $C_i, C_j \in C$ do
        if $affinity(C_i, C_j) > max_{aff}$ then
            $max_{aff}$ = affinity($C_i, C_j$);
            $C_{m1} = C_i$;
            $C_{m2} = C_j$;
    merge($C_{m1}, C_{m2}$);
    update relevant affinities;
for $C_t \in C$ do
    if $\frac{CU(C_t)}{max(CU(C_k) \forall C_k \in C)} \leq min_{uty}$ then
        delete $C_t$;
return C;
```

**Algorithm 1:** A novel clustering algorithm for categorical data with utility information

for which each transaction in the cluster has at least $min_{aff}$ of shared relative utility among them. This is evident from the definition of affinity.

- A balanced structure: Ratio of cluster utility of each cluster to that of the cluster with the maximum cluster utility will be more than or equal to the predefined threshold $min_{uty}$.

- Greedy path taken: Clusters with the most affinity with each other merge at each step.

Besides the above three properties, we expect the cluster structure to capture the high utility patterns in the data. This is the primary goal of this work. A recent transactional clustering framework SCALE(Yan et al. (2010)) introduces a cluster validation criterion called LISR(Large Item Size Ratio) which captures large items(frequently occurring category types) preserved in clustering. Since we capture patterns based on combination criterion of frequency and utility, using validation criterion like LISR is inadequate. Therefore we introduce two new validation criterion based on the high utility itemsets (HUI) present in the clusters. We call them $HUI_{capture}$ and $HUI_{inaccurate}$. They are defined as follows:

$$\text{HUI in C} = \{X \text{ such that } au(X, C_k) >= \phi' | C_k \in C\} \tag{3.4}$$

$$\text{where } \phi' = \frac{\frac{\phi}{|D|}}{\sum_{C_k \in C} |C_k|}.$$

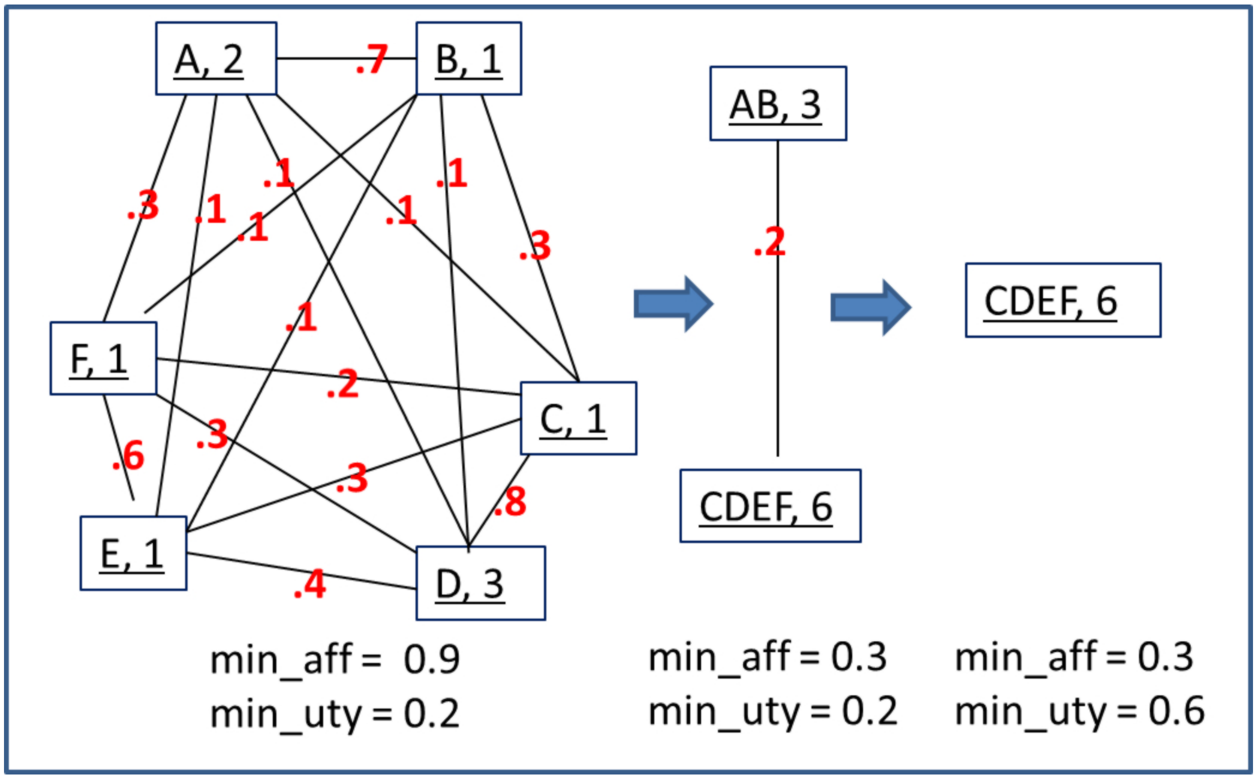


Figure 3.1 Illustration of the clustering process

The set of HUI in C represents the high utility patterns present in the cluster structure C. If the entire data set $D$ is used in clustering then $\phi = \phi'$. However, since clustering is an expensive operation often random sampling techniques are used to select transactions for clustering. In those cases we scale down the threshold $\phi$ to $\phi'$.

$$HUI_{capture} = \frac{|\{\text{HUI in C}\} \cap \{\text{HUI in D}\}|}{|\{\text{HUI in D}\}|} \tag{3.5}$$

$HUI_{capture}$ represents the degree to which the cluster structure $C$ captures the high utility patterns present in the dataset $D$. Successful $HUI_{capture}$ is characterized by a value greater than the ratio of size of data used for clustering to the size of the whole dataset $\frac{\sum_{C_k \in C} |C_k|}{|D|}$. Higher values of $HUI_{capture}$ imply higher quality of the cluster structure.

$$HUI_{inaccurate} = \frac{|\{\text{HUI in C}\} - \{\text{HUI in C} \cap \text{HUI in D}\}|}{|\{\text{HUI in D}\}|} \tag{3.6}$$

$HUI_{inaccurate}$ represents the degree of inaccurately captured high utility patterns in C with respect to the high utility patterns present in the data $D$. Lower values of $HUI_{inaccurate}$ imply higher quality of the cluster structure.

## 3.5 Experimental evaluation and discussion

To validate our algorithm we perform clustering on a real data set obtained from RetailDataset (2016) and provided by Brijs et al. (1999a). It contains anonymous customer transaction data from a Belgian retail store and contains 88,163 transactions. We randomly generated the external utilities (between 1-50) for various category types by using a uniform random number generator. Using this data we performed the following experiment:

- For $\phi$=50000 we calculated the list of high utility itemsets (HUI) in the complete data set $D$. We found 111 HUIs. We did this by implementing a popular itemset mining technique called the two-phase algorithm (Liu et al. (2005b)). The choice of $\phi$ was based on obtaining a reasonably large but manageable number of HUI.
- For four combinations of our algorithm parameters we calculated the cluster structure and the list of HUI for each cluster structures. The selection of data for clustering is done using uniform random sampling.
- Finally, we evaluate our validation metrics: $HUI_{capture}$ and $HUI_{inaccurate}$. We present the results in Figure 3.2 (extracted from Lakhawat et al. (2016)).

The following inferences can be drawn from the results:

- The HUI captured in clusters are significantly greater than the data used for clustering. This implies high quality of the cluster structure.
- The extremely low (and 0) values of HUI inaccurately captured imply high accuracy of the cluster structure.

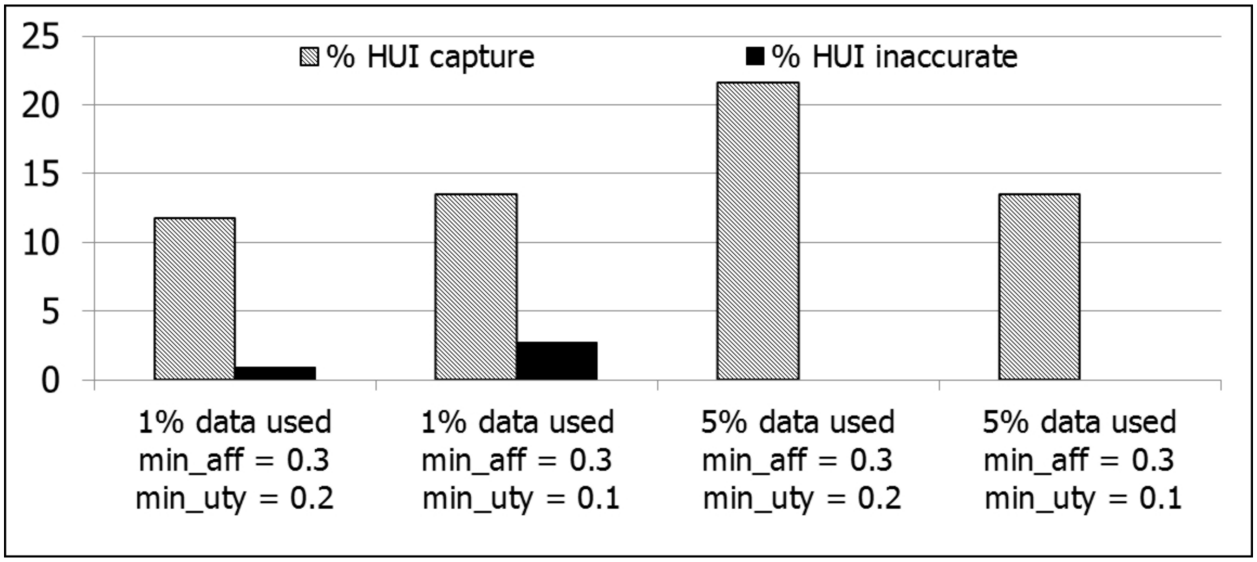


Figure 3.2  Cluster validations metrics for the retail store data set

We also perform comparative experiments with two other categorical clustering algorithms: K-modes (Huang (1998)) and BKPlot (Chen and Liu (2005)). Since the provided implementations of K-modes (Kmo (2015)) and BKPlot (BKP (2008)) accept data sets only with uniform transaction length, we perform the comparative analysis on the Soybean data set obtained from UCI (1987). It is a collection of attributes of various soybean plants type. Soybean data set is reasonably small so we dont use the random sampling and instead perform clustering on the entire dataset. To maintain uniformity in cluster structure we evaluate a four cluster solution for each algorithm. For using in our algorithm we also generate utility values(between 1-50) for each category type using a uniform random number generator. Since the data set is small and we dont use random sampling, we perform comparison based on patterns (high utility itemsets) missed by clusters formed by each algorithm. Table 3.1 (extracted from Lakhawat et al. (2016)) shows the comparative results. $\phi$ represents the threshold value used to calculate high utility patterns (itemsets). Our algorithm performs better (or equal) than the other two for all cases. While the soybean data set is a small one, it still illustrates the fact that well established algorithms like K-modes and BKPlot miss out on high utility patterns.

Table 3.1 Comparative results for Soybean dataset: patterns missed

| $\phi$ | **Our Algorithm** | **BKPlot** | **K-modes** |
|---|---|---|---|
| 800 | 0 | 0 | 8 |
| 400 | 2 | 5 | 7 |
| 200 | 0 | 2 | 4 |

## 3.6 Conclusion

In the growing body of work of clustering algorithms for transactional data we identified the need for an algorithm which captures the utility information in the transactional data. The current algorithms for transactional clustering aim to capture high frequency patterns while clustering. We argue that this can result in misleading clusters especially for cases where the utilities of category types have a steep distribution. High utility clusters can be missed if we focus only on frequency information while clustering. We propose a novel clustering algorithm for transactional data which captures the utility information. We also propose two validation criterion for the obtained cluster structure based on how accurately it captures the high utility patterns in the data.

Our experiments on a real data sets show that the clustering algorithm successfully captures the high utility patterns in the data. Our comparative experiment results further illustrate the effectiveness of our algorithm over the popular K-modes and BKPlot algorithms. For future work, we plan to do experiments on data sets from various applications like bioinformatics, click stream data etc. We believe interpretations of clusters found in different data sets will lead to interesting results and evolution of our algorithm. We plan to develop the idea of utility aware clustering for entropy based clustering methods as well.

# CHAPTER 4. A CLUSTERING BASED PREDICTION SCHEME FOR HIGH UTILITY ITEMSETS

We strongly believe that the current Utility Itemset Mining (UIM) problem model can be extended with a key modeling capability of predicting future itemsets based on prior knowledge of clusters in the dataset. Information in transactions fairly representative of a cluster type is more a characteristic of the cluster type than the the entire data. Subjecting such transactions to the common threshold in the UIM problem leads to information loss. We identify that an implicit use of the cluster structure of data in the UIM problem model will address this limitation. We achieve this by introducing a new clustering based utility in the definition of the UIM problem model and modifying the definitions of absolute utilities based on it. This enhances the UIM model by including a predictive aspect to it, thereby enabling the cluster specific patterns to emerge while still mining the inter-cluster patterns. By performing experiments on two real data sets we are able to verify that our proposed predictive UIM problem model extracts more useful information than the current UIM model with high accuracy. This work was published as Lakhawat et al. (2017).

## 4.1 Introduction

Itemset mining is an important problem in data mining. The key objective in itemset mining is to identify the frequently occurring patterns of interest in a collection of data objects. Itemset mining is among the areas of data mining which have received high interest in the last decade (Liao et al. (2012)). There are two primary reasons for these developments. First, there is a primary need to extract highly repetitive patterns from data in many data mining applications. Second, data mining problems from various domains can be easily modelled as an itemset mining problem. As a result, various application areas like market basket analysis (Ngai et al. (2009)), bioinformatics

(Alves et al. (2009); Naulaerts et al. (2015)), website click stream analysis (Ahmed et al. (2009); Li et al. (2008)) etc. have witnessed significant use of itemset mining techniques.

The first model (Agrawal et al. (1994)) of itemset mining problem was based on identifying patterns solely on their occurrence frequency. However, a subsequent model emerged ( Chan et al. (2003); Liu et al. (2005a); Tseng et al. (2010, 2015)) in which utility values were assigned to the data elements based on their relative importance in the analysis. The pattern identification criterion in this new model is a combination of occurrence frequency and utility value. In this work, we enhance the effectiveness of the Utility Itemset Mining model by adding a prediction aspect to it. Having reasonably accurate knowledge of possible future itemsets is of immense value in all applications of Utility Itemset Mining where data is scarce or dynamic in nature and where discovery of knowledge sooner and with lesser amount of data adds much more value to them.The key intuition for this work arises from the existence and knowledge of clusters present in the data. In this work, we show that prior knowledge of the clusters present in the data has high potential to guide the future itemsets discovery.

Building on this idea we propose a prediction scheme for high utility itemsets which captures frequency, utility and cluster structure information to predict the possible future itemsets with high accuracy. Experiments shows that we are able predict a good number of future itemsets with high accuracy over the baseline scheme. While Utility Itemset Mining is not a machine learning problem, but if it were then our contribution would be analogous to the Bayesian version of this problem with the cluster structure acting as the Prior.

We presented illustrative examples and formal models for the FIM and UIM probelms in Chapter 2. We strongly believe that the UIM problem model can be further extended to add a prediction aspect to it. Let us consider the example in Figure 2.2. Suppose we have reasonable confidence that the customer for transaction T4 is a college student. Then the information present in T4 is more representative of a customer class of college students than the entire customer population. Leveraging this knowledge can help us predict a latent behavior of college students if present in the data. While ignoring this knowledge leads to information loss due to generalization. This motivated us to investigate ways for leveraging the knowledge of clusters present in data in current UIM model.

### 4.1.1 Motivation for a prediction enabled UIM model

Datasets which can be modeled as transactional data have frequently occurring (repeating) patterns of interest in them. This is the key information which itemset mining techniques strive to extract from these datasets. For example, in retail transactions datasets this information means items which are frequently bought together by customers. However, on the same transactional datasets clustering analysis is performed to study the cluster structure of these datasets. For retail transactions data sets this is the basis of the customer segmentation analysis (Ngai et al. (2009)), where similar sets of transactions are clustered together to identify and study various customer types present in the data.

Clustering of transactional type datasets is performed in various biomedical applications as well. Gene expression data is one such example data type which is analyzed using both itemset mining (Alves et al. (2009); Naulaerts et al. (2015)) and clustering techniques (Andreopoulos et al. (2009)). This implies that itemset mining and clustering study different aspects of the same data set. *While itemset mining abstracts the dataset in form of itemsets, clustering abstracts it in form of clusters of transactions.* Figure 4.1 (extracted from Lakhawat et al. (2017)) presents an illustration of the above idea. If we imagine the dataset to be a solid cylinder, then a top/plan view (corresponding to itemset mining) will show a circle (correspondingly itemsets). While a side/elevation view (corresponding to clustering) will show a rectangle (correspondingly clusters).

Performing clustering or itemset mining analysis while ignoring the other creates a handicap as we do not use all available information fully. Recent transactional data clustering techniques are starting to adapt to this fact. For example, a recent transactional clustering algorithm proposed in Yan et al. (2010) introduces the idea of weighted coverage density. Coverage density is a metric of cluster quality which is used to guide clustering algorithms. Recognizing the fact that frequently occurring patterns are a key characteristic of transactional data, the authors in Yan et al. (2010)

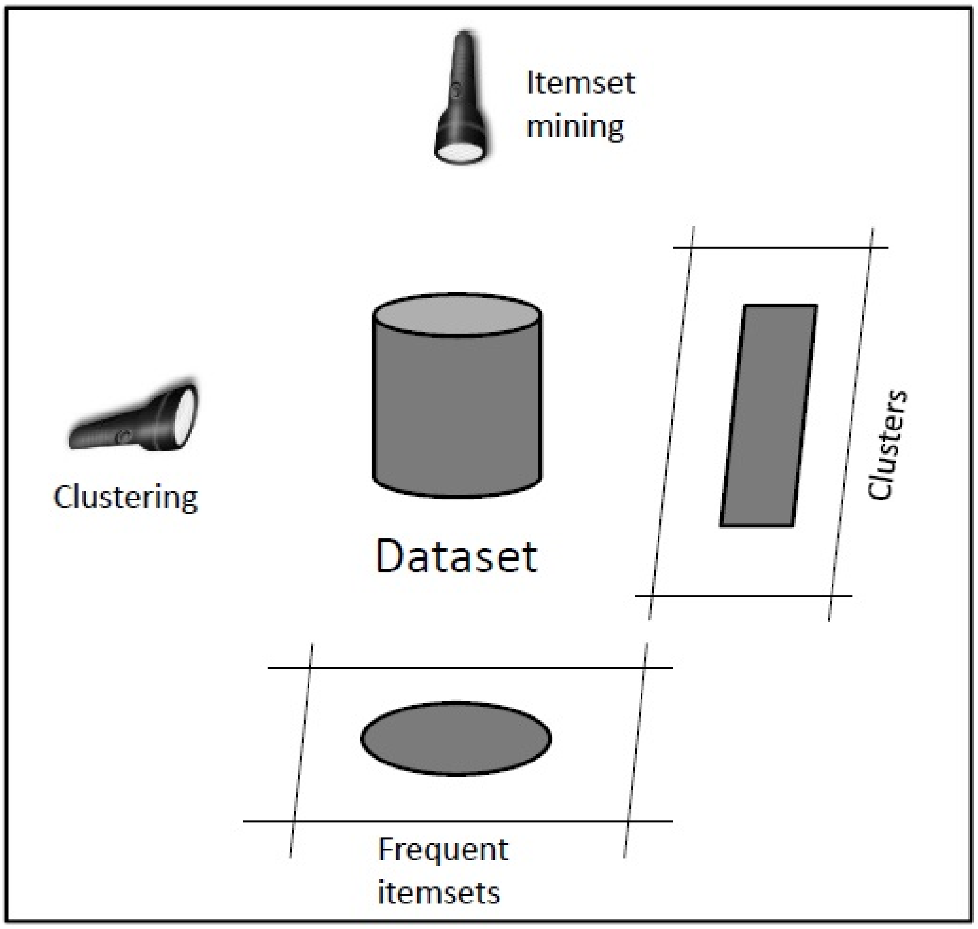


Figure 4.1 Illustration of different abstractions of dataset

assign weights to items in the coverage density function based on their occurrence frequency. This leads to clusters which are more practically useful. There are two issues if we consider the current UIM problem model and the clustering problem:

1. If we divide the entire set of transactions into clusters and perform itemset mining in each cluster separately, we might miss an inter-cluster pattern.

2. If we perform itemset mining in the whole dataset disregarding clustering, a pattern highly specific to a cluster might be missed due to no support from any other cluster.

This directs to us that we need to somehow implicitly use knowledge the cluster structure while performing itemset mining.

### 4.1.2 Need to implicitly use the Cluster Structure

An implicit use of cluster structure of data in itemset mining can potentially address these issues. The knowledge of cluster structure can help identify transactions which are highly representative of a cluster type. The cluster types usually represent some real world entity (for example type of customer). The information in these special transactions is more characteristic of their cluster type than the entire data. Therefore subjecting these transactions to the common threshold in the UIM problem is not appropriate. To overcome this problem, we conclude that some extra importance must be provided to these special transactions. We do this by introducing a new clustering based utility in the definition of the UIM problem model. The modified UIM problem model enables the cluster specific patterns to emerge while still mining the inter-cluster patterns. In essence, we develop a mechanism to enhance the importance (utility) of certain transactions which translates into inflation in utility of certain itemsets. Those itemsets which are enough inflated to cross the threshold will constitute the predictions. This modification in the model can integrate into all UIM techniques as it does not affect the itemset mining part of the techniques.

Revisiting the example in Figure 2.2, the new predictive UIM model gives extra importance/utility to the items in transaction T4 by identifying it as a special transaction (representative of a college student). Let us assume that the Music CD bought by this college student is of a current hit album. Then the pattern of this Music CD bought along with typical college student items is likely to repeat. This will lead to eventual discovery of this Music CD as high utility item. The predictive UIM model will facilitate a sooner (using less data) discovery of such items.

In rest of the paper, we first discuss the key works done on the itemset mining problem. Then we formally describe the itemset mining problem followed by the definition of our new clustering based utility to extend the UIM model. We then have a discussion on the use clustering algorithm followed by the experiments on to real data sets before we conclude.

## 4.2 Related Work

The problem of itemset mining was first introduced by Agrawal et al in Agrawal et al. (1994) as frequent itemset mining in context of market basket analysis. They introduced the idea of a downward closure property for generating the potential (candidate) frequent itemsets of size k using the already discovered frequent itemsets of size k-1. This is also popularly known as the apriori technique. This helped to substantially reduce the search space for the frequent itemsets. Building up on this idea many subsequent works extended it by introducing sampling techniques (Toivonen et al. (1996)), dynamic itemset counting (Brin et al. (1997)), parallel implementations (Agrawal and Shafer (1996)) etc.

A limitation of "apriori" logic based techniques is that sometimes they can generate a large number of candidate itemsets. Since each candidate itemset requires a scan over the entire dataset it also slows the mining process significantly. A popular technique to overcome this issue has been proposed in Han et al. (2000) called FP-Growth. It performs itemset mining by generating a tree structure rather than candidate generation. There are also techniques proposed which mine the dataset in vertical format (that is list items with sets of transactions) rather than the traditional horizontal format (list of transactions with items). One such work is propose by Zaki in Zaki (2000).

Frequent itemset mining lacked important modeling capabilities like relative importance of various items (called utility) and the frequency of an item in a particular transaction, leading to the emergence of utility itemset mining in (Chan et al. (2003); Liu et al. (2005a); Tseng et al. (2010, 2015)) among others, where itemsets are mined on the basis of utility support in the dataset rather than frequency support. This makes the problem model more realistic and of higher practical value.

The downward closure property for candidate generation does not apply directly for utility mining. This led to the idea of a transaction weighted utility, which enabled the apriori type candidate generation again. This was the basis of the initial work done in utility mining with subsequent techniques proposed on various strategies for pruning the search space.

The problem of candidate set explosion is also present in these works due to the use of "apriori" logic. To counter this Tseng et al. (2010) proposes a tree based model called UP-Growth for Utility mining which traverses the dataset only twice.

Recently in Tseng et al. (2015) authors proposed Utility mining algorithms which use a closed set representation for itemsets which is very concise and yet shows competing performance.

## 4.3 A novel cluster based utility to enhance the UIM model

We discussed in the first section that the goal is to extend the current UIM problem model to add prediction capability to it by implicitly using the cluster structure of data in itemset mining. *Certain transactions are more representative of a cluster type over others.* The information in these special transactions is more characteristic of their cluster type than the entire data. Therefore we do not wish to subject these transactions to the common threshold in the UIM problem. To overcome this problem, we develop a mechanism to attach extra utility to these transactions. We do this by introducing a new clustering based utility in the definition of the UIM problem model. This addition translates into predicting capability of the UIM model.

NOTE: We urge the reader to briefly review the formal definition of the UIM model, as the following proposed model builds on top of that.

We define a new utility by calling it cluster utility of a transaction (and the items in it). This is a transaction specific utility for items and is same for all items in a transaction. We reiterate following two two clustering concepts along the UIM model before we define the cluster utility.

$C$ as the set of all given clusters. Each cluster is defined as: $C_j = \{T_1, T_2, \ldots\}$. Cluster $C_j$ is a subset of transactions from $D$.

We introduce an affinity metric which represents the degree of similarity between a cluster $C_j$ and a transaction $T_i$.

$$affinity(T_i, C_j) = \text{similarity b/w} T_i \text{ and } C_j \tag{4.1}$$

These additions to the UIM problem model assume that a fairly accurate cluster structure is given and an appropriate affinity metric is provided. The accuracy here defines an attribute that a cluster structure which portrays the characteristics (repetitive patterns) of interest in the dataset. By appropriateness of the affinity metric we mean a metric which captures the type of similarity (based on constituent items) between a cluster and a transaction that is of interest in the analysis. These assumptions are fairly reasonable as there is a large body of work directed towards of categorical (transactional) clustering. These clustering techniques define subsets of transactions as clusters in the same way as we define them in our predictive UIM problem model. Use of some version of a similarity metric is common for these techniques (Huang (1998); Guha et al. (1999); Chen and Liu (2005)). The affinity metrics used in them can be used in our extended UIM problem model by interpreting a transaction as single element cluster.

$$cu(a, T_i) = 1 + k * max\{affinity(T_i, C_j) \forall C_j \in C\} \tag{4.2}$$

In above equation, $k$ is a tunable parameter and decides how aggressively the cluster information is used in the predictive UIM. Note that the cluster utility is same for all items in a transaction. The rationale behind this definition is to decide the cluster utility of a transaction based on the cluster which is most similar to it.

We integrate this new internal utility in the calculation of the absolute utilities. The new definition of absolute utility of an item $a$ in a transaction $T_i$ is given by the following:

$$au(a, T_i) = eu(a) * iu(a, T_i) * cu(a, T_i) \tag{4.3}$$

This implicitly changes the definitions of $au(X, T_i)$, $TU(T_i)$, $TWU(X, D)$, Set of HTWUI, $au(X, D)$ and the set of HUI. All techniques for UIM use the absolute utilities as the building blocks to search for high utility itemsets (Chan et al. (2003); Liu et al. (2005a); Tseng et al. (2010, 2015)), so this enhanced predictive UIM problem model will integrate into all of them.

### 4.3.1 Impacts of the enhanced predictive UIM problem model

The following are the impacts of making the above updates to the current UIM model.

1. Assuming that the affinity function to have range [0, 1]. The cluster utility of any item will fall in range [1, 1+k]. Cluster utility closer to 1 will imply their respective transaction to be almost non-representative of any given cluster type. Higher values will imply more similarity of their respective transaction with some given cluster.

2. Since the new definition of absolute utility of an item in a transaction is the product of cluster utility, internal utility and external utility, all absolute utilities will either increase or remain same in the new predictive model.

3. For the same threshold $\Phi$, the predictve model will always find equal or more number of HUI than the current model. Also the set of HUI found by the current model will always a subset of the HUI found by the predictive model.

4. Higher values of parameter $k$ will aggressively use the cluster information and therefore produce more number of HUI. This is recommended when additional emphasis on cluster specific patterns is required.

5. The additional (predicted) itemsets found should be interpreted in the following two ways.

   - When more data arrives later, the additional itemsets found by the model at a previous time are likely to be found in the list of HUI of the current model at that time. The interpretation of this is that a certain pattern(s) are present in particular cluster(s), but with the given amount of data they do not have enough utility support to appear in the list of HUI of the current UIM model. However, with the numbers accumulating with time they will soon show up in the list of HUI in the future. The predictive UIM model recognizes them and helps them getting discovered sooner (with fewer data).

   - If the data is static (or no new data will be available at a later point in time), the additional HUI found in the predictive model are the ones which missed out in the list of HUI of current model due to being specific to only one (or very few) cluster(s) present among many and hence could not gather enough numbers to cross the threshold. However such additional HUI can have application specific importance. For example, a purchase pattern for a specific customer type can be used to create targeted advertisements for those customers.

6. Making this addition modifies the definition of various absolute utilities. However, the use of absolute utilities to find the set of HUI remains the same. Therefore this new model has to ability to be able to be integrated into all UIM techniques.

7. Each cluster in the cluster structure of the data usually represents some real world entity. This has the following implications.

    - Once a satisfactory cluster structure is obtained it can be reused for same type of data. This is because the purpose of cluster structure is only to identify if a particular transaction is fairly representative of a cluster type. This means that the computational expense of clustering need not be repeated every time.
    - The entire dataset might not be needed to obtain an accurate cluster structure. If the size of the dataset is much bigger compared to the cluster structure present in it, then a randomly sampled fraction of dataset is sufficient to capture the cluster structure.

8. The predictive model always finds equal or more HUI than the current model, it can potentially extract the complete set of HUI based on the current model while using fewer data. It can also find additional useful HUIs which the current model missed. This translates into earlier access to actionable information and access to additional useful information.

## 4.4 Choice of clustering technique

Since the proposed predictive UIM problem model assumes the knowledge of an accurate cluster structure and an appropriate similarity metric as discussed in the previous section, it is important to choose a suitable clustering technique. There is a large body of work directed towards clustering of categorical and transactional data. The clustering techniques return the clusters in form of sets of transactions with similar transactions in each set. A majority of these techniques Huang (1998); Guha et al. (1999); Chen and Liu (2005) employ some similarity metric between the clusters to guide the clustering process using divisive, agglomerative or repartitioning algorithms. The same affinity metrics can be used in the enhanced UIM model by interpreting a transaction as single element cluster. The choice of clustering technique used can be subjective based on the preferences and requirements of the application domain.

Review suggests that certain categorical (transactional) clustering algorithms perform clustering on the basis of frequently occurring patterns in the transactions. Such schemes may be applicable when the external utility information is not very important. However in most real world applications, various item types have different relative importance in the analysis. This is the reason for emergence of UIM as an evolved version of FIM. A better suited clustering technique for use in this enhanced UIM problem model should be based on high utility patterns in the data rather than high frequency ones. We have developed a clustering technique which successfully captures the high utility patterns in the data in Chapter 3 Lakhawat et al. (2016). This clustering technique is our most preferred choice and applied here. In the next section we perform experiments on two real datasets to evaluate results of the predictive UIM problem model.

## 4.5 Experiments on real datasets

We perform an analysis of the results from the predictive UIM problem model proposed here. We use two real datasets called BMSWebView1 (obtained from BMSWebView1 (2016)) and Retail dataset (provided by Brijs et al. (1999b) and obtained from RetailDataset (2016)). BMSWebView1 is a real life dataset of website clickstream data with 59,601 transactions in it. Retail dataset contains 88,163 anonymized transactions from a Belgian retail store. We randomly generated the external utilities (between 1-50) for various item types in both the datasets by using a uniform random number generator. It is common to generate utility values when evaluating algorithms for UIM (Tseng et al. (2015)). To obtain the cluster structure to be used for the predictive UIM problem model, we use the utility based categorical clustering algorithm discussed earlier and in the Appendix. For finding the high utility itemsets (HUIs) we implemented a popular UIM technique called the two-phase method (Liu et al. (2005a)). It essentially finds all the potential HUI using the transaction weighted downward closure property we discussed in an earlier section and then scans the dataset to determine the actual HUIs.

### 4.5.1 Experimental Design

We created the following experimental design to compare the effectiveness of our predictive UIM problem model with the current UIM problem model:

1. We create the following 4 versions of both the data sets:

   - Containing first 25% of the data.
   - Containing first 50% of the data.
   - Containing first 75% of the data.
   - Containing the complete data.

   We interpret the complete dataset as all the information which future holds. The purpose of this step is to create scenario where as more data arrives with time it leads to more itemsets being discovered.

2. For each of these datasets we find the set of HUI using the current UIM model. For the retail dataset we use $\Phi$ = 50,000 and for the BMSWebView1 data set we use $\Phi$ = 20,000. The choice of these threshold values is based on discovering a manageable number of HUI. Higher values of $\Phi$ lead to fewer HUI and vice versa. This step establishes the checkpoints for the itemsets discovered by the current UIM model for each version of both the datasets.

3. We generate two cluster structures for both the Retail dataset and the BMSWEbView1 dataset by using 1% and 5% of uniformly randomly sampled data using our clustering algorithm as described before. This step results in a total of 4 cluster structures which will be

used to model the predictive UIM problem for each version of the two datasets. The purpose of selecting two different fractions of datasets in clustering is to observe their effect in the discovery of itemsets.

4. Next we assign the cluster utility to each transaction and their constituent items based on the chosen cluster structure. We do this assignment in a conservative, moderate or aggressive manner based on the following criterion:

$$\text{conservative } k = \{^{0 \text{ if affinity}(T_i, C_j) < 0.25}_{1 \text{ otherwise}} \tag{4.4}$$

$$\text{moderate } k = 1 \tag{4.5}$$

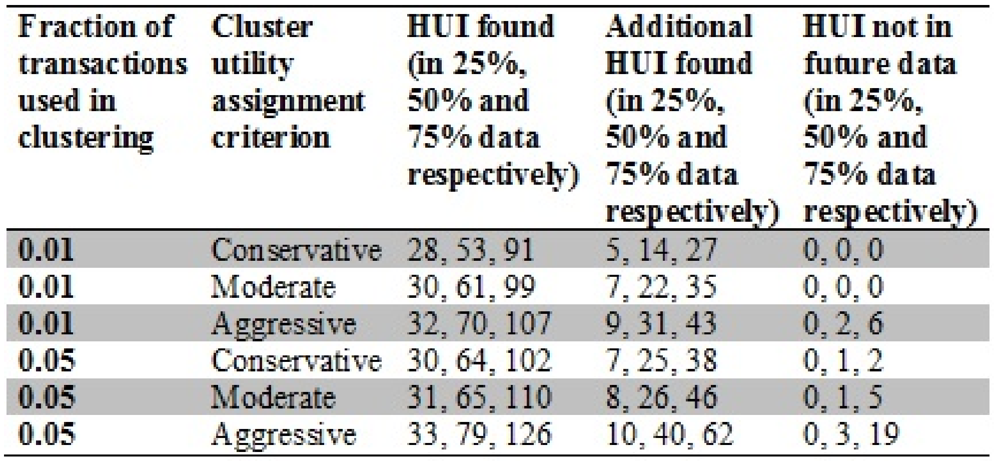
$$\text{aggressive } k = \{^{1 \text{ if affinity}(T_i, C_j) < 0.5}_{2 \text{ otherwise}} \tag{4.6}$$

5. After assigning the cluster utility we calculate the new values for all absolute utilities. We then find out the set of HUI for each of the above cases based on our predictive UIM problem model (for their respective  values) and compare them with the ones found when using the current UIM problem model on the same version of dataset. The key information pieces of interest are:

   - **HUI found:** This is the number of HUI found by the predictive UIM model for each version of both datasets for the two cluster structures. This will always be equal to or more than the number HUI found using the current UIM problem model.
   - **Additional HUI found:** This is the additional number of HUIs found by the predictive UIM problem model over the current UIM problem model. This is the most important information of interest. This represents additional itemsets the new model was able to extract using the knowledge of cluster structure of the dataset.
   - **HUI not in future data**: This is the number of HUI found by the predictive UIM problem model which are not present in the list of HUI for the current UIM model when using the complete dataset. The HUI in this category represent patterns which are very

cluster specific and could not find enough support from the complete data set to cross the threshold . While these itemsets cannot be called high utility itemsets (HUI) in the conventional definition, they do have high utility with respect to their cluster type and they might be very close to crossing the threshold for the current UIM problem model as well. This attribute of these itemset makes a useful set of information.

These results from the above experiment are presented in Figure 4.2 and Figure 4.3 (extracted from Lakhawat et al. (2017)). We also performed this experiment for a much wider range of parameters and discovered similar insights. The results from those experiments are presented in the APPENDIX at the end of the thesis.



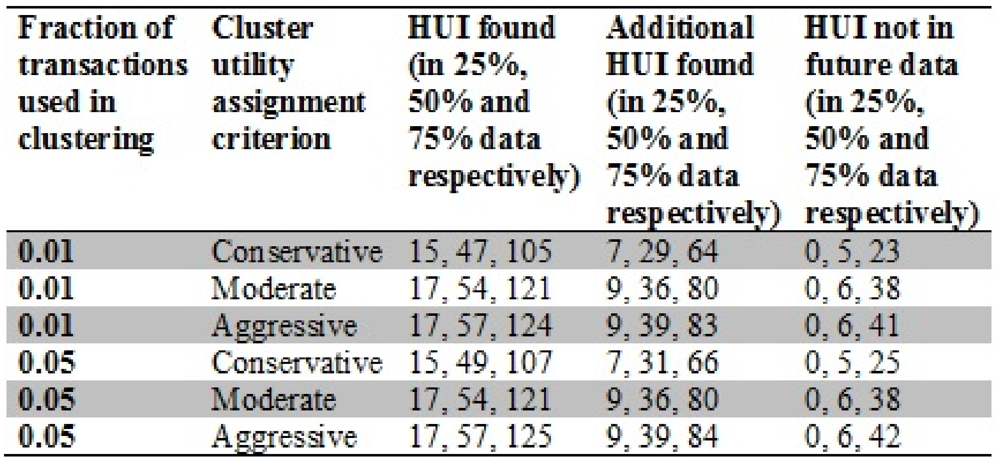
| Fraction of transactions used in clustering | Cluster utility assignment criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | HUI not in future data (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|
| 0.01 | Conservative | 28, 53, 91 | 5, 14, 27 | 0, 0, 0 |
| 0.01 | Moderate | 30, 61, 99 | 7, 22, 35 | 0, 0, 0 |
| 0.01 | Aggressive | 32, 70, 107 | 9, 31, 43 | 0, 2, 6 |
| 0.05 | Conservative | 30, 64, 102 | 7, 25, 38 | 0, 1, 2 |
| 0.05 | Moderate | 31, 65, 110 | 8, 26, 46 | 0, 1, 5 |
| 0.05 | Aggressive | 33, 79, 126 | 10, 40, 62 | 0, 3, 19 |

Figure 4.2 Experiment results: Retail dataset

### 4.5.2 Key Inferences from the Experimental Results

The following inferences are drawn from the obtained results.

1. Increasing the fraction of transactions used in clustering results in increase of number of HUI found and additional HUI found. This is expected, as with more transactions being used in clustering the cluster structure found is expected to be closer to the true cluster structure of the dataset. This results in more transactions finding higher affinity values with



| Fraction of transactions used in clustering | Cluster utility assignment criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | HUI not in future data (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|
| 0.01 | Conservative | 15, 47, 105 | 7, 29, 64 | 0, 5, 23 |
| 0.01 | Moderate | 17, 54, 121 | 9, 36, 80 | 0, 6, 38 |
| 0.01 | Aggressive | 17, 57, 124 | 9, 39, 83 | 0, 6, 41 |
| 0.05 | Conservative | 15, 49, 107 | 7, 31, 66 | 0, 5, 25 |
| 0.05 | Moderate | 17, 54, 121 | 9, 36, 80 | 0, 6, 38 |
| 0.05 | Aggressive | 17, 57, 125 | 9, 39, 84 | 0, 6, 42 |

Figure 4.3 Experiment results: BMSWebView1 dataset

their respective clusters. Higher affinities imply higher cluster based utilities, which further implies higher absolute utilities for itemsets. Higher absolute utilities mean more itemsets are likely to cross the threshold Φ.

Figure 4.4 to Figure 4.7 (extracted from Lakhawat et al. (2017)) show the graphical illustrations. The Y-axis shows the HUI found in Figure 4.4 and Figure 4.5. Additional HUI found are shown on the Y-axis in Figure 4.6 and Figure 4.7. Four different predictive UIM problem models are shown in these figures based on two cluster structures and two cluster utility assignment criterion. The X-axis for these figures shows the dataset version used. Figure 4.4 and Figure 4.5 also shows the HUI found when using the current UIM problem model.

2. Varying the cluster utility criterion from conservative to moderate to aggressive results in increase in the number of HUI found and additional HUI found. This is expected, as this stepped variation results in increase of cluster utility for the transactions. Increase in cluster utility results in increase of absolute utility for itemsets at each step. Increase in absolute utility for itemsets means more itemsets are likely to cross the threshold . A graphical illustration is shown in Figure 4.4. There are few HUI found (for the predictive model) which are not present in the list of HUI for the complete data (when using the current model) for cases of aggressive cluster utility assignment and especially when using 75% of data. This should

interpreted in the correct perspective. Aggressive cluster utility assignment should be used when the analysis is especially focused on discovering all possible cluster specific patterns along with the global patterns. As the current UIM problem model completely disregards the cluster structure, comparison with it in this case becomes less relevant. Furthermore, when we use the 75% version of the data with the predictive UIM problem model, the complete data set is inadequate to verify the validity of the additional HUI discovered and more data might be needed to do so.

3. The predictive UIM problem model extracts significantly more (30% to 50% more for most cases in our experiments when being conservative or moderate in cluster utility assignment) actionable information (HUI) from the data compared to the current UIM problem model. While most of additional HUI found by the new model are found by the current model when additional data is available, few which are not found, are also useful itemsets. These itemsets represent patterns which are specific to cluster types and were not discovered by the current model due to the information loss problem discussed previously. Overall the predictive UIM model leverages the knowledge of the cluster structure while mining for itemsets based on utility and frequency for improved information extraction.

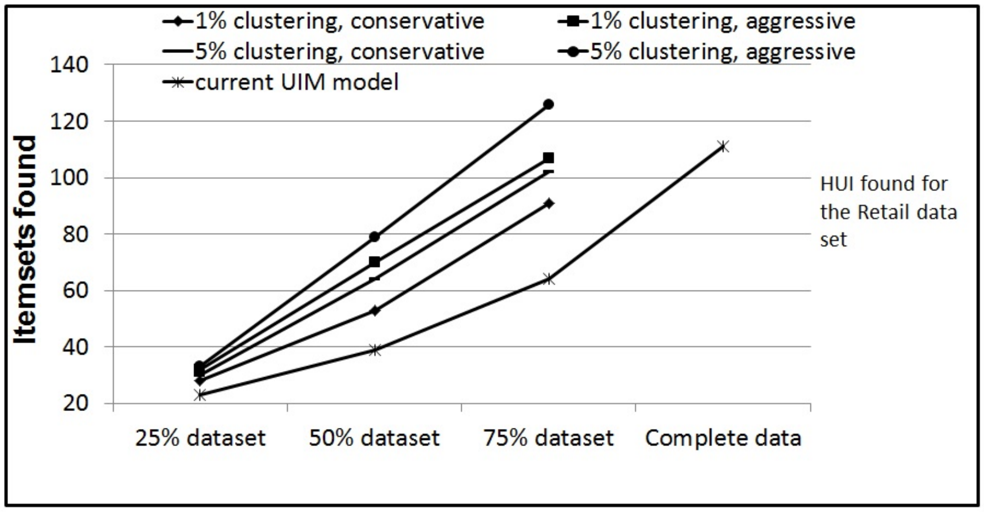


Figure 4.4 HUI found for the Retail dataset

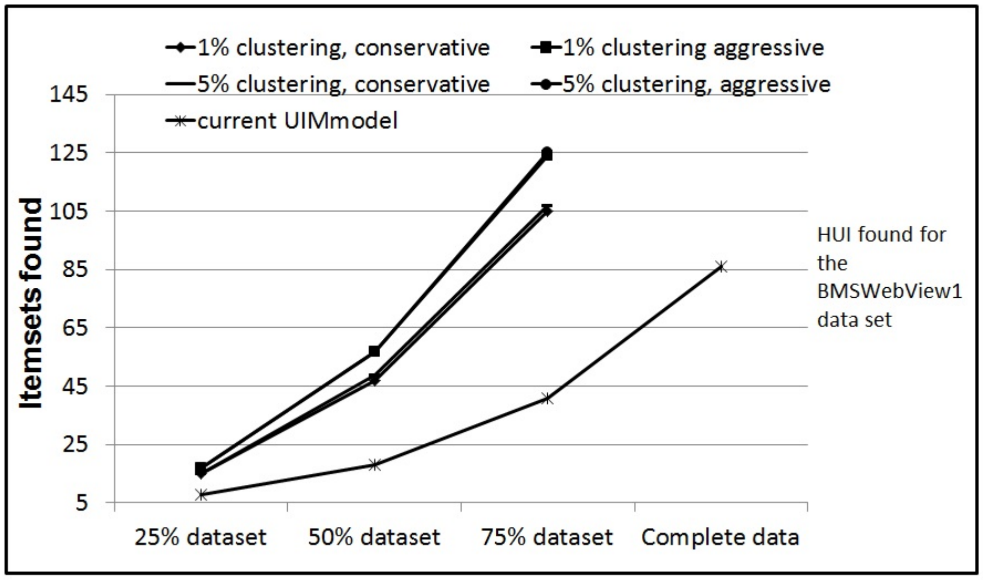


Figure 4.5 HUI found for the BMSWebView1 dataset

### 4.5.3 A note on prediction accuracy

Since we propose this new UIM model as a predictive one, we need to address the accuracy of this prediction with respect to a baseline. Since the current UIM model does not do any prediction, it cannot be considered a baseline. As in our model we are inflating the utility of certain transactions (and hence itemsets), we need to establish that the decision to do it to chosen transactions is better than doing sp uniformly to all transactions. In other words, how much the accuracy suffers if we were to inflate the utility of every transaction in the data. We performed an Itemset search by doing this (inflation by a factor of 3) and discovered that the accuracy suffers heavily. Specifically accuracy here means how many of the predicted itemsets (Addition HUI found) are indeed found to be present in the future data. The inflation by factor of 3 is a baseline for our aggressive cluster utility assignment. For the Retail dataset accuracy dropped to 50.2% (from 96.2%) and 24.9% (from 84.9%) when working on 50% and 75% data respectively. While the for the BMSWebView1 dataset it dropped to a 44.7% (from 89.5%) and 19.6% (from 66.4%) when working on 50% and 75% data respectively. The performance of our predictive model is significantly better (refer Figure 4.2 and Figure 4.3) than these.

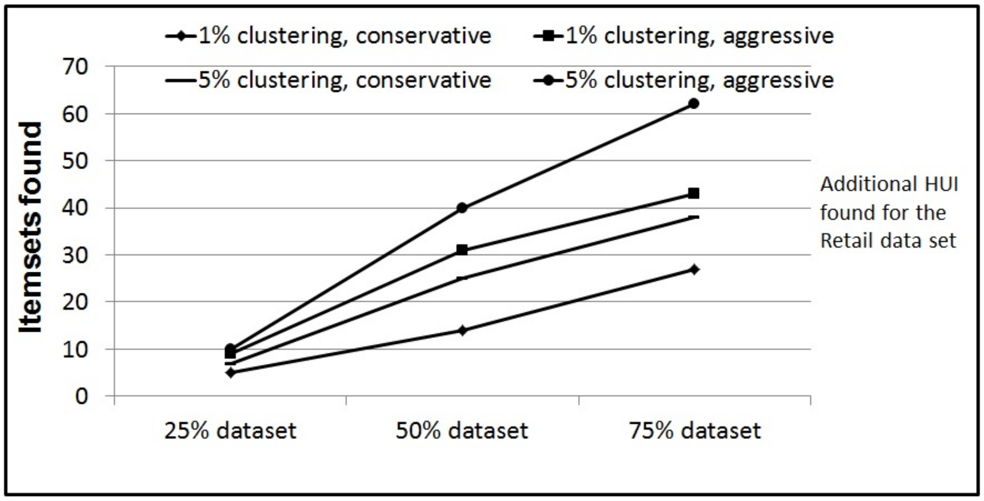


Figure 4.6 Additinal HUI found for the Retail dataset

## 4.6 Example practical impact of the enhanced predictive UIM problem model

Data is used to guide forecasting, planning and decision making in almost all science and business applications. Availability of actionable information is time critical for various reasons ranging from generating more profit for businesses or early release of a drug. Faster processing of the data is one of the ways to achieve actionable information sooner. However when availability of data is the bottleneck (which is the case for many applications in present times), it is most important to extract as much actionable information from the data as possible. With all the data available as well it is always preferred to extract as much useful information from it as possible. We perform an illustrative experiment to demonstrate that the benefit of the predictive UIM problem model.

For illustration, let us assume that for a retail store with no advertising 1000 of items in each HUI are sold every month. With correct advertising assume a X % increase in the sales. By correct advertising we mean advertising based on discovered HUI from the data. Therefore the sales achieved by the store in a month will be based on their choice UIM problem model used in the analysis. For this analysis we use 50% of the Retail dataset with $\Phi = 50000$ and 10% of the transactions for clustering. The results are shown in Figure 4.8 (extracted from Lakhawat et al. (2017)).

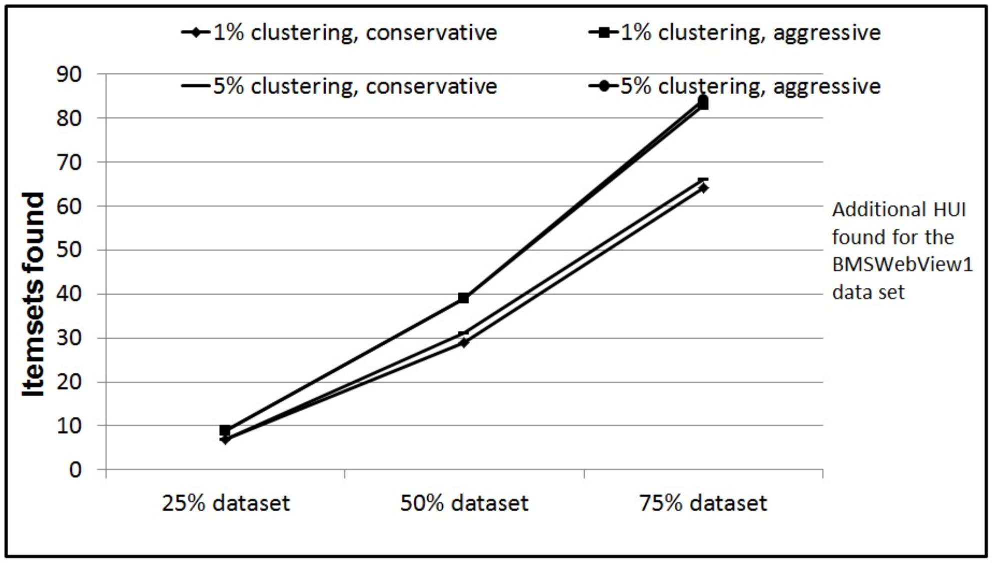


Figure 4.7 Additional HUI found for the BMSWebView1 dataset

## 4.7 Conclusion

We establish that the current Utility Itemset Mining (UIM) problem model can be extended by adding a key modeling capability of prediction by capturing cluster specific patterns in the dataset. All transactions possess information in them regarding the degree to which they belong to a cluster of similar objects from the entire data. If a transaction is fairly representation of cluster type then the information in it is more characteristic of their cluster type than the entire data. Therefore ignoring this knowledge and subjecting these transactions to the common threshold in the UIM problem leads to information loss.

We identify that an implicit use of cluster structure of data in the UIM problem model will address the above limitation. We do this by introducing a new clustering based utility in the definition of the UIM problem model and modifying the definitions of absolute utilities based on it. This modified predictive UIM problem model enables the cluster specific patterns to emerge while still mining the inter-cluster patterns and can integrate into all UIM techniques. Through performing experiments on two real data sets we are able to verify that our proposed predictive UIM problem model extracts more useful information than the current UIM model. This enhancement

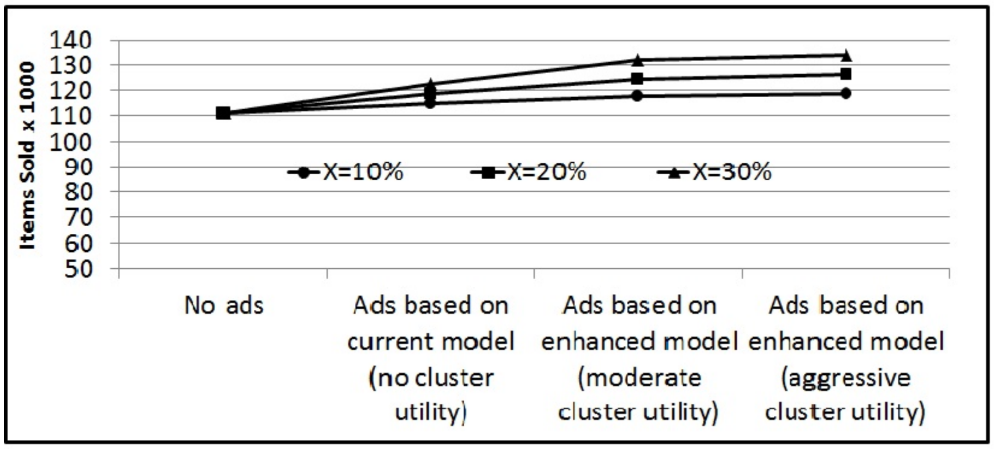


Figure 4.8 Example impact of UIM model used

in the UIM problem model leads to improved information extractions by facilitating a sooner (using less data) discovery of HUI and also discovery of cluster specific useful patterns.

# CHAPTER 5. ADAPTIVE CLUSTER BASED DISCOVERY OF HIGH UTILITY ITEMSETS

Utility Itemset Mining (UIM) is a key analysis technique for data which is modeled by the Transactional data model. While improving the computational time and space efficiency of the mining of itemsets is important, it is also critically important to predict future itemsets accurately. In todays time, where both scientific and business competitive edge is commonly derived from first access to knowledge via advanced predictive ability, this problem becomes increasingly relevant. We established in our most recent work that having prior knowledge of approximate cluster structure of the dataset and using it implicitly in the mining process, can lend itself to accurate prediction of future itemsets. We evaluate the individual strength of each transaction for itemset prediction purposes, and reshape the transaction utilities based on that. We extend our work by identifying that such reshaping of transaction utilities should be adaptive to the anticipated cluster structure, if there is a specific intended prediction window. We define novel concepts for making such an anticipation and integrate Time Series Forecasting into the evaluation. We perform additional illustrative experiments to demonstrate the application of our improved technique and also discuss future direction for this work. This work has been selected for publication with the Springer group in a special journal edition for selected and revised proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2017, held in Funchal, Madeira-Portugal, in November 2017.

## 5.1 Introduction

While the knowledge of cluster structure can potentially aid greatly in identifying future patterns, there is further temporal information which they can provide. The transactions are presented in timestamped order, and we intuitively understand that patterns emerging via incoming transactions will have a temporal aspect to them as well. In the same way, the cluster structure also has a dynamic temporal aspect to it. To understand it simply, in the example that we have been discussing above, it is quite possible that a particular cluster is showing higher propensity for increase in it utility contribution. What this exactly means will become clearer in the following sections. For now we can interpret it as, if a particular cluster (say the college students) shows a comparatively rapid increase in contribution to the populations (number of purchases), it will also have greater contribution in future itemsets. If we can adapt to the rate of change of this contribution, we can improve our prediction quality and target it to a specific window in future. The definition of this future can be based on the interest of the analysis. The key to using this information effectively will depend on developing meaningful concepts which can capture it and also allow the use predictive tools (like Time Series forecasting) on them.

We now build up on that idea by defining concepts for anticipated "cluster contribution" for a future window of itemsets, and connect it to Time Series forecasting. We then integrate this knowledge into our Itemset Mining model (Lakhawat et al. (2017)) to predict itemsets specifically with the inclusion of this future window. Whenever there is the concept of prediction, the concept of an intended prediction window is eminently implicit. Therefore, in case of itemsets as well there is a definite need to obtain predictions pertaining a specific window on time in the future. In this work, we strive to make begin making this need full.

The proposed prediction scheme here for high utility itemsets captures frequency, utility, cluster structure information and the dynamics of the cluster structure to predict the possible future itemsets with high accuracy. We have shown via experiments (Lakhawat et al. (2017)) on real datasets that we are able predict a good number of future itemsets with high accuracy over the baseline. And with the addition of the adaptive aspect in this work, the application of our technique becomes further rich.

## 5.2 Towards an adaptive discovery framework

After understanding how the evolved UIM can have predictive capability for future itemsets in Chapter 4, let us now understand how it can be done adaptively for a specific future window. We have identified that a metric of affinity is required for our model to function. This metric can be used interchangeably between transaction-transaction, cluster-cluster, cluster-transaction pairs. This specific affinity metric used here can be noted in the Chapter 3 in equation 3.3. As we mentioned previously, the key to achieving this adaptive discovery will lie in capturing potential contribution of clusters in the future itemsets. In fact, if we have an idea of contributions of clusters relative to each other, that should suffice as an implicit rationale behind our work is that most transactions are "emerging" from a cluster or a weighted combination of them. The affinity metric can help obtain that.

Before we can start to anticipate such future contributions, we need to establish what the cluster contributions are up to the present moment in time. As we iterate through the transactions serially in the dataset, we can compute an affinity value for that transaction for each cluster. This can be interpreted as the contribution of the cluster in this particular transaction. And if we normalize these values across all clusters, we can obtain the contributions of clusters relative to each other for this transaction. Now if we compute a running sum of these contributions for each cluster, then at any point in this run, the a sorted list of normalized values of these contributions will also the representation of the cluster contributions to the dataset and the itemsets discovered up to this point.

Now these running sums will be in a timestamped ordered series, so we can estimate a future value at any point in time for these running sums. While these estimates will only be predictions, they will be based on the trends encountered in this "contribution metric" so far. Therefore if we calculate these estimates of cluster contributions, we can know how much of the itemsets to appear up to this future point will be "emerge" from each cluster. Again, a sorted list of normalized values of these estimates will represent the "anticipated" contributions of clusters. For performing the actual estimation, Time Series forecasting techniques can be applied. Since the cluster development in each analysis can be unique, no one single Time Serires model or set of parameters can be assigned which will perform well in each case. However, certain traits will be there. As this metric is a rolling sum, and therefore monotonically increasing, the series is not stationary. Therefore, stationarity has to be induced each time. We suggest starting by differencing the series, and such trends will usually be captured well by that. Autocorrelations maybe be checked, but for most cases we suggest not using it as the transactions can be coming in a random order in terms of their being influenced by individual clusters. And finally, we suggest keep a moving average component in the Time Series model, as there will some propagating baseline contribution for each cluster and this can help capture that.

The next step now would be transferring back the knowledge of these anticipated cluster contributions to the predictive UIM model which we described in the previous section. An intuitive way to do that would be selecting the clusters with the highest and the lowest anticipated contributions in the intended future window, and inflating the importance of the former and reverse for the latter in the analysis. This can be easily accomplished by reshaping the cluster utilities (as describe earlier) of each transaction. Each transaction has a cluster counterpart in that analysis which it is most affine to, and there is a tunable parameter "k" which reflects how aggressively we wish to pursue the predictive aspect. So for the cluster with the highest anticipated contribution in the intended future window, we assign most aggressive values of "k" and for the cluster with the lowest anticipated contribution we assign most conservative values of "k". This is of course the most straightforward way of incorporating this "adaptive" aspect of our UIM model. Multiple clusters can be selected based on their anticipated contributions, and then a gradient of aggressive to conservative assignment of "k" can be performed while reshaping the cluster utilities for each transaction. Once the new cluster utilities are assigned, then it is just about generating the itemsets based on those using the same methodology we described in the previous section. The itemsets thus generated would include the predictions for the intended widow while adapting the anticipated contributions for the cluster structure.

## 5.3 Data Illustration

As we discussed in the previous sections describing the adaptive Itemset discovery aspect, this aspect of the model can be very subjective in terms of choice of Time Series model and parameters to use, and the format of incorporation of the anticipated cluster contributions in doing the prediction. Therefore, instead of providing concrete guidelines we perform an illustration of how to proceed in performing this adaptive itemset discovery. We do this in the following steps:

1. We create a cluster structure using 1% data of the BMSWebView1 dataset based on our clustering algorithm as described earlier.

2. We select the first 50% slice of the BMSWebView1 dataset, and evaluate the rolling cluster contribution sums for it.

3. We estimate the anticipated contributions for each cluster to the point in time where 75% of transactions will arrive, and normalize them. For doing this we difference the time series by oder 1 and use a moving average component of magnitude 1. For computation, we used the statmodels library Seabold and Perktold (2010). This model and parameters were picked based on trying various combination and selecting the one which is able to reasonably model the series data. It should be noted that highly accurate estimations are not required here as we only need an idea of relative order among the estimates.

4. We identify the cluster with the highest and the lowest anticipated contribution, and do the following with the transactions associated with them for cluster utility assignment purposes:

$$\text{lowest anticipated, } k = \{ {}^{0 \text{ if affinity}(T_i, C_j) < 0.25}_{1 \text{ otherwise}} \tag{5.1}$$

$$\text{highest anticipated, } k = \{ {}^{1 \text{ if affinity}(T_i, C_j) < 0.4}_{3 \text{ otherwise}} \tag{5.2}$$

5. After doing this reshaping we evaluate the itemsets for the 50%slice of BmsWebView1 dataset for $\Phi = 20{,}000$ using the updated utilities and the predictive UIM model. This would include

the predictions based on the adaptive anticipations based on 50% data for when the data should reach 75%.

6. And lastly, we evaluate the itemsets for the 75% slice of the dataset using the classical UIM model. We discover that using the adaptive search results in 10% more itemsets discovered in this case (predictions), of which none are inaccurate (all are present in the 75% data). This number will however vary when we change of the several parameters we selected for this illustrative experiment. The adaptive search approach presented here should used a exploratory analysis technique and be tuned based on the dataset, cluster structure and application requirements.

## 5.4 Conclusion

In this extended version of our previous model Lakhawat et al. (2017) we revisit that the current Utility Itemset Mining (UIM) problem model can be extended by including an important modeling capability of prediction, in identifying cluster specific patterns in the dataset. The idea that all constituent transactions possess information in them of the degree to which they belong to a cluster of similar transactions in the whole data. Ignoring the presence of cluster structure and subjecting all transactions to the common threshold in the UIM problem leads to information loss.

Implicit use of cluster structure of data to solve this problem is highlighted. The use clustering based utility in the definition of the UIM problem model and modifying the definitions of absolute utilities based on it allows the cluster specific patterns to emerge while still mining the inter-cluster patterns and can integrate into all UIM techniques. Experiments results on two real data sets are presented to verify that the predictive UIM problem model extracts more useful information than the current classic UIM model. Significant revisions and extension in this work are:

1. We identify that there is a need of adaptive anticipation in our predictive Utility Itemset Mining model.

2. We develop and define concepts which can capture the idea of adaptive anticipation of the cluster structure.

3. We integrate these concepts into the predictive model by firstly discussing ways to analyze and evaluate the metrics associated with them, and then transferring that knowledge into our itemset mining model. A core part of this scheme proposed by involves fitting and forecasting Time Series models.

4. We conduct an illustrative experiment showing the way to perform adaptive cluster based discovery of High Utility Itemsets in a straightforward manner. We recommend an exploratory rationale to conduct such analysis

For the future work, we plan to incorporate additional techniques which will aid in the adaptive search. We believe using supervised classification techniques to adapt to application/analysis/data specific requirements can help the model further evolve.

# CHAPTER 6. BIG DATA CHALLENGES: FROM THE LENS OF CLUSTER ANALYSIS

All knowledge discovery techniques have to be responsive to the three Vs (Volume, Velocity and Variety) of bigdata to be effective. Cluster analysis is a class of unsupervised learning problems where the goal is to obtain a set of clusters (group of similar objects) for a given data set. Each cluster is unique in terms of a chosen set of attributes. For best use of clustering techniques in the bigdata age, it is imperative to understand the effect of the nature of bigdata on them. In this chapter, we discuss how the three Vs of bigdata affect various categories of clustering techniques. We also discuss three future directions for bigdata cluster analysis research. We believe that the future clustering algorithms should strive to be distributed in nature, robust to noisy data and should focus on development of generic clustering engines which can process a variety of heterogeneous data. This work has been published in Somani and Deka (2017)

## 6.1 Introduction

Cluster analysis (Clustering) is a fundamental problem in unsupervised machine learning domain. It has a huge range of applications ranging in varied fields like bioinformatics, gene sequencing, market basket research, medicine, social network analysis, and recommender systems etc. The main idea in clustering is to arrange similar data objects from a given dataset into clusters (groups). Clusters usually represent some type of real world entities or a meaningful abstraction. The knowledge of the abstraction acts as a stepping stone to further analysis and is therefore a fundamental requirement for data analysis problems. With the growing capabilities in data collection, transmission, storage and computing, there is a steady increase in cluster analysis based research.

The realm of bigdata has affected almost all knowledge extraction techniques. The existing data mining techniques have had to adapt to the three primary facets of bigdata: Volume, Velocity

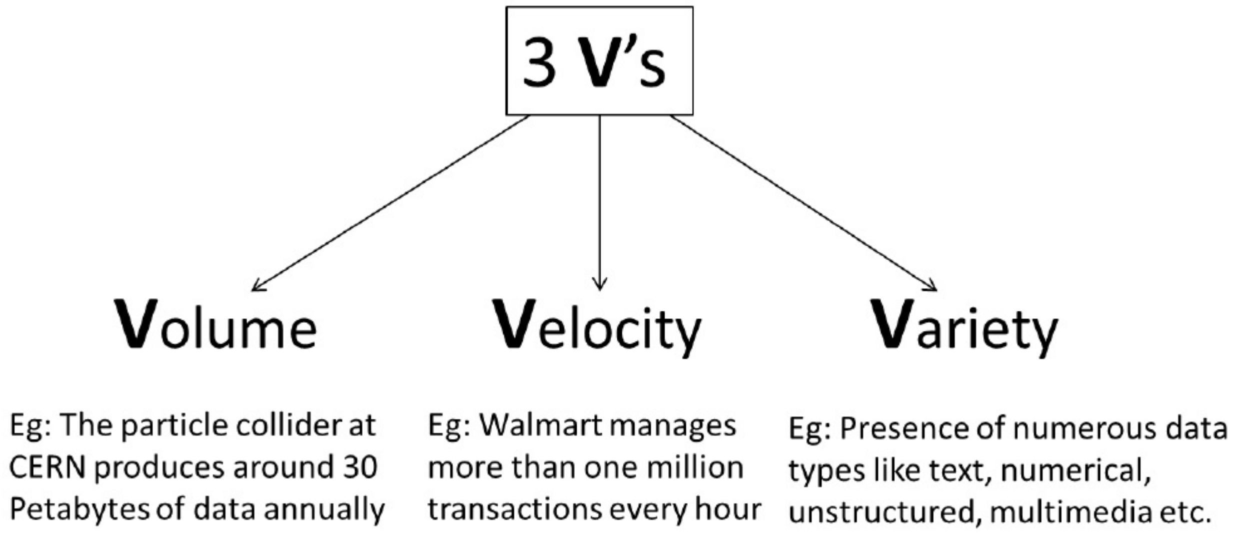

3 V's
Volume
Velocity
Variety
Eg: The particle collider at CERN produces around 30 Petabytes of data annually
Eg: Walmart manages more than one million transactions every hour
Eg: Presence of numerous data types like text, numerical, unstructured, multimedia etc.

Figure 6.1 The three primary V's of Big Data

and Variety (Laney (2001)). These are also referred to as the three Vs of bigdata, as depicted in Figure 6.1 (extracted from Somani and Deka (2017)). Volume refers to the big size of data being collected every day and used for analysis. Velocity refers to the continuous data collection pipelines active in various fields. Variety refers to the heterogeneous nature of the data ranging to text, image, numerical etc. in unstructured, semi-structured and structured form. The ways in which these three Vs affect any machine learning technique can be very different based on its functioning.

In this chapter we first will provide an overview of cluster analysis along with a broad categorization of clustering techniques. Following this, we discuss each category of clustering technique along with a detailed study of one prominent technique in that category. While studying these techniques we also focus on aspects which are likely to be affected by the three Vs of bigdata. Finally we discuss the important future directions for bigdata cluster analysis research and provide our perspective on them.

## 6.2 Overview of Cluster analysis

Any data considered for cluster analysis is implicitly expected to contain a set of clusters in it. Cluster can be imagined as group of data objects which are similar to each based on a specified criterion. Individual clusters are expected to possess different qualities (usually judged by a specific metric) than distinguishes them one another. Such a scheme of clusters is found in many real life applications and knowledge of such schemes helps directly or indirectly to extract useful information from the data. The varied nature of data and the types of similarity desired in clusters leads to the development of various clustering techniques. The eventual goal always is to obtain a suitable condensed representation of the data. Figure 6.2 (extracted from Somani and Deka (2017)) gives a visual illustration of the clustering process.

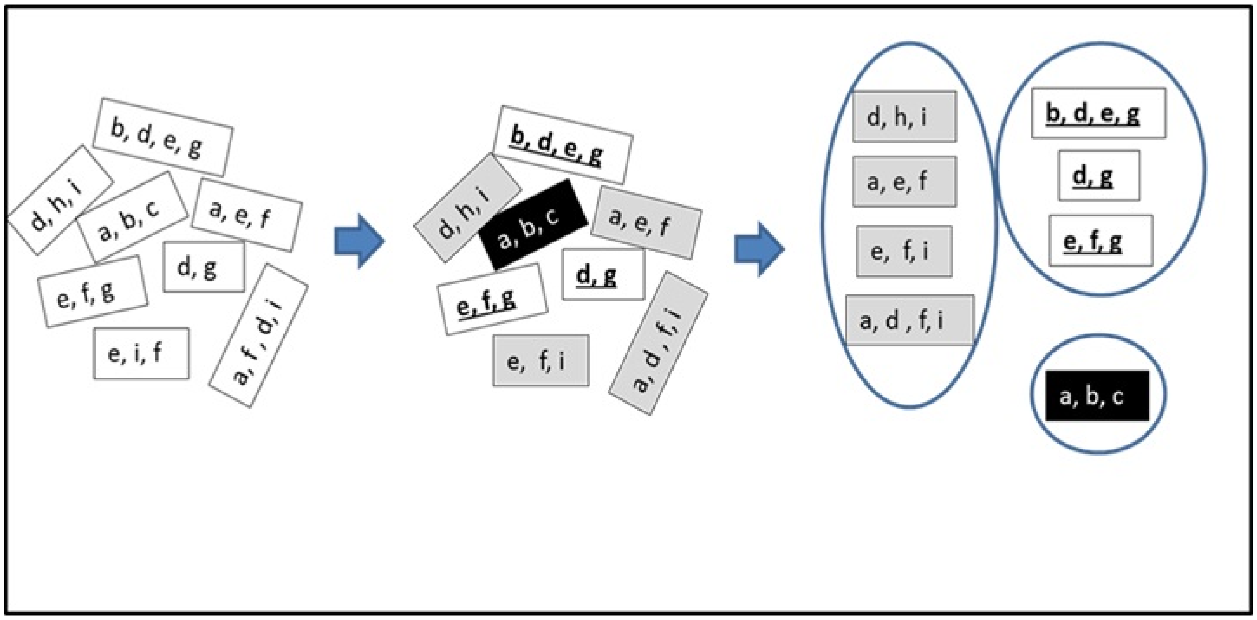


Figure 6.2 Visualization of Clustering

The clusters obtained from the given data are sometimes the desired final information. For example, with a clustering analysis of bio-informatics data researchers are able to identify useful associations among many datasets. Such information helps in designing future experiments. While on other occasions the knowledge of clusters can help quickly associate a new object to a cluster type and make a corresponding decision for it. For example, based on a customers shopping basket an online retailer can immediately try to predict which type of a customer (that is most similar to which cluster) she is and recommend suitable products. Attributes like the type of data (numerical, categorical, multimedia etc.), the expected properties of the data (correlation, skewness, mean etc.), size and speed of the data etc. all play an important role in deciding the type of cluster analysis to be performed. We use a small example scenario to explain the intuition behind the clustering problem.

### 6.2.1 An illustrative example

Let us consider a log kept of all items bought by each customer by an online retail store. Table 6.1 (extracted from Somani and Deka (2017)) is an illustration of one such log of a short period on the store database.

Table 6.1 A toy illustration of transactions in 5 minutes at a retail store website

| Customer ID | Time of purchase | Items bought |
|---|---|---|
| 1 | 3:00 PM | Printing paper, Printer cartridge |
| 2 | 3:02 PM | Detergent, Cooking oil, Bread, a Novel abc |
| 3 | 3:02 PM | Coffee beans, Sugar, Bread, a Novel abc |
| 4 | 3:03 PM | Bread, Coffee beans, Soap |
| 5 | 3:03 PM | Tshirt, Shorts, Socks |
| 6 | 3:05 PM | Tshirt, Shoes |

By observing the contents of Table 6.1 we can notice that customers 2, 3 and 4 bought similar items and customer 5 and 6 bought similar items, but different than customers 2, 3, and 4. Items bought by customer 1 are not similar to any other customer. This observation points to an attribute of the dataset that there are types of customers which buy similar items. But how do we formally analyze and report such an attribute? This is one of the primary question that needs to be answered by a cluster analysis.

Let us try to identify the important factors behind developing such a cluster analysis technique. First we need to identify a metric to compare the similarity of various customers. Let us use the number of common items bought by two customers as a metric in this case. So based on this metric we can state that customer 2 and customer 3 have two common items, and therefore, are more similar than customer 5 and customer 6 who only have one common item. Second, we also need to identify a metric of similarity for clusters (groups of customers in the example). A possible metric can be the ratio of number of common items between the two clusters to the mean size of the two clusters.

**Similarity.** If we consider a cluster of (customer 2, customer 3) and a singleton cluster of (customer 4), then the similarity between them would be $\frac{2}{1.5} = 1.3$. Note that the definitions of similarity for clusters of customers and single customers become equivalent for singleton clusters. This can be a flawed similarity metric because once the cluster sizes become varied, the effect of averaging will distort the similarity values. However for the sake of simplicity let us continue with this definition. It should be noted that real world clustering techniques are designed to cater to such problems like scalability among others.

**Clustering Step.** Let us now try to devise a clustering algorithm based on this definition. One possible way would be to start with assumption that each customer transaction is a singleton cluster and merge them incrementally based on the definition of similarity. In each incremental step we can merge a pair of clusters with most similarity. This idea is actually the skeleton of an important class of clustering techniques called hierarchical agglomerative clustering.

**Termination.** One final step that we now need to perform is a criterion to terminate the clustering process. A simple way in this case can be a predefined number of clusters to be obtained before stopping. Many existing clustering techniques do follow such a criterion. However the state of the art techniques try to identify such a criterion based on the dataset and how the clustering process is proceeding. That is, whether the formation of new clusters is yielding new information or is it over simplifying the dataset. In this example, let us assume that we shall find three clusters and terminate the process.

Following this clustering technique steps identified above, we will first merge customers 2 and 3 or 3 and 4 into a cluster. Following that we will form a cluster of customers 2, 3 and 4. Next, we will form a cluster of customers 5 and 6. At this point we have three clusters formed: (customer 1), (customer 2, customer 3, customer 4) and (customer 5, customer 6). We can name them clusters A, B and C, respectively. This gives us a formal way to analyze and report any clusters present in a dataset of type present in Table 1. More sophisticated version of processes lead to the development of various cluster analysis techniques. The knowledge of clusters in any dataset acts as a final or intermediate step in further analysis for various scientific and business applications. For example, in our illustrative example here assume a new customer 7 adds the following to her cart: Bread, Coffee beans and Sugar. Then we can immediately associate this customer to cluster B and suggest the Novel abc for her to buy. With this understanding, let us now discuss a broad categorization of clustering techniques.

### 6.2.2 Types of clustering techniques

Clustering techniques are developed and chosen based on the prior understanding of the data. Since these belong to the unsupervised machine learning category, making a right choice of clustering technique is very important. Based on the wide variety of data types, many corresponding clustering techniques have been developed, as depicted in Figure 6.3 (extracted from Somani and Deka (2017)). Most of the clustering techniques can be categorized based on the cluster building methodology. The four broad categorizations (Berkhin (2006)) based on it are: Hierarchical clustering, Density based clustering, Partitioning based clustering and Grid based clustering. There are few other techniques which do not fall under any of the above categories. They can be categorized them under a fifth category termed as Miscellaneous clustering techniques. In the following sections, we briefly discuss all of them while discussing one specific example from each category in more detail.

## 6.3 Hierarchical Clusteringtle

As the name suggests these techniques follow a hierarchy (ranking) criterion to perform the clustering. This ranking is typically based on some metric of similarity among clusters and a top-down or bottom-up approach is used based on the choice of clustering.

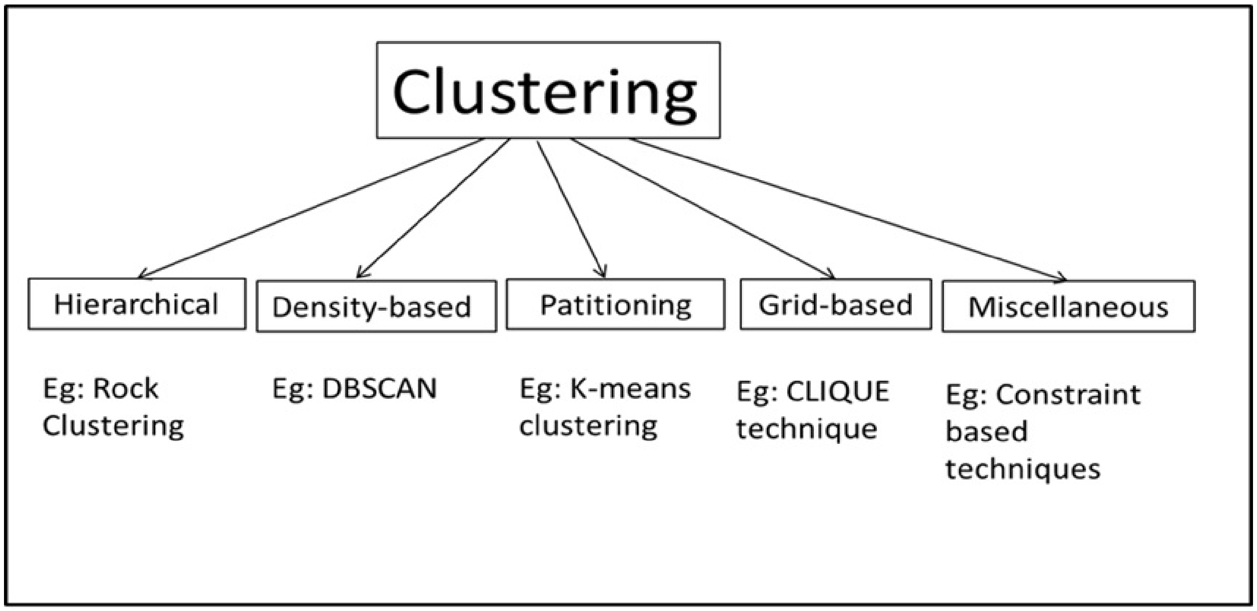


Figure 6.3 Categorization of clustering types

The top-down and the bottom-up hierarchical clustering are also commonly known as divisive and agglomerative hierarchical clustering respectively. Figure 6.4 (extracted from Somani and Deka (2017)) aids in visualizing the two types of hierarchical clustering. The key benefits of hierarchical clustering are:

1. Ability to define data specific merging/dividing criterion, making it widely applicable
2. Option to achieve cluster structures of various shapes and sizes based on the termination criterion

However, the same reasons can also lead to inaccurate clusters sometimes due to:

1. Lack of knowledge of how the merging/dividing criterion will influence the clustering process
2. Not knowing the best level to stop the clustering can cause information loss as well

A key concept in hierarchical clustering is a dendrogram. It is a tree type structure made of clusters based on hierarchical relationships among them. Since it is a tree type structure, there is a natural concept of child, sibling and parent. Children of a cluster divide its data into sub-clusters. Therefore it is very convenient with hierarchical clustering to achieve any desired level of coarseness/fineness. As shown in Figure 6.4 bottomup clustering starts by defining each data point as a single cluster and keeps merging clusters based on similarity. Top-down clustering follows the exact opposite criterion by defining the entire dataset as a single cluster and keeps dividing it into sub-clusters based on dissimilarity. Both the processes continues until a termination criterion (frequently, a requested number k of clusters) is achieved.

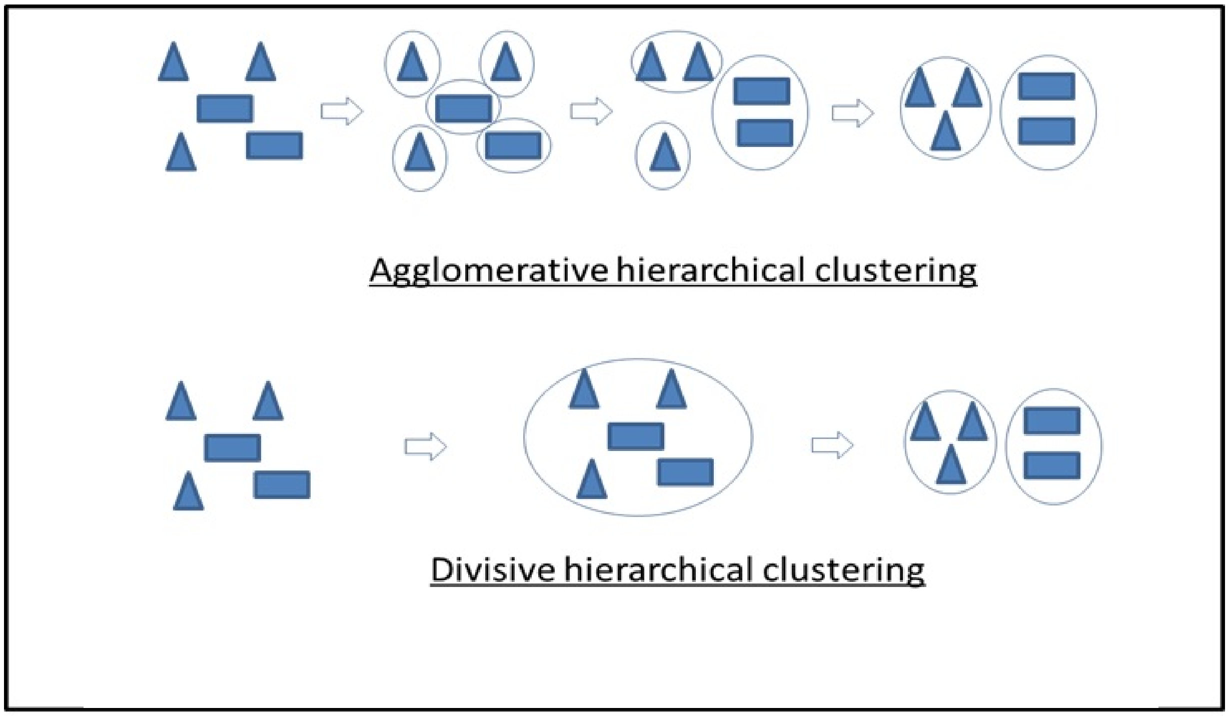

Agglomerative hierarchical clustering
Divisive hierarchical clustering

Figure 6.4 Two types of hierarchical of clustering

**Agglomerative Hierarchical Clustering.** For agglomerative hierarchical clustering there can be numerous choices for merging techniques with different pros and cons among them based on complexity, convergence speed etc. A popular choice is to calculate average distance between all possible pairs of data points in the two clusters to be used as the final distance between them. Other techniques recommend using the difference between some versions of mean of data in the two clusters to evaluate their distance. Techniques like using the shortest distance between any two data points to be used as cluster distances are not recommended as these lead to big clusters getting bigger over time. In case of large clusters, it is not uncommon to use variety of sampling techniques to reduce the computational complexity while still achieving reasonable accuracy.

**Divisive Clustering.** As shown in Figure 6.4, divisive clustering techniques initialize with the entire dataset defined as a single cluster. The final goal is to achieve a dendrogram of desired qualities with all leaf nodes as the final set of clusters. In certain situations divisive clustering can be favored over agglomerative clustering, especially when clusters of varying granularity are expected or more than one type of clustering criterion are required. Due to being a hierarchical clustering, there is an option of evaluating the clusters formed at each step. Therefore, if it is expected that current data arrangement already consists of few clusters then divisive clustering can save a lot of computation by partitioning those clusters in just one step.

A suitable similarity (or dissimilarity) index is very important in all hierarchical clustering techniques. The reason being these indexes drive the entire clustering process and therefore inaccuracy in them will propagate throughout the whole analysis. Therefore, the state of the art techniques of this category justify their choice of such metric very succinctly. And follow it by verifying the validity of clusters found in applicable datasets. Such metric are also required to have good scalability because of their repeated usage in their respective clustering algorithm and the possibility of large clusters in the data. Let us now study in further detail one of the most popular hierarchical clustering techniques called the ROCK clustering.

### 6.3.1 ROCK clustering technique

For an algorithm specific understanding of the hierarchical clustering technique we shall analyze the popular ROCK clustering algorithm (Guha et al. (1999)). The main strength of this algorithm is that it provides a very intuitive clustering logic for categorical data. Ideas like using average or other numerical operations are not applicable to categorical data. To overcome this, ROCK clustering develops a similarity graph, with links connecting objects based on similarity among them. The core concept is that more number of links between objects implies more similarity. Such methodology allows application to numerical, Boolean and categorical data or their combinations. Zaïane et al. (2002) mentions that since this algorithm uses global parameters, it inhibits the clustering process to generate naturally present clusters of uneven sizes.

The idea of links is very important in this clustering technique. The authors define a link(a,b) to be the number of common neighbors to a and b. Therefore higher values of link(a,b) lead to higher chance for a and b to be in the same cluster. The objective is for each cluster to have its highest possible degree of connectivity. Hence the sum of link(a,b) for the data point pairs (a,b) should be maximized for each cluster as well minimized for each pair (a,b) from different clusters. This rationale leads to the authors to the following criterion function for k clusters which they try to maximize:

$$\sum_{i=1}^{k} n_i \sum_{a,b \in C_i} \frac{link(a,b)}{n_i^{1+f(\theta)}} \tag{6.1}$$

They define $f(\theta)$ to be a function which depends on the data and the clusters of interest. The function $f(\theta)$ is designed to have the following property: each data point in cluster $C_i$ should have approximately $n_i^{f(\theta)}$ neighbors in the cluster $C_i$. Besides ensuring strong connectivity, $f(\theta)$ also allows clusters to be unique. It is explained in further detail in Guha et al. (1999). With the goal of maximizing this function, the ROCK algorithm follows an agglomerative approach to incrementally merge clusters until a predefined number of clusters are achieved.

### 6.3.2 Expected effects of three Vs on ROCK clustering technique

The computation of links is one of the more expensive steps in the technique with the authors achieving complexity of $O(n^{2.37})$ with matrix multiplication techniques. This however will not be the biggest challenge among the three Vs of bigdata as there is potential to achieve distributed versions of this technique. Overall algorithmic complexity is described by the authors to be $O(n^2 + nm_m m_a + n^2 logn)$ in the worst case, where $m_a$ is the average number of neighbors and $m_m$ is the maximum number of neighbors of a point. Since this overall nature of the algorithms is agglomerative in nature, many steps in the algorithm can be distributed.

The Velocity of data can be a bottleneck if it is above certain bounds, but we believe the biggest challenge to adopt for ROCK clustering (and most other hierarchical clustering) techniques would be of the Variety of data. The link based connectivity definitions will be too simplistic for rich variety and unstructured datasets. The knowledge of number of clusters needed to be formed will also be almost impossible to get Apriori in most cases. Therefore in future versions of such algorithms it will be imperative to develop clustering progress monitoring techniques, so that each cluster analysis can be tailored for specific datasets.

## 6.4 Density based clustering

If we interpret a dataset as a density distribution in some Euclidean space, then the connected components of such a space can be interpreted as the clusters present in the dataset. The central idea in density based clustering is to do such mapping as accurately as possible, and then through iterative steps improve the connectivity strength of clusters.

Every density based clustering algorithm starts with the strongly connected parts of the dataset as individual clusters and then keeps expanding these clusters by searching for points which enhance the density function. Combination of two reasons makes this category of techniques unique. Firstly the initial cluster assignment captures the global picture of the data set and ensures that the analysis doesnt strongly deviate from that. And second reason being that the iterations to improve the connectivity captures the local information related to each cluster and therefore improving the overall accuracy. This allows the density based techniques to detect clusters of varied shapes and sizes along with the freedom to rearrange data in any order necessary.

Clusters eventually found are dense sections from the data set filtered out and leaving behind the sparse sections. There is a risk in these techniques that sometimes two dense sections get fused together due to inaccurate parameter settings. Such oversimplifications need to be avoided. Figure 6.5 (extracted from Somani and Deka (2017)) visually illustrates this drawback. Due to lack of visual interpretation of data density in some cases, visualization can be challenging for density based clustering. It is not uncommon to find clusters of seemingly arbitrary shapes.

The implicit assumption in density based clustering is that the outliers and noise in the data dont have enough density to significantly affect other clusters or form a cluster themselves. Therefore at the end of density clustering any section of data which is declared sparse should be assumed as not useful in clustering analysis (Aggarwal and Reddy (2013)). Key ideas to focus on when working with a density based algorithm are density estimation and the definition of connectivity. Next we study and present details of one of the most popular density based algorithm called DBSCAN.

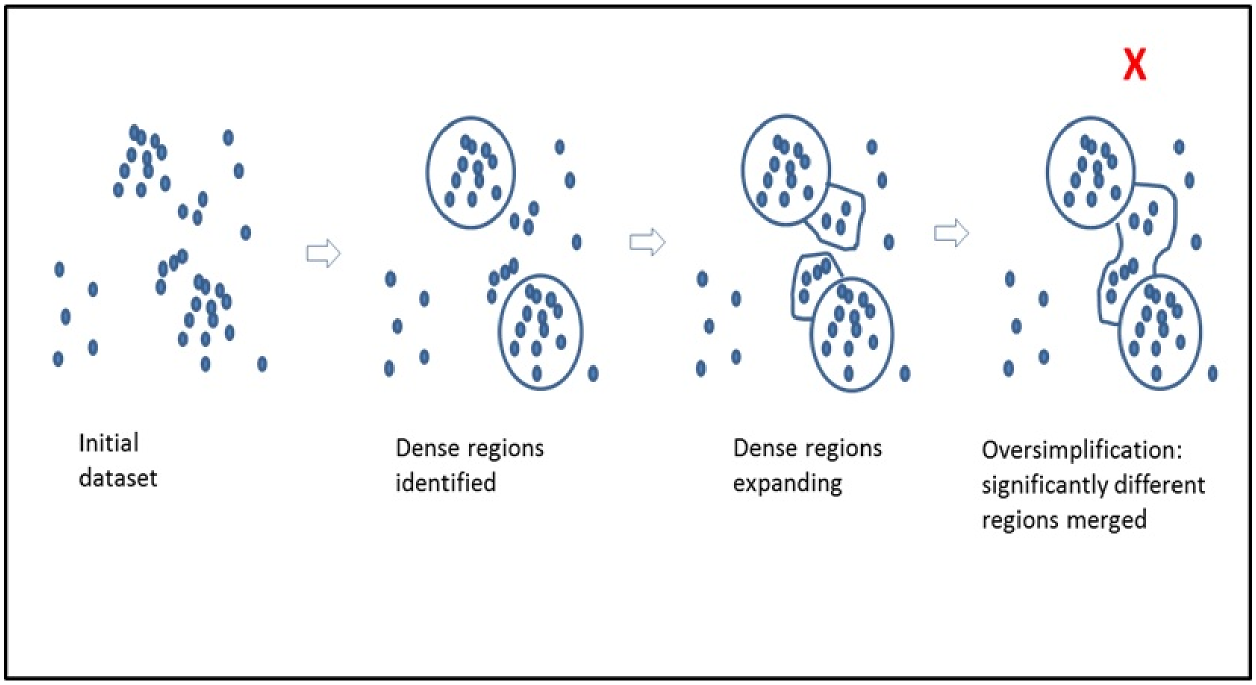


Figure 6.5 Potential drawback of density based clustering

### 6.4.1 DBSCAN clustering technique

DBSCAN (Ester et al. (1996)) builds up the clusters by finding densely connected data points in a dataset. Its starts by finding any "core point". A data point is called a "core point" if it has more than a minimum number of data points near it. By near it, we imply that the distance metric of the algorithm should be less than a pre-defined value (usually referred to as "Eps"). The minimum number of points required to call a data point as "core point" are referred to as MinPts. Any point which is in the range of "Eps" for a given point is said to be directly reachable from that point. While any point which can be reached by hopping on successive directly reachable points is called a reachable point. Therefore, points not reachable from any other data point are considered sparse and ignored from the analysis.

DBSCAN starts with any arbitrary point and checks all other points in its surrounding “Eps” space. If no point is found then this point is ignored and some other arbitrary point is selected. If point(s) are found in the “Eps” space, then points in their “Eps” are searched for. This goes on until each point in the cluster has no other reachable point from it. Once this is done, another arbitrary point from outside this cluster selected and the process is repeated. The algorithm terminates when each point is either belonging in some cluster, or is a completely isolated point (not reachable from anywhere).

It should be noted that even though the initial core point selection is arbitrary, every point in its associated cluster will be eventually found. And the same cluster would have been found even if any other point in that cluster would be the initial selection to start the process. However, this is true when the value of “Eps” and MinPts are fixed. The accurate estimation of these two parameters is always a challenge when using the DBSCAN algorithm.

### 6.4.2 Expected effects of three Vs on DBSCAN clustering technique

While the DBSCAN algorithm has many strong points, it still needs to adapt to the nature of bigdata. In its native form DBSCAN requires the entire dataset to perform clustering. This approach is infeasible even for moderate sized dataset in todays times. However, to cater to the large volume of parallel versions of DBSCAN have already been proposed (Januzaj et al. (2004)).

While the big Volume of the data is a problem, another problem is the Velocity of the data. This translates into a need for faster approximate versions of the algorithm. Certain sampling based versions of the DBSCAN have been introduced (Borah and Bhattacharyya (2004)). The goal of these techniques is to obtain a near accurate cluster structure while not having to process the entire dataset.

The Variety aspect of bigdata is also an important factor, and addressing that in a density based algorithm like DBSCAN can be especially challenging. The key challenge comes with suitable definitions for connectivity and density functions.

## 6.5 Partitioning based clustering

As the name suggests, partitioning based clustering relies on achieving the final clusters through successive partitioning and refinement steps. There is always either a clustering quality measurement criterion or an indicator function which decides the conclusion of the algorithm. The intrinsic benefit of partitioning based clustering methods is that they can exploit any existing data partitioning which the data is provided in and start building up the clusters from there. While some other techniques like hierarchical clustering will always start the clustering process from the same state irrespective of the formatting of data provided. The design of partitioning based algorithms also facilitates generating distributed and parallel solutions very convenient.

The idea of iteratively improving the search space has been very effective in various domains, like linear programming (Luenberger (1973)) or advanced sorting techniques (Mehlhorn (2013)) to name a few. Therefore it is not surprising that it yields strong results in cluster analysis as well. Classical partitioning techniques like the K-means algorithm is still highly used and is very effective for a variety of applications. A disadvantage of this type of clustering is that getting a rough idea of cluster structure can be difficult for the initial iterations.

### 6.5.1 K-Means clustering technique

If we were to develop a list of most used clustering algorithms, most would agree that K-means algorithm deserves a place in that list. The simple yet powerful clustering logic of this technique along with applicability to most numerical data applications are the reasons for it.

The key idea in K-means clustering is to identify the k means from the data set which are representative of most data points around them. While each of these k means are sufficiently different than each other. The choice of a suitable number k is a challenge and there are ways to estimate the best k for a given data set in order to avoid overfitting. The main function to be minimized in this technique is the distance between the mean (also referred to as centroid) and the points associated with each mean. Vector based distance like the following is a common distance function used:

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} ||x_i - C_j||^2 \tag{6.2}$$

In the above function k represents the number of clusters, $C_j$ represents a mean and $x_i$ represents a point associated with that mean. Such objective function is applicable when data is numerical in nature and also each data point has same dimensions. For other cases like categorical data, techniques with similar logic have been developed (Huang, 1998) which use mode as the central idea instead of mean.

The basic version of K-means algorithm (Hartigan and Wong (1979)), initializes the clustering process by defining first set of K-means and then assigning the points to each mean based on minimizing the total distances as per the objective function. This assignment is called partitioning the data. Following this, new means (centroids) are computed for each data partition individually. These individual means from each partition are now considered the new k means and are used to create new partitions based on minimizing the objective function. This process is ideally repeated until the new partitions are not changing. That is all the k means coincide with the individual partition means. Figure 6.6 (extracted from Somani and Deka (2017)) provides a visualization of the K-means clustering.

Finally, the initialization of the clusters also has an effect on the clustering performance. Milligan (1981) suggest the use of a dendrogram (via hierarchical agglomerative clustering) to identify the initial cluster assignment. While (Hartigan and Wong (1979)), use the nearest neighbor density to develop an initial clustering. A random sampling based approach is described in Bradley and Fayyad (1998).

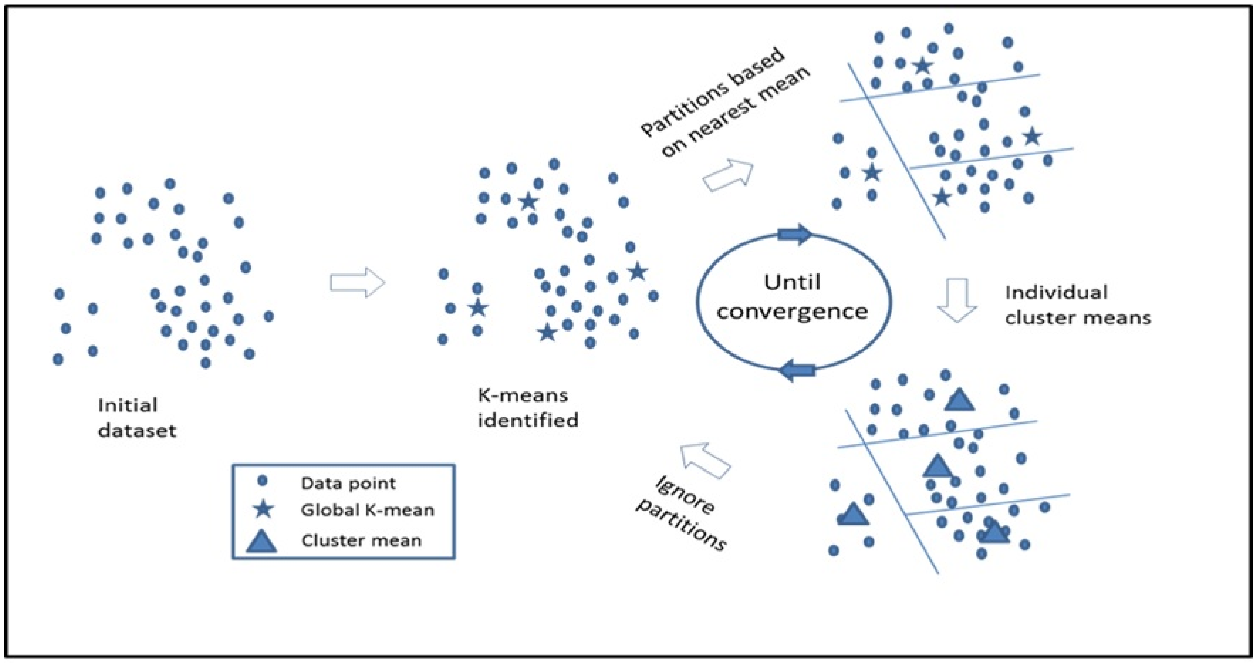


Figure 6.6 Key concept of K-means clustering

### 6.5.2 Expected effects of 3 Vs on K-means clustering technique

K-means will probably continue to be one of the more popular clustering techniques in the times of bigdata. Many variants of the K-means technique have been developed over time. Because the core premise of the technique is simple and effective, various types of numerical data respond well to this analysis. Since there is scope for both parallelization and adding machine learning components to the K-means technique, it will continue to be effective in terms of handling the Volume and Velocity components of bigdata.

However, further sophistication is needed in terms of handling the Variety aspects of the bigdata (like images, texts etc.). Combining other machine learning techniques (or even some other clustering techniques) with K-means to achieve more versatility in terms of data interpretation will be required.

## 6.6 Grid based clustering

It can be argued that most grid based clustering techniques can be classified as density, partitioning or hierarchical clustering techniques. Or at least be called as a combination of them. However, a true classification of a clustering technique should be based on its core clustering strategy. The key concept in grid based clustering is that of projecting the dataset (or more specifically the data space) to a more appropriate space, which can be easier to manipulate and identify the clusters in. This makes grid based clustering unique and separates it into its own category of techniques. High dimensional data sets found in current applications make this type of clustering especially relevant.

Reducing the computation complexity is one of the main goals of using grid based techniques. Once data is transformed into the desired grid space, rest of the process is to find clusters by using the grids. The grids can have attributes like mean or density associated with them. Success of grid based clustering techniques depends upon the accuracy of such transformations in capturing the essential data attributes required for clustering. Let us now discuss in further detail, the popular grid based CLIQUE clustering technique.

### 6.6.1 CLIQUE clustering technique

The authors in (Agrawal et al. (1998)) proposed the CLIQUE clustering algorithm. It is a grid based clustering algorithm which uses the concept of data density to locate clusters. Like all grid clustering techniques, the goal is to find clusters in the subspace of the high dimensional datasets. A strong feature of CLIQUE clustering is that it presents the final cluster description as a disjunctive normal form (DNF) expression. This being the minimal description of the clustering aids in understanding and interpretability of the cluster structure.

The authors claim that it is important to use a subset of real dimensions of data instead of creating new dimension by mathematical operations. The reasoning being that real dimensions represent some factual information, while created dimension might be difficult to interpret. Since this is density based clustering approach, a cluster is interpreted as a region with data points present in high density compared to other regions. Therefore, the CLIQUE clustering algorithm proceeds to identify simplifying transformations of dataset which either make high density areas visible or easier to locate.

The CLIQUE algorithm transformations begin with mapping its dataset points onto the units of the grid structure. A data point will be mapped onto a particular unit (grid) if the value of all the dimensions of this data point lie within the specified range of this unit. Once the transformation process (also called as taking projections) is over, all the units are observed and the units with more than a certain number of data points in them are considered dense. Remaining units are ignored. Different choices of dimensions to transform will result in different projections of the dataset. Goal is to find the most suitable projection for clustering.

Algorithmically, the clustering process can be divided into three components: identifying the subspaces (a bottom up technique is used to achieve this), identifying clusters in the subspaces (a graphical representation is used followed by using depth-first search) and finally generation of minimum description of clusters. The authors show that an optimal solution is NP Hard, and develop a greedy approach to achieve these.

### 6.6.2 Expected effects of three Vs on CLIQUE clustering technique

While the worst case behavior of the subspace discovery process is shown to be exponential in the highest dimension, in most practical cases it can be controlled by pre-defining the number of passes over the data set. This criterion will need to be studied in further detail with the increasing Variety in the datasets in recent times.

A strength of the CLIQUE clustering technique is that once the subspaces are identified, the cluster discovery and the minimum description generation can be done in a distributed fashion. This will help in subduing the effects of the Volume and Velocity aspects of biodata.

Overall such grid based techniques have the important feature of simplifying a complex dataset. This makes such techniques really promising going ahead. We expect a flavor of these techniques to be present in most future bigdata clustering techniques.

## 6.7 Miscellaneous clustering techniques

Certain clustering techniques do not fit in any of the above categories. Often these techniques are an amalgamation of a clustering technique and another application specific technique. Berkhin (2006) discusses these in much detail. In this section we provide an overview of these methods.

In recent years a new class of clustering has emerged from a practical point of view, called constraint-based clustering (Tung et al. (2001)). The need for constraint-based clustering is understandable as it is hard to imagine any real world applications without specific constraints associated with them. While traditional clustering techniques typically aim to maximize (or minimize) certain clustering criterion, constraint-based clustering tries to do the same along with maintaining the specified constraints validity. If constraint-based clustering was an optimization problem, then traditional clustering would be a problem of just maximizing the objective function. Tung et al. (2001) specifies few ways in which constraints are imposed in clustering algorithms: constraint on certain objects to be clustered together, constraint on objects not to be clustered together, bounds on certain algorithm parameters, specific constraints on individual clusters etc.

There have been certain modifications done on the partitioning clustering methods as well. For example, supervised learning techniques are used to train classifiers specific to clustering decisions (Liu et al. (2000)). These classifiers then influence the future decisions for partitioning. Swarm optimization techniques have also me used in conjunction with K-means clustering to accelerate the centroid discovery process (Van der Merwe and Engelbrecht (2003)).

Recent works have also used evolutionary techniques in clustering algorithms. The SINICC (SImulation of Near-optima for Internal Clustering Criteria) (Brown and Huntley (1992)) algorithm uses simulated annealing in a framework of partitioning based clustering. Genetic Algorithms are also being applied in cluster analysis. The GGA (Genetically Guided Algorithm) (Hall et al. (1999)) applies the fuzzy and hard means logic in finding the K-means clustering. The authors find that the use of genetic algorithms especially helps in the initialization aspect of the K-means clustering.

This is far from an exhaustive discussion of clustering techniques merging with other data mining techniques to provide improved application specific results. However, it sheds light on such a need in data mining research. With the data being richer and detailed in the future (Variety), the need to use multiple data mining techniques in combination is going to increase.

## 6.8 Future directions for bigdata cluster analysis research

An understanding of the effects of the nature of bigdata on existing powerful clustering techniques helps in adapting to it. However, it is also very important to understand the qualities that are desirable in the future clustering algorithms which will seamlessly fit in the bigdata realm. In this section we discuss three future directions for bigdata cluster analysis research. Independent as well as collaborative pursuit in these areas will be required to manage the three Vs of bigdata.

### 6.8.1 Distributed techniques

With the data being collected globally and stored in geographically distributed data centers the need for distributed clustering techniques will be inevitable. Clustering by nature is not a distributed operation so innovative techniques are required to bridge that gap. According to (Aggarwal and Reddy (2013)), the general framework for all distributed clustering techniques can be described using the following four steps.

1. Divide and distribute the data.
2. Local clusters are found at each node.
3. Global clusters are found by gathering all the distributed information.
4. Cluster refinement takes place at each of the local nodes using the knowledge of global clusters.

While distributed techniques have benefits of parallel performance speedup and distributed memory utilization, the communication costs have to be managed. Especially in recent times with the datasets being massive, moving data from one node to another can very expensive. Therefore, a key challenge for distributed clustering algorithms is to find data representations for local clusters which require least memory bandwidth for communication while still maintain high degree of useful information. Dataspace projections used in grid based clustering techniques can be used here towards this end. Network architects will also need to keep the demand for such applications in consideration while designing the next generation data center networks.

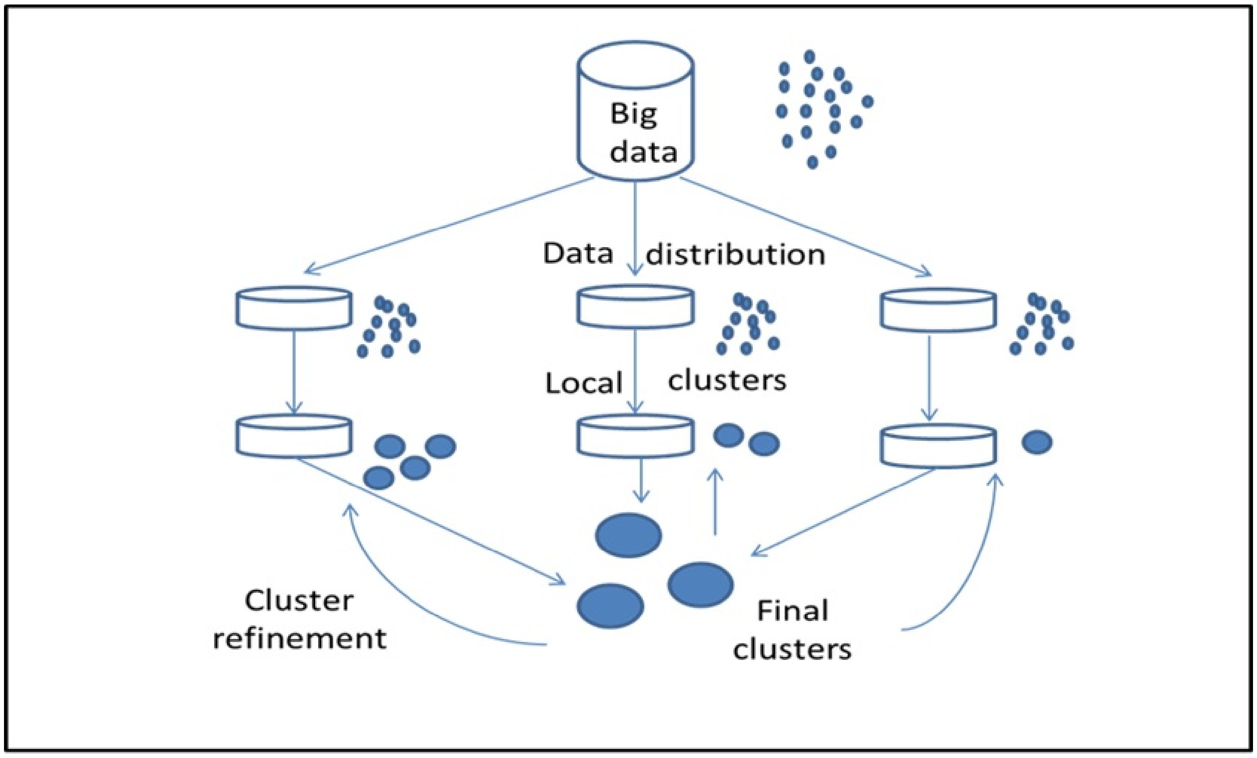


Figure 6.7 Distributed clustering flow

Figure 6.7 (extracted from Somani and Deka (2017)) depicts a distributed clustering flow of data in such class of algorithms. In the following, we discuss further details of three recent distributed clustering algorithms: DBDC (a density based distributed clustering technique), ParMETIS (a graph partitioning based clustering technique) and PKMeans (a MapReduce based K-means type clustering technique). (Aggarwal and Reddy (2013)) describes these and more such algorithms in further detail.

**Density based Distributed Clustering.** DBDC (Januzaj et al. (2004)) as the name suggests uses the density based clustering (more specifically the DBSCAN algorithm) as the key concept. This turns out to be a good choice because data densities are naturally good for superimposition.

Therefore, obtaining the global clustering is very straightforward. Another benefit of using density based clustering in distributed framework is that it allows for very sparse (hence less communication cost) local clustering by slight parameter adjustment if needed.

**Graph Partitioning based Clustering.** ParMETIS (Karypis and Kumar (1999)) is built as a distributed version of graph partitioning METIS (Karypis and Kumar (1998)) algorithm. METIS is one of the popular graph partitioning algorithms. Graph partitioning aims to divide the nodes of a graph into subgraphs with nearly equal sum of node weights, while minimizing the sum of edge weights of boundary edges. METIS achieves this by first using matching techniques to gather strongly connected nodes in groups, followed by partitioning these groups into k subgraphs. This is followed by some final refinements. ParMETIS perform the matching techniques in distributed fashion, and then incrementally achieves a global matching which is then transmitted to every node. The refinements are then done locally for ParMETIS.

**MapReduce based K-means type Clustering.** PKMeans (Zhao et al. (2009)) is a MapReduce based version of the popular K-means algorithm. Parallel performance is achieved by dividing the two steps of K-means among the Mapper and the Reducer part. First step of K-means clustering (partitioning), where each data point needs to be associated with mean is done by the Mapper. The second iterative step where the individual cluster means are computed is done by the Reducer. Since K-means is a partitioning based clustering, it gels well with the MapReduce framework.

### 6.8.2 Robust techniques

More data also means more noise. Therefore, the future clustering techniques will need to be more adept in dealing with noisy and inaccurate data. While the need for robustness in clustering techniques had already been detected, it becomes more relevant in the times of bigdata. The lack of robustness in one of the most famous clustering algorithms K-means had led to the development of its variant called trimmed K-means (Cuesta-Albertos et al. (1997)). (García-Escudero et al. (2010)), (García-Escudero and Gordaliza (1999)) establish that the k-means method has breakdown point of 0. This implies a single outlier data element placed randomly far away can completely disturb the k-means clustering.

Trimming is a useful concept to achieve robustness. Simple trimmed mean cuts a particular ratio of the largest and the smallest observations out before finding the mean of the data. However, for using in clustering we cannot simply trim data from top and bottom. A self-trimming method like the following as shown by (García-Escudero et al. (2010)), (García-Escudero and Gordaliza (1999)) should allow less useful parts of data (outliers) to be removed and not let it influence the clustering:

$$arg_{Y}^{min} {}^{min}_{m_1,\ldots,m_k} \sum_{X_i \in Y} {}^{min}_{j=1,\ldots,k} ||X_i - m_j||^2 \tag{6.3}$$

This is a double minimization problem to find k centers $m_1, \ldots, m_k$. The range of Y will be a subset of the entire sample $X_1, \ldots, X_n$. Such condition adds to robustness of finding K-means clusters and makes is less susceptible to extreme outliers.

Similar robustness while using the knowledge of data distributions and the ideas from sampling theory will need to be adopted into more clustering techniques in the future. This will ensure the maximum extraction of useful information from the data, while minimizing the effects of outliers.

### 6.8.3 Clustering engines

Fundamental machine learning tasks like clustering can be generalized and there can be generic techniques which cluster a variety of data. A sizeable amount of bigdata being heterogeneous in nature calls for development of such clustering engines.

While this research area is in a relatively nascent state, there are certain interesting contributions. In (Ferragina and Gulli (2004)), the authors investigate the web snippet hierarchical clustering problem in a software based perspective and develop an algorithmic solution. They draw snippets from 16 different web-engines and build clusters on-the-fly without any knowledge of any pre-defined clusters. They label the clusters based on sentences structures and content, while using 2 knowledge bases built during preprocessing for selecting sentences and assigning clusters. Their technique arranges clusters into a hierarchy and then allocates them to appropriate nodes based on relevant sentences in them. They also allow overlapping of clusters across various levels of the hierarchy to be flexible in assignments.

Authors in (Carpineto et al. (2009)) provide a detailed discussion on Web clustering engines. They claim that clustering engines successfully overcome many of the challenges of present day search engines by generating clusters of search results as an extra feature. They strongly emphasize that clustering engines are complementary to the search engines. They mention that benefits like subtopic search from a group of search results and fast information summarization make clustering engines highly useful. With the huge variety of data available through web search these days, any tool which helps in arranging and visualizing this information does seem very useful. The views presented by the authors re-enforce our belief that moving forward in the bigdata age, clustering engines will play an important role in the industry.

## 6.9 Conclusion

We have presented the needs, the processes currently utilized for analysis, and future direction for cluster analysis. We discussed the intuition behind clustering methods and the various types of cluster analysis techniques based on their clustering criterion. We analyzed the strengths and weaknesses of various clustering techniques and discussed in detail one popular technique in each category. We also discussed the possible effects of the 3 Vs of bigdata on each of these techniques. We also described a few additional clustering techniques which do not fall into any particular category. Finally, three important future research directions for cluster analysis in the bigdata age were discussed, namely: distributed clustering, robust clustering techniques and clustering engines.

# CHAPTER 7. DESIGNING A PREDICTIVE BIG DATA FRAMEWORK FOR ITEMSET MINING

By studying the various clustering frameworks and effect of the Big Data phenomenon on them, we understand specific effects of the volume, velocity and variety aspects on the knowledge discovery process. It can be argued that the use of massive computing, storage and networking architectures has been the triggering aspect. However we noticed in the previous chapter that the data mining algorithms and model definitions have also either simultaneously evolved or are in dire need of so. In this chapter we will identify certain adaptations required of the itemset mining domain and propose a framework design to bring those into effect.

## 7.1 Introduction

From the study of clustering solutions we notice that understanding the effect of Big Data is a semi-subjective process while also dependent on the application requirements and the algorithm in question. In case of the ROCK clustering algorithm we saw that there is some scope of parallelization to negate the Volume and Velocity aspects, but the Variety can be a challenge due to the link calculation function. Algorithms like DBSCAN are better adept than others to handle Big Data due to availability of random sampling based versions. The powerful K-means algorithm needs to adapt to more complex data types. Gird-based algorithms seems to provide the highest promise to handle all three aspects. Their key strength being the ensemble friendly nature. This allows them to work with different abstractions of data at the same time in the knowledge discovery process. We believe that such capability is the key to negating the Variety aspect of Big Data. Looking at the future directions of ongoing research in Clustering analysis, we similarly notice two key goals. One, to distribute and parallelize computing in a variety of ways. Other, to evolve the models to accommodate the multi-dimensional and noisy nature of Big Data.

Itemset mining faces the similar set of challenges. The problem of scaling the computation is directly related to the counting of itemsets, which is the core computation in itemset mining. There are several techniques proposed to address this issue using distributed computing, sampling techniques and graph based frameworks. We discussed these in Chapter 4, but this is not the problem focus of this work. The goal of this work is to enable the itemset mining process to capture knowledge from multiple dimensions of the data and overall better adapt to the nature of Big Data. The evolution of Frequent Itemset Mining into Utility Itemset Mining was the first crucial step taken in this direction. Moving forward, we postulate that such an itemset mining framework needs to address the following issues:

1. Incorporate the knowledge of cluster structure (which is a key aspect of the transactional data) into the itemset discovery process. We discussed in Chapter 4 about the reasons of doing this, and the summary being preventing information loss. Doing this added a predictive aspect to the problem of itemset mining if compared to current itemset mining framework. However, it is exactly this aspect which can provide the competitive edge in all its applications. Essentially this technique helps identify the most likely itemsets to emerge in future.

2. Provide adaptive predictions based on anticipated cluster contributions. We illustrated this technique in Chapter 5 by defining certain concepts using time series forecasting. Length of the future window of analysis varies from one application/dataset/objective to another, and this technique allows the analyst to tune the itemset mining process accordingly.

3. Based on the analysis goal, application type and the dataset itself, the analyst must have an optimal configuration of the preferred prediction. By configuration here we mean the optimal mix of prediction metrics like accuracy, recall, false positives etc. for any particular analysis. It is very easy to provide such a configuration in form of an objective function of these prediction metrics. A framework which maximizes a specified objective function while performing the itemset mining will be most preferred.

We have already developed techniques to address the first two issues in Chapter 4 and 5 respectively. We will now introduce a technique to address the third issue and the propose a framework design which incorporates all of them.

## 7.2 A predictive framework for itemset mining

The first step towards creating this framework is identifying a way to maximize an objective function composed of prediction metrics while performing the itemset mining operation. We posit that if we analyze this problem from the point of view of action prior to the mining process, we get one step closer to achieving a solution. The predictive itemset mining model which we have proposed in previous chapters works because we inflate the utilities of certain items both selectively (affinity to clusters) and adaptively (anticipated contributions of clusters). Therefore, we perform this inflation on basis of two different abstractions of the dataset. The choice of inflation function (for example the "con", "mod" and "agg" definitions used in Chapter 4) exercised by us till now has been guided by a simple decision of aggressiveness/conservativeness in making predictions. However, if for example the optimal predictions desired by the analyst might be of the following configuration: a maximal score composed of 70% weight to number of predictions and 30% negative weight to inaccurate predictions. In such a scenario, its impossible to eye-ball and pick an inflation function which might be most suitable. Moreover, for different prediction instances on the same dataset, the choice of inflation function is not guaranteed to be the same. Therefore, we will need to make such a choice on case by case basis.

Based on the above discussion, we conclude that we must learn to make these choices for any given objective function for any given dataset. The primary information which can help us in making this choice are the anticipated cluster contributions for the predicting window. It should be noted that by cluster contribution, we mean the concept defined by us in Chapter 5. A secondary information which can also help is a measure of the anticipated rate of increase of cluster contributions. While an exact calculation for this measure can be tedious, anticipated cluster contributions until half of the prediction window along with those for the entire prediction window together can provide a rough estimate for the rate. Now for each of this set of cluster contributions (including both for full window and half window), among all of the choices for inflation, there must be one choice which maximizes the objective function. Let us suppose that we have knowledge of several instances of set of cluster contributions along with their best choice of inflation function, we can start to learn how to make such a choice. Therefore it would translate to the classical machine learning problem of classification. Based on this rationale we now formally define an "Inflation Choice Module". We will later use this module as one of the building blocks for our itemset mining framework.

### 7.2.1 Inflation Choice Module

As mentioned above core action of this Inflation Choice Module (ICM) would be solving a classification problem. The set of features in solving this problem will be:

$$\text{set of features} = (CC_1, \ldots, CC_k, CC_1^{'}, \ldots, CC_k^{'}) \tag{7.1}$$

Where $k$ is the number of clusters identified in the dataset, $CC_1, \ldots, CC_k$ are the anticipated clustered contributions (normalized) in the choice of prediction window and $CC_1^{'}, \ldots, CC_k^{'}$ are the anticipated clustered contributions (normalized) until the half of prediction window.

We normalize the cluster contributions in order to make the feature space uniform for all cases.

$$\text{set of labels} = \text{set of inflation actions} \tag{7.2}$$

Our first choice of machine learning model to solve this problem is a tree based random forest model. The feature space for this problem has no guarantee for linearity and will most probably need implicit feature selection, which is why we prefer a tree based model. The choice to use a random forest model of trees will be key to avoid any bias in the model. The random forest model uses a technique called "bagging" which is essentially creating an ensemble model of many trees and performing a voting operation to choose among the label recommendation from all sub-models. This operates on the principal of variance-bias trade-off where by increase the variability in the model we reduce the risk of bias. Because in our ICM module we want a generalizable model which can operate on variety of unseen data, reducing the bias as much as possible is crucial for us. But as with any machine learning exercise, it is always wise to try different models to make sure the choice is the right one.

### 7.2.2 Framework design

With a solution to all the three issues previously mentioned, we will now present the design for our itemset mining framework. Please refer to Figure 7.1. The framework assumes a predictive pipeline, where there is live data for which predictions are made specified prediction windows. A good quality and nicely balanced training dataset is assumed. The first step then is to identify an approximate cluster structure using the training data. We have presented how to perform this in Chapter 3. The amount of data used in the developing the clusters and the algorithm parameters are subjective to the user. However, a good approximation of the real cluster will greatly help in keeping the predictions accurate.

The second step is to train the classifier in ICM. For this we create several instances of labeled data by evaluating the objective function for all possible elements in the set of inflation actions for a subset of the training data and an associated prediction window. For each instance of data point creation, we evaluate the anticipated cluster contributions for both the full prediction window and half of that. The normalized values for these act as the features, and the inflation action which provides the highest objective function is chosen as the label. For calculating the objective function scores for any inflation action, we execute that action and get the predictions, which in turn provide the corresponding prediction metrics. They are directly plugged into the objective function. We suggest on deciding a fixed prediction window length, and keeping it uniform while selecting different parts of the training data for creating features and labels. This must be the same prediction window length which is eventually intended to be used for making prediction on live data. This ensures the model to be transfer its learning as accurately as possible.

Once the initial training steps are complete, the framework is ready to perform predictions on live data. For any collection of transactions, in order to make itemset predictions we perform the following steps:

1. Decide the prediction window length.
2. Use the cluster structure to identify the anticipated cluster contributions for this window (both full and half length), and normalize them.
3. Use the classifier in the ICM to predict the best inflation action (one which maximizes the objective function) in this scenario.
4. Accordingly assign utilities in the predictive itemset mining model.
5. Mine the itemsets.

### 7.2.3 Framework implementation

In this section we will demonstrate an implementation instance of our framework design, and apply it to make itemset predictions on a Web Click stream dataset (BMSWebView2 (2018)). The dataset contains 77,512 transactions (sequence of web clicks). We assign the external utilities (between 1-50) using a uniform random number generator, in a similar way as used in Chapters 4, 5 and 6. As mentioned before, this is common practice while using benchmark datasets in testing itemset mining techniques. Due to the big size of the dataset, we use only 20% of the data in our analysis. We use half of the selected data to train our classifier and the other half for testing.

Firstly, we generate and store a cluster structure using 20% randomly sampled transactions from the training data. We do this by using our clustering algorithm detailed in the chapter 3, and get a 3 cluster structure. The $min_{aff}$ and $min_{uty}$ parameters used in the clustering algorithm are 0.2 and 0.1 respectively. We then randomly select a 3,000 transactions chunk from the training data, to create a learning instance for the classifier. First 2,000 transactions are used to generate features based on forecasts for the next 1,000 tractions (prediction window) as discussed in the

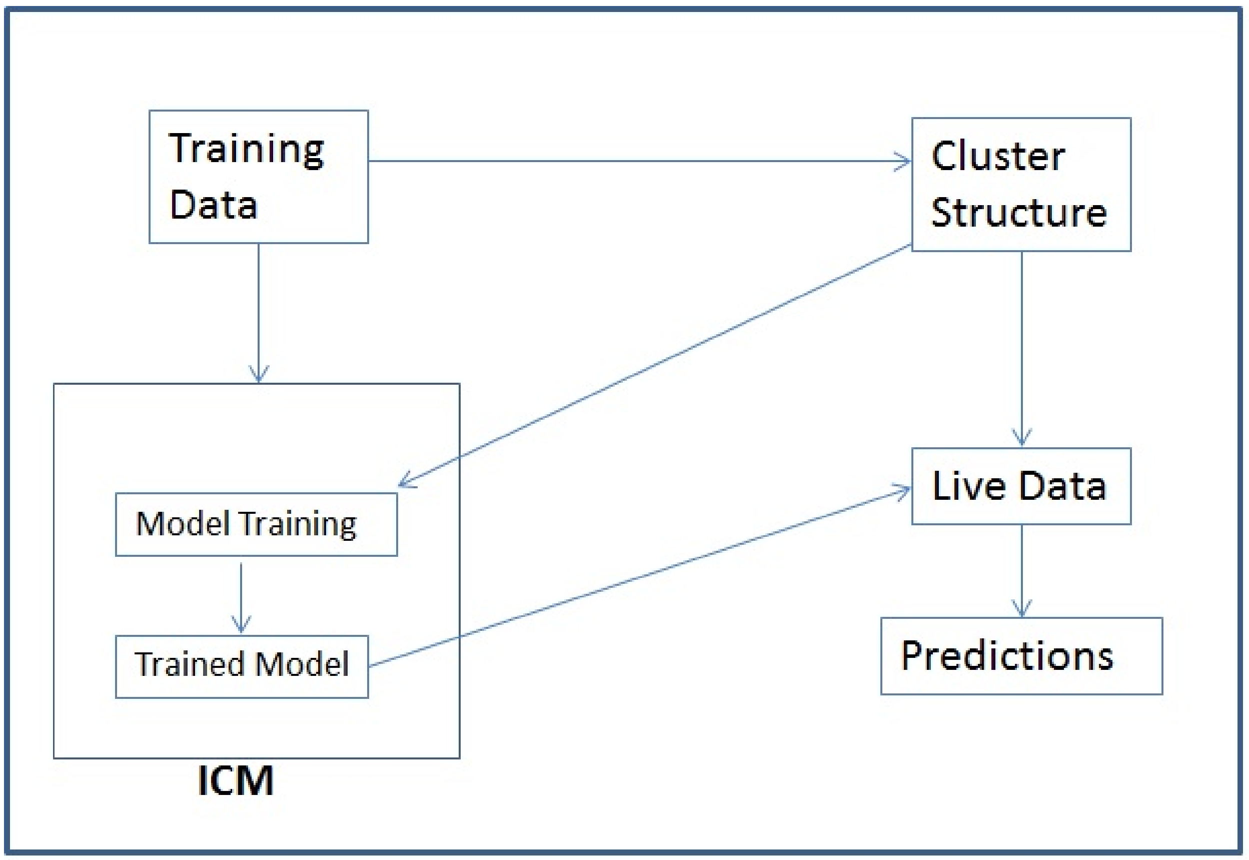


Figure 7.1 Framework design for itemset mining

ICM description above. For forecasting, the Seabold and Perktold (2010) library is used to fit an ARIMA model, as shown in Chapter 5. Two custom prediction metrics are also defined:

$$\text{metric 1} = 0.7 * (\text{number of predictions}) - 0.3 * (\text{inaccurate predictions}) \tag{7.3}$$

$$\text{metric 2} = 0.3 * (\text{number of predictions}) - 0.7 * (\text{inaccurate predictions}) \tag{7.4}$$

Where number of predictions are the additional itemsets which the predictive model generated in comparison to the conventional model. Inaccurate prediction is the number of inaccuracies in the predicted itemsets. With the two custom prediction criteria defined, we then identify best inflation action (label) for each of the two criteria. The threshold for the itemset mining problem is set to $\phi = 1500$ and the following three inflation actions are defined in order to have a manageable number of predictions:

$$\text{conservative } k = \{^{1 \text{ if affinity}(T_i, C_j) < 0.01}_{3 \text{ otherwise}} \tag{7.5}$$

$$\text{moderate } k = 3 \tag{7.6}$$

$$\text{aggressive } k = \{^{3 \text{ if affinity}(T_i, C_j) < 0.01}_{5 \text{ otherwise}} \tag{7.7}$$

Please note that the "k" mentioned above acts as the scaling (inflating) factor to the transactional utilities as defined previously in Chapter 4. Rationale behind the set of parameters used in the inflation actions above is to obtain a diverse yet manageable number of predictions. Prediction score is computed for each of the two metrics for each choice above. Inflation action with the highest prediction criterion score is chosen as the respective label for that metric. This experiment is repeated 100 times to get enough learning instances for each prediction criterion. We train random forest classifiers using the training data features and labels for each of the two custom predictions criteria. It is one of the most robust classification techniques which uses bagging to overcome bias in the training data. This technique is chosen since it can work well with a variety of data. The Scikit-learn library (Pedregosa et al. (2011)) is used to instantiate the random forest classifier. 20 estimators with a depth of 2 are used, since the learning data is not that huge. Once the classifiers are trained, our entire framework is ready to perform customized itemset predictions.

In order to test the predictive power of our framework, we use the other half of the data which has not been used in the model learning. We randomly select 5 sub-divisions of 3000 transactions from this half of the data. For each sub-division we create the set of features and labels in the same way as done in the training phase for both the prediction criteria. This gives us 10 data points to validate the accuracy of our predictive framework. We calculate the accuracy of our framework based on this test data, by using the respective random forest classifier. It should be noted that the final prediction of our framework is 'itemsets found', which does not have a traditional baseline in terms of accuracy. But since in the final step we use a classifier, we define a baseline of 33.33% accuracy in case of this experiment since there are 3 inflation choices (labels) to choose from. The rationale behind the 33.33% baseline is that if we randomly pick any one of the three inflation actions, the probability that it is the optimal choice is 1/3. This is based on the assumption that the labels are from independent identically distributed (i.i.d.) variables. We repeat the above steps 40 times (20+20 for each custom prediction criteria), giving us 40 accuracy scores for our framework.

Our predictive framework performed with an accuracy of “0.895” with a 95% confidence interval of (0.844, 0.945) against a random baseline of “0.33”. This shows that the framework makes intelligent predictions in terms of the inflation function choice (label). Inflation labels have one-to-one correspondence with the following action: the degree of aggressiveness in making predictions. Higher the degree, more the number of predictions we make, but also more likelihood of having inaccurate predictions. Their choice is important because based on the most preferred trade off (chosen by the analyst) between number of predictions and inaccuracies, there is always an optimal choice of inflation label. But that can vary based on cluster structure, part of the data we are working with and the trends in that part of data. Hence, the framework decides which is the most optimal label choice by using a trained classifier and the cluster structure. The accuracy of “0.895” is that of how accurate the framework is, in making the right choice. It is crucial to the data and analysis, because it tells how likely the analyst will get the predictions aligned with their preference (of trade off). It should be noted that the key contribution of this framework design is identifying that prediction of itemsets can be customized by inserting a trained classifier before generating predictions using the cluster structure.

## 7.3 Conclusion

In this chapter we proposed the design of a framework to perform itemset mining, while addressing the future data analytics challenges in this domain. After performing a study of the broad field of cluster analysis and its evolution in the time of Big Data in the previous chapter, we similarly identify the path of evolution for itemset mining. We find that besides scaling of computation, the itemset mining model needs to adapt to current trends in analytics. We identified in Chapter 6 that to negate the effect of Variety aspect of Big Data we need to concurrently work with different abstractions of data. Implicitly using the cluster structure in itemset mining process to predict future itemsets in Chapter 4 was a step in this direction. The second step in this direction was to capture the trends of cluster contributions to transactions and connect them to the mining process in Chapter 5. Lastly, in this chapter we take a third step in this direction. Based on the analysis goal, application type and the dataset itself, each analyst has an optimal configuration (a custom combination of various predictive metrics) of predictions. We demonstrate that it is possible to align itemset mining with such an optimal configuration, by transforming it into a classification problem. We propose an itemset mining framework which incorporates all the above three steps in making itemset predictions. We create an implementation of the framework on a Web analytics data set, and notice that it successfully makes optimal prediction configuration choices with a high accuracy of “0.895”. The accuracy of this choice is important because it is what aligns the itemset predictions with the optimal configuration chosen by the analyst as referred above. Making the choice is difficult because it can change based on cluster structure, preference of the analyst, part of the data we are working with and the trends in that part of data. The framework proposed achieves it by modeling this into a classification problem and then using the classifiers along with the cluster structure to make itemset predictions.

# CHAPTER 8. CONCLUSION

In this work we have striven to advance two key knowledge discovery techniques for the transactional data model: cluster analysis and itemset mining. Transactional data model is one of the widely used quantitative models in a variety of analytics application domains. These two techniques find widespread use in a variety of applications because of their simple and intuitive abstractions. We propose a novel clustering algorithm for transactional data which captures the utility information. We also propose two validation criterion for the obtained cluster structure based on how accurately it captures the high utility patterns in the data. When comparing our technique with competing algorithms, we miss much fewer high utility patterns of importance than them.

We establish that the current Utility Itemset Mining (UIM) problem model can be extended by adding a key modeling capability of prediction by capturing cluster specific patterns in the dataset. All transactions possess information in them regarding the degree to which they belong to a cluster of similar objects from the entire data. If a transaction is fairly representative of a cluster type then the information in it is more characteristic of the cluster type than the entire data. Therefore ignoring this knowledge and subjecting these transactions to the common threshold in the UIM problem leads to information loss. We identify that an implicit use of cluster structure of data in the UIM problem model will address this limitation. We do this by introducing a new clustering based utility in the definition of the UIM problem model and modifying the definitions of absolute utilities based on it. This modified predictive UIM problem model enables the cluster specific patterns to emerge while still mining the inter-cluster patterns. It can integrate into all UIM techniques as well. We show that our model makes accurate predictions successfully, by discovering 30-40% future itemsets in most experiments on two benchmark datasets with negligible inaccuracies. There are no other present itemset prediction models, so accurate prediction is an accomplishment of ours. We then identify that there is a need of adaptive anticipation in our predictive Utility Itemset Mining model. We develop and define concepts which can capture the idea of adaptive anticipation of the cluster structure by using time series forecasting techniques. We then illustrate how to integrate these concepts into the predictive UIM model.

In order to understand the future challenges of knowledge discovery, we then perform a study on effect of Big Data phenomenon on the broad field of cluster analysis. Specifically, we study each different type of clustering rationale along with an example of a popular technique based on that rationale. We discuss the effect of three V's (Volume, Velocity and Variety) of Big Data on each of these techniques. We also observe the ongoing future directions of research for clustering analysis.

Performing the study of the field of cluster analysis and its evolution in the time of Big Data, helped us identify the challenges in evolution for the itemset mining model. We propose the design of a framework to perform itemset mining while addressing these challenges. In this framework we use the knowledge of cluster structure both selectively(as discussed in Chapter 4) and adaptively (as discussed in Chapter 5). Furthermore we identified that based on analysis goal, application type and the dataset itself, each analyst has an optimal configuration (a custom combination of various predictive metrics) of predictions. Therefore the itemset mining analysis must have an option to be aligned with such a configuration. We show that it is possible to learn to achieve this, by transforming this problem into a classification problem. Therefore, In proposing our final itemset mining framework we incorporate all our knowledge discovery solutions regarding itemset predictions. We create an implementation of the framework on a Web analytics data set, and notice that it successfully makes optimal prediction configuration choices with a high accuracy of "0.895" against a random baseline of "0.33". This showed that the framework makes intelligent predictions in terms of the inflation function choice. The accuracy of this choice is important because it is what aligns the itemset predictions with the optimal configuration referred above. The key contribution of the framework design is modeling this as a classification problem and then using the classifiers along with the cluster structure to make itemset predictions.

Throughout this work we focused on knowledge discovery through the idea of repeating patterns either in discovery of clusters of transactions or high utility itemsets. Knowledge of repeating patterns provides the useful associations in a dataset. For the future direction of this work, we would like to incorporate the idea of causation into our knowledge discovery solutions. Searching for causation will require modifying the search criterion, and identifying concepts which can realistically capture causation. This will help in bridging the gap between knowledge and actionable knowledge.

# Bibliography

(1987). Uci machine learning repository. archive.ics.uci.edu/ml/datasets/Soybean+(Small). Accessed: 2016-09-01.

(2008). Bkplot implementation. cecs.wright.edu/ keke.chen/. Accessed: 2016-09-01.

(2015). K-modes implementation. https://github.com/nicodv/kmodes. Accessed: 2016-09-01.

Aggarwal, C. C. and Reddy, C. K. (2013). *Data clustering: algorithms and applications*. CRC press.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM.

Agrawal, R. and Shafer, J. C. (1996). Parallel mining of association rules. *IEEE Transactions on Knowledge & Data Engineering*, (6):962–969.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

Ahmed, C. F., Tanbeer, S. K., Jeong, B.-S., and Lee, Y.-K. (2009). Efficient tree structures for high utility pattern mining in incremental databases. *Knowledge and Data Engineering, IEEE Transactions on*, 21(12):1708–1721.

Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and Sridharan, M. (2010). Data center tcp (dctcp). In *ACM SIGCOMM computer communication review*, volume 40, pages 63–74. ACM.

Alves, R., Rodriguez-Baena, D. S., and Aguilar-Ruiz, J. S. (2009). Gene association analysis: a survey of frequent pattern mining from gene expression data. *Briefings in Bioinformatics*, page bbp042.

Andreopoulos, B., An, A., Wang, X., and Schroeder, M. (2009). A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3):297–314.

Andritsos, P., Tsaparas, P., Miller, R. J., and Sevcik, K. C. (2004). Limbo: Scalable clustering of categorical data. In *International Conference on Extending Database Technology*, pages 123–146. Springer.

Bai, L., Liang, J., Dang, C., and Cao, F. (2013). The impact of cluster representatives on the convergence of the k-modes type clustering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1509–1522.

Barbará, D., Li, Y., and Couto, J. (2002). Coolcat: an entropy-based algorithm for categorical clustering. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 582–589. ACM.

Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer.

BMSWebView1 (2016). Smpf: An open-source data mining library, accessed: 2016-06-14. http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php.

BMSWebView2 (2018). Smpf: An open-source data mining library, accessed: 2018-09-10. http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php.

Borah, B. and Bhattacharyya, D. (2004). An improved sampling-based dbscan for large spatial databases. In *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*, pages 92–96. IEEE.

Bradley, P. S. and Fayyad, U. M. (1998). Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer.

Brewer, E. A. (2000). Towards robust distributed systems. In *PODC*, volume 7.

Brijs, T., Swinnen, G., Vanhoof, K., and Wets, G. (1999a). Using association rules for product assortment decisions: A case study. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 254–260. ACM.

Brijs, T., Swinnen, G., Vanhoof, K., and Wets, G. (1999b). Using association rules for product assortment decisions: A case study. In *Knowledge Discovery and Data Mining*, pages 254–260.

Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD Record*, volume 26, pages 255–264. ACM.

Brown, D. E. and Huntley, C. L. (1992). A practical application of simulated annealing to clustering. *Pattern recognition*, 25(4):401–412.

Carpineto, C., Osiński, S., Romano, G., and Weiss, D. (2009). A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17.

CERN (2015). Cern website. http://home.cern/about/computing.

Chan, R. C., Yang, Q., and Shen, Y.-D. (2003). Mining high utility itemsets. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 19–26. IEEE.

Chen, K. and Liu, L. (2005). The" best k" for entropy-based categorical data clustering.

Chen, X., Huang, J. Z., and Luo, J. (2016). Purtreeclust: A purchase tree clustering algorithm for large-scale customer transaction data. In *32nd IEEE International Conference on Data Engineering*, pages 661–672. IEEE.

Cuesta-Albertos, J. A., Gordaliza, A., Matrán, C., et al. (1997). Trimmed $k$-means: An attempt to robustify quantizers. *The Annals of Statistics*, 25(2):553–576.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Ferragina, P. and Gulli, A. (2004). The anatomy of a hierarchical clustering engine for web-page, news and book snippets. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 395–398. IEEE.

Ganti, V., Gehrke, J., and Ramakrishnan, R. (1999). Cactusclustering categorical data using summaries. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 73–83. ACM.

García-Escudero, L. Á. and Gordaliza, A. (1999). Robustness properties of k means and trimmed k means. *Journal of the American Statistical Association*, 94(447):956–969.

García-Escudero, L. A., Gordaliza, A., Matrán, C., and Mayo-Iscar, A. (2010). A review of robust clustering methods. *Advances in Data Analysis and Classification*, 4(2-3):89–109.

Gibson, D., Kleinberg, J., and Raghavan, P. (1998). Clustering categorical data: An approach based on dynamical systems. *Databases*, 1:75.

Guha, S., Rastogi, R., and Shim, K. (1999). Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521. IEEE.

Hall, L. O., Ozyurt, I. B., Bezdek, J. C., et al. (1999). Clustering with a genetically optimized approach. *IEEE Transactions on evolutionary Computation*, 3(2):103–112.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304.

Januzaj, E., Kriegel, H.-P., and Pfeifle, M. (2004). Dbdc: Density based distributed clustering. In *International Conference on Extending Database Technology*, pages 88–105. Springer.

Kambatla, K., Kollias, G., Kumar, V., and Grama, A. (2014). Trends in big data analytics. *Journal of Parallel and Distributed Computing*, 74(7):2561–2573.

Karypis, G. and Kumar, V. (1998). Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129.

Karypis, G. and Kumar, V. (1999). Parallel multilevel series k-way partitioning scheme for irregular graphs. *Siam Review*, 41(2):278–300.

Kriegel, H.-P., Borgwardt, K. M., Kröger, P., Pryakhin, A., Schubert, M., and Zimek, A. (2007). Future trends in data mining. *Data Mining and Knowledge Discovery*, 15(1):87–97.

Lakhawat, P., Mishra, M., and Somani, A. K. (2016). A novel clustering algorithm to capture utility information in transactional data. In *KDIR*, pages 456–462.

Lakhawat, P., Mishra, M., and Somani, A. K. (2017). A clustering based prediction scheme for high utility itemsets. In *KDIR*, pages 123–134.

Laney, D. (2001). 3d data management: Controlling data volume, velocity and variety. *META group research note*, 6(70):1.

Li, H.-F., Huang, H.-Y., Chen, Y.-C., Liu, Y.-J., and Lee, S.-Y. (2008). Fast and memory efficient mining of high utility itemsets in data streams. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 881–886. IEEE.

Li, T., Ma, S., and Ogihara, M. (2004). Entropy-based criterion in categorical clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 68. ACM.

Liao, S.-H., Chu, P.-H., and Hsiao, P.-Y. (2012). Data mining techniques and applications–a decade review from 2000 to 2011. *Expert Systems with Applications*, 39(12):11303–11311.

Liu, B., Xia, Y., and Yu, P. S. (2000). Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 20–29. ACM.

Liu, Y., Liao, W.-k., and Choudhary, A. (2005a). A fast high utility itemsets mining algorithm. In *Proceedings of the 1st international workshop on Utility-based data mining*, pages 90–99. ACM.

Liu, Y., Liao, W.-k., and Choudhary, A. (2005b). A two-phase algorithm for fast discovery of high utility itemsets. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 689–695. Springer.

Luenberger, D. G. (1973). Introduction to linear and nonlinear programming.

Mehlhorn, K. (2013). *Data structures and algorithms 1: Sorting and searching*, volume 1. Springer Science & Business Media.

Milligan, G. W. (1981). A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199.

Naulaerts, S., Meysman, P., Bittremieux, W., Vu, T. N., Berghe, W. V., Goethals, B., and Laukens, K. (2015). A primer to frequent itemset mining for bioinformatics. *Briefings in bioinformatics*, 16(2):216–231.

Ngai, E. W., Xiu, L., and Chau, D. C. (2009). Application of data mining techniques in customer relationship management: A literature review and classification. *Expert systems with applications*, 36(2):2592–2602.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Qian, Y., Li, F., Liang, J., Liu, B., and Dang, C. (2015). Space structure and clustering of categorical data.

RetailDataset (2016). Frequent itemset mining dataset repository, accessed: 2016-06-14. http://fimi.ua.ac.be/data/.

Saha, I. and Maulik, U. (2014). Incremental learning based multiobjective fuzzy clustering for categorical data. *Information Sciences*, 267:35–57.

Seabold, S. and Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Shah, M., Ranganathan, P., Chang, J., Tolia, N., Roberts, D., and Mudge, T. (2010). Data dwarfs: Motivating a coverage set for future large data center workloads. In *Proc. Workshop Architectural Concerns in Large Datacenters*. Citeseer.

Somani, A. K. and Deka, G. C. (2017). *Big Data Analytics: Tools and Technology for Effective Planning*. Chapman & Hall/CRC.

Sun, H., Chen, R., Jin, S., and Qin, Y. (2015). A hierarchical clustering for categorical data based on holo-entropy. In *2015 12th Web Information System and Application Conference (WISA)*, pages 269–274. IEEE.

Tishby, N. and Slonim, N. (2000). Data clustering by markovian relaxation and the information bottleneck method. In *NIPS*, pages 640–646. Citeseer.

Toivonen, H. et al. (1996). Sampling large databases for association rules. In *VLDB*, volume 96, pages 134–145.

Tsai, H.-H. (2012). Global data mining: An empirical study of current trends, future forecasts and technology diffusions. *Expert systems with applications*, 39(9):8172–8181.

Tseng, V. S., Wu, C.-W., Fournier-Viger, P., and Yu, P. S. (2015). Efficient algorithms for mining the concise and lossless representation of high utility itemsets. *Knowledge and Data Engineering, IEEE Transactions on*, 27(3):726–739.

Tseng, V. S., Wu, C.-W., Shie, B.-E., and Yu, P. S. (2010). Up-growth: an efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 253–262. ACM.

Tung, A. K., Han, J., Lakshmanan, L. V., and Ng, R. T. (2001). Constraint-based clustering in large databases. In *International Conference on Database Theory*, pages 405–419. Springer.

Van der Merwe, D. and Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 215–220. IEEE.

Wang, K., Xu, C., and Liu, B. (1999). Clustering transactions using large items. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 483–490. ACM.

WSDaily (2012). Wall street daily. http://www.wallstreetdaily.com/2012/03/21/forever-growth-trends-and-five-stocks-set-to-profit/.

Xiong, T., Wang, S., Mayers, A., and Monga, E. (2012). Dhcc: Divisive hierarchical clustering of categorical data. *Data Mining and Knowledge Discovery*, 24(1):103–135.

Yan, H., Chen, K., Liu, L., and Yi, Z. (2010). Scale: a scalable framework for efficiently clustering transactional data. *Data mining and knowledge Discovery*, 20(1):1–27.

Yan, H., Zhang, L., and Zhang, Y. (2005). Clustering categorical data using coverage density. In *International Conference on Advanced Data Mining and Applications*, pages 248–255. Springer.

Yang, Y., Guan, X., and You, J. (2002). Clope: a fast and effective clustering algorithm for transactional data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 682–687. ACM.

Zaïane, O. R., Foss, A., Lee, C.-H., and Wang, W. (2002). On data clustering analysis: Scalability, constraints, and validation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 28–39. Springer.

Zaki, M. J. (2000). Scalable algorithms for association mining. *Knowledge and Data Engineering, IEEE Transactions on*, 12(3):372–390.

Zhao, W., Ma, H., and He, Q. (2009). Parallel k-means clustering based on mapreduce. In *IEEE International Conference on Cloud Computing*, pages 674–679. Springer.

# APPENDIX. EXPERIMENTAL RESULTS

Results for *Phi* = 40k and 25%, 50% and 75% of data: According to current model: HUI in100% data = 158, HUI in 25% data = 27, HUI in 50% data = 58, HUI in 75% data = 92 According to predictive UIM model:

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Cluster utility criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.15 | 0.1 | con | 35, 78, 125 | 8, 20, 33 | 0, 0, 2 |
| 0.01 | 0.15 | 0.15 | con | 35, 77, 125 | 8, 19, 33 | 0, 0, 2 |
| 0.01 | 0.15 | 0.2 | con | 35, 78, 125 | 8, 20, 33 | 0, 0, 1 |
| 0.01 | 0.2 | 0.1 | con | 35, 78, 124 | 8, 20, 32 | 0, 0, 1 |
| 0.01 | 0.2 | 0.15 | con | 35, 78, 126 | 8, 20, 34 | 0, 0, 1 |
| 0.01 | 0.2 | 0.2 | con | 35, 78, 127 | 8, 20, 35 | 0, 0, 3 |
| 0.01 | 0.25 | 0.1 | con | 35, 78, 127 | 8, 20, 35 | 0, 0, 2 |
| 0.01 | 0.25 | 0.15 | con | 35, 78, 125 | 8, 20, 33 | 0, 0, 1 |
| 0.01 | 0.25 | 0.2 | con | 35, 78, 127 | 8, 20, 35 | 0, 0, 2 |
| 0.01 | 0.3 | 0.1 | con | 35, 78, 127 | 8, 20, 35 | 0, 0, 3 |
| 0.01 | 0.3 | 0.15 | con | 35, 78, 126 | 8, 20, 34 | 0, 0, 3 |
| 0.01 | 0.3 | 0.2 | con | 35, 78, 126 | 8, 20, 34 | 0, 0, 2 |
| 0.05 | 0.15 | 0.1 | con | 35, 83, 146 | 8, 25, 54 | 0, 0, 10 |
| 0.05 | 0.15 | 0.15 | con | 35, 83, 146 | 8, 25, 54 | 0, 0, 9 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Cluster utility criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.05 | 0.15 | 0.2 | con | 35, 81, 142 | 8, 23, 50 | 0, 0, 7 |
| 0.05 | 0.2 | 0.1 | con | 35, 82, 146 | 8, 24, 54 | 0, 0, 9 |
| 0.05 | 0.2 | 0.15 | con | 35, 82, 146 | 8, 24, 54 | 0, 0, 10 |
| 0.05 | 0.2 | 0.2 | con | 35, 84, 147 | 8, 26, 55 | 0, 0, 11 |
| 0.05 | 0.25 | 0.1 | con | 35, 82, 146 | 8, 24, 54 | 0, 0, 10 |
| 0.05 | 0.25 | 0.15 | con | 35, 83, 146 | 8, 25, 54 | 0, 0, 10 |
| 0.05 | 0.25 | 0.2 | con | 35, 85, 147 | 8, 27, 55 | 0, 0, 10 |
| 0.05 | 0.3 | 0.1 | con | 35, 82, 148 | 8, 24, 56 | 0, 0, 11 |
| 0.05 | 0.3 | 0.15 | con | 35, 82, 147 | 8, 24, 55 | 0, 0, 10 |
| 0.05 | 0.3 | 0.2 | con | 35, 83, 148 | 8, 25, 56 | 0, 0, 11 |
| 0.1 | 0.15 | 0.1 | con | 36, 89, 154 | 9, 31, 62 | 0, 0, 13 |
| 0.1 | 0.15 | 0.15 | con | 36, 90, 153 | 9, 32, 61 | 0, 0, 13 |
| 0.1 | 0.15 | 0.2 | con | 35, 89, 154 | 8, 31, 62 | 0, 0, 13 |
| 0.1 | 0.2 | 0.1 | con | 36, 89, 153 | 9, 31, 61 | 0, 0, 13 |
| 0.1 | 0.2 | 0.15 | con | 36, 90, 153 | 9, 32, 61 | 0, 0, 13 |
| 0.1 | 0.2 | 0.2 | con | 35, 89, 153 | 8, 31, 61 | 0, 0, 13 |
| 0.1 | 0.25 | 0.1 | con | 35, 90, 155 | 8, 32, 63 | 0, 0, 14 |
| 0.1 | 0.25 | 0.15 | con | 35, 90, 157 | 8, 32, 65 | 0, 0, 14 |
| 0.1 | 0.25 | 0.2 | con | 35, 90, 153 | 8, 32, 61 | 0, 0, 13 |
| 0.1 | 0.3 | 0.1 | con | 35, 90, 157 | 8, 32, 65 | 0, 0, 15 |
| 0.1 | 0.3 | 0.15 | con | 35, 89, 157 | 8, 31, 65 | 0, 0, 15 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Cluster utility criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.1 | 0.3 | 0.2 | con | 35, 90, 157 | 8, 32, 65 | 0, 0, 14 |
| 0.01 | 0.15 | 0.1 | mod | 35, 81, 143 | 8, 23, 51 | 0, 0, 8 |
| 0.01 | 0.15 | 0.15 | mod | 35, 81, 140 | 8, 23, 48 | 0, 0, 6 |
| 0.01 | 0.15 | 0.2 | mod | 35, 83, 142 | 8, 25, 50 | 0, 0, 7 |
| 0.01 | 0.2 | 0.1 | mod | 35, 82, 141 | 8, 24, 49 | 0, 0, 6 |
| 0.01 | 0.2 | 0.15 | mod | 35, 81, 140 | 8, 23, 48 | 0, 0, 6 |
| 0.01 | 0.2 | 0.2 | mod | 35, 82, 143 | 8, 24, 51 | 0, 0, 7 |
| 0.01 | 0.25 | 0.1 | mod | 35, 83, 143 | 8, 25, 51 | 0, 0, 7 |
| 0.01 | 0.25 | 0.15 | mod | 35, 81, 140 | 8, 23, 48 | 0, 0, 6 |
| 0.01 | 0.25 | 0.2 | mod | 35, 82, 144 | 8, 24, 52 | 0, 0, 9 |
| 0.01 | 0.3 | 0.1 | mod | 35, 81, 142 | 8, 23, 50 | 0, 0, 7 |
| 0.01 | 0.3 | 0.15 | mod | 35, 82, 143 | 8, 24, 51 | 0, 0, 7 |
| 0.01 | 0.3 | 0.2 | mod | 35, 82, 142 | 8, 24, 50 | 0, 0, 7 |
| 0.05 | 0.15 | 0.1 | mod | 35, 89, 154 | 8, 31, 62 | 0, 0, 11 |
| 0.05 | 0.15 | 0.15 | mod | 35, 89, 155 | 8, 31, 63 | 0, 0, 13 |
| 0.05 | 0.15 | 0.2 | mod | 35, 89, 154 | 8, 31, 62 | 0, 0, 12 |
| 0.05 | 0.2 | 0.1 | mod | 35, 89, 155 | 8, 31, 63 | 0, 0, 14 |
| 0.05 | 0.2 | 0.15 | mod | 35, 90, 156 | 8, 32, 64 | 0, 0, 14 |
| 0.05 | 0.2 | 0.2 | mod | 36, 89, 156 | 9, 31, 64 | 0, 0, 13 |
| 0.05 | 0.25 | 0.1 | mod | 35, 89, 157 | 8, 31, 65 | 0, 0, 14 |
| 0.05 | 0.25 | 0.15 | mod | 35, 89, 157 | 8, 31, 65 | 0, 0, 13 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Cluster utility criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.05 | 0.25 | 0.2 | mod | 35, 89, 156 | 8, 31, 64 | 0, 0, 14 |
| 0.05 | 0.3 | 0.1 | mod | 35, 89, 156 | 8, 31, 64 | 0, 0, 13 |
| 0.05 | 0.3 | 0.15 | mod | 36, 89, 158 | 9, 31, 66 | 0, 0, 14 |
| 0.05 | 0.3 | 0.2 | mod | 35, 89, 157 | 8, 31, 65 | 0, 0, 13 |
| 0.1 | 0.15 | 0.1 | mod | 37, 91, 166 | 10, 33, 74 | 0, 0, 21 |
| 0.1 | 0.15 | 0.15 | mod | 37, 91, 164 | 10, 33, 72 | 0, 0, 19 |
| 0.1 | 0.15 | 0.2 | mod | 36, 91, 161 | 9, 33, 69 | 0, 0, 16 |
| 0.1 | 0.2 | 0.1 | mod | 38, 92, 167 | 11, 34, 75 | 0, 0, 21 |
| 0.1 | 0.2 | 0.15 | mod | 37, 91, 165 | 10, 33, 73 | 0, 0, 19 |
| 0.1 | 0.2 | 0.2 | mod | 36, 91, 165 | 9, 33, 73 | 0, 0, 19 |
| 0.1 | 0.25 | 0.1 | mod | 36, 91, 165 | 9, 33, 73 | 0, 0, 19 |
| 0.1 | 0.25 | 0.15 | mod | 36, 92, 166 | 9, 34, 74 | 0, 0, 20 |
| 0.1 | 0.25 | 0.2 | mod | 36, 92, 166 | 9, 34, 74 | 0, 0, 20 |
| 0.1 | 0.3 | 0.1 | mod | 37, 92, 169 | 10, 34, 77 | 0, 0, 23 |
| 0.1 | 0.3 | 0.15 | mod | 37, 93, 168 | 10, 35, 76 | 0, 0, 22 |
| 0.1 | 0.3 | 0.2 | mod | 37, 93, 167 | 10, 35, 75 | 0, 0, 21 |
| 0.01 | 0.15 | 0.1 | Agg | 38, 87, 153 | 11, 29, 61 | 0, 0, 15 |
| 0.01 | 0.15 | 0.15 | Agg | 37, 90, 154 | 10, 32, 62 | 0, 0, 15 |
| 0.01 | 0.15 | 0.2 | Agg | 38, 90, 156 | 11, 32, 64 | 0, 0, 17 |
| 0.01 | 0.2 | 0.1 | Agg | 39, 89, 155 | 12, 31, 63 | 0, 0, 15 |
| 0.01 | 0.2 | 0.15 | agg | 38, 87, 153 | 11, 29, 61 | 0, 0, 16 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Cluster utility criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.2 | 0.2 | agg | 39, 90, 158 | 12, 32, 66 | 0, 0, 19 |
| 0.01 | 0.25 | 0.1 | agg | 39, 91, 157 | 12, 33, 65 | 0, 0, 18 |
| 0.01 | 0.25 | 0.15 | agg | 38, 86, 154 | 11, 28, 62 | 0, 0, 15 |
| 0.01 | 0.25 | 0.2 | agg | 38, 88, 155 | 11, 30, 63 | 0, 0, 17 |
| 0.01 | 0.3 | 0.1 | agg | 39, 89, 158 | 12, 31, 66 | 0, 0, 19 |
| 0.01 | 0.3 | 0.15 | agg | 38, 88, 157 | 11, 30, 65 | 0, 0, 17 |
| 0.01 | 0.3 | 0.2 | agg | 38, 87, 150 | 11, 29, 58 | 0, 0, 13 |
| 0.05 | 0.15 | 0.1 | agg | 42, 99, 184 | 15, 41, 92 | 0, 1, 35 |
| 0.05 | 0.15 | 0.15 | agg | 42, 99, 184 | 15, 41, 92 | 0, 2, 36 |
| 0.05 | 0.15 | 0.2 | agg | 42, 99, 184 | 15, 41, 92 | 0, 1, 35 |
| 0.05 | 0.2 | 0.1 | agg | 43, 101, 183 | 16, 43, 91 | 0, 2, 34 |
| 0.05 | 0.2 | 0.15 | agg | 42, 100, 186 | 15, 42, 94 | 0, 2, 36 |
| 0.05 | 0.2 | 0.2 | agg | 42, 98, 182 | 15, 40, 90 | 0, 3, 33 |
| 0.05 | 0.25 | 0.1 | agg | 42, 100, 186 | 15, 42, 94 | 0, 1, 38 |
| 0.05 | 0.25 | 0.15 | agg | 42, 99, 184 | 15, 41, 92 | 0, 1, 34 |
| 0.05 | 0.25 | 0.2 | agg | 43, 100, 181 | 16, 42, 89 | 0, 1, 33 |
| 0.05 | 0.3 | 0.1 | agg | 42, 102, 186 | 15, 44, 94 | 0, 2, 35 |
| 0.05 | 0.3 | 0.15 | agg | 42, 100, 186 | 15, 42, 94 | 0, 2, 36 |
| 0.05 | 0.3 | 0.2 | agg | 42, 102, 184 | 15, 44, 92 | 0, 3, 34 |
| 0.1 | 0.15 | 0.1 | agg | 45, 112, 199 | 18, 54, 107 | 0, 6, 47 |
| 0.1 | 0.15 | 0.15 | agg | 45, 113, 201 | 18, 55, 109 | 0, 6, 49 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Cluster utility criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.1 | 0.15 | 0.2 | agg | 43, 111, 200 | 16, 53, 108 | 0, 5, 49 |
| 0.1 | 0.2 | 0.1 | agg | 45, 112, 201 | 18, 54, 109 | 0, 5, 49 |
| 0.1 | 0.2 | 0.15 | agg | 43, 114, 203 | 16, 56, 111 | 0, 7, 51 |
| 0.1 | 0.2 | 0.2 | agg | 44, 110, 200 | 17, 52, 108 | 0, 5, 48 |
| 0.1 | 0.25 | 0.1 | agg | 43, 112, 205 | 16, 54, 113 | 0, 6, 53 |
| 0.1 | 0.25 | 0.15 | agg | 44, 109, 201 | 17, 51, 109 | 0, 5, 49 |
| 0.1 | 0.25 | 0.2 | agg | 43, 109, 201 | 16, 51, 109 | 0, 3, 49 |
| 0.1 | 0.3 | 0.1 | agg | 43, 110, 205 | 16, 52, 113 | 0, 5, 53 |
| 0.1 | 0.3 | 0.15 | agg | 44, 115, 210 | 17, 57, 118 | 0, 6, 58 |
| 0.1 | 0.3 | 0.2 | agg | 44, 115, 209 | 17, 57, 117 | 0, 7, 57 |

Results for $Phi$ = 45k and 25%, 50% and 75% of data: According to current model: HUI in 100% data = 130, HUI in 25% data = 25, HUI in 50% data = 46, HUI in 75% data = 78 According to predictive UIM model:

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.15 | 0.1 | con | 30, 66, 105 | 5, 20, 27 | 0, 0, 1 |
| 0.01 | 0.15 | 0.15 | con | 30, 66, 104 | 5, 20, 26 | 0, 0, 0 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.15 | 0.2 | con | 30, 66, 106 | 5, 20, 28 | 0, 0, 2 |
| 0.01 | 0.2 | 0.1 | con | 30, 65, 106 | 5, 19, 28 | 0, 0, 2 |
| 0.01 | 0.2 | 0.15 | con | 30, 64, 105 | 5, 18, 27 | 0, 0, 1 |
| 0.01 | 0.2 | 0.2 | con | 30, 68, 106 | 5, 22, 28 | 0, 0, 2 |
| 0.01 | 0.25 | 0.1 | con | 30, 66, 106 | 5, 20, 28 | 0, 0, 1 |
| 0.01 | 0.25 | 0.15 | con | 30, 65, 104 | 5, 19, 26 | 0, 0, 0 |
| 0.01 | 0.25 | 0.2 | con | 30, 66, 105 | 5, 20, 27 | 0, 0, 1 |
| 0.01 | 0.3 | 0.1 | con | 30, 68, 109 | 5, 22, 31 | 0, 0, 3 |
| 0.01 | 0.3 | 0.15 | con | 30, 65, 106 | 5, 19, 28 | 0, 0, 2 |
| 0.01 | 0.3 | 0.2 | con | 30, 67, 107 | 5, 21, 29 | 0, 0, 2 |
| 0.05 | 0.15 | 0.1 | con | 32, 74, 120 | 7, 28, 42 | 0, 0, 4 |
| 0.05 | 0.15 | 0.15 | con | 32, 73, 118 | 7, 27, 40 | 0, 0, 4 |
| 0.05 | 0.15 | 0.2 | con | 32, 73, 117 | 7, 27, 39 | 0, 0, 4 |
| 0.05 | 0.2 | 0.1 | con | 32, 73, 120 | 7, 27, 42 | 0, 0, 4 |
| 0.05 | 0.2 | 0.15 | con | 32, 75, 120 | 7, 29, 42 | 0, 0, 4 |
| 0.05 | 0.2 | 0.2 | con | 32, 75, 119 | 7, 29, 41 | 0, 0, 4 |
| 0.05 | 0.25 | 0.1 | con | 32, 73, 120 | 7, 27, 42 | 0, 0, 4 |
| 0.05 | 0.25 | 0.15 | con | 32, 75, 120 | 7, 29, 42 | 0, 0, 4 |
| 0.05 | 0.25 | 0.2 | con | 32, 75, 120 | 7, 29, 42 | 0, 0, 4 |
| 0.05 | 0.3 | 0.1 | con | 32, 74, 121 | 7, 28, 43 | 0, 0, 4 |
| 0.05 | 0.3 | 0.15 | con | 32, 74, 120 | 7, 28, 42 | 0, 0, 4 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.05 | 0.3 | 0.2 | con | 32, 74, 121 | 7, 28, 43 | 0, 0, 4 |
| 0.1 | 0.15 | 0.1 | con | 34, 77, 127 | 9, 31, 49 | 0, 1, 6 |
| 0.1 | 0.15 | 0.15 | con | 34, 77, 125 | 9, 31, 47 | 0, 1, 6 |
| 0.1 | 0.15 | 0.2 | con | 33, 77, 127 | 8, 31, 49 | 0, 1, 6 |
| 0.1 | 0.2 | 0.1 | con | 33, 77, 127 | 8, 31, 49 | 0, 1, 5 |
| 0.1 | 0.2 | 0.15 | con | 34, 77, 127 | 9, 31, 49 | 0, 1, 6 |
| 0.1 | 0.2 | 0.2 | con | 34, 77, 127 | 9, 31, 49 | 0, 1, 6 |
| 0.1 | 0.25 | 0.1 | con | 34, 77, 125 | 9, 31, 47 | 0, 1, 6 |
| 0.1 | 0.25 | 0.15 | con | 35, 77, 128 | 10, 31, 50 | 0, 1, 7 |
| 0.1 | 0.25 | 0.2 | con | 33, 77, 125 | 8, 31, 47 | 0, 1, 6 |
| 0.1 | 0.3 | 0.1 | con | 34, 77, 127 | 9, 31, 49 | 0, 1, 6 |
| 0.1 | 0.3 | 0.15 | con | 35, 77, 128 | 10, 31, 50 | 0, 1, 7 |
| 0.1 | 0.3 | 0.2 | con | 35, 77, 128 | 10, 31, 50 | 0, 1, 7 |
| 0.01 | 0.15 | 0.1 | mod | 31, 72, 119 | 6, 26, 41 | 0, 0, 4 |
| 0.01 | 0.15 | 0.15 | mod | 31, 71, 118 | 6, 25, 40 | 0, 0, 4 |
| 0.01 | 0.15 | 0.2 | mod | 32, 73, 118 | 7, 27, 40 | 0, 0, 4 |
| 0.01 | 0.2 | 0.1 | mod | 32, 73, 118 | 7, 27, 40 | 0, 0, 4 |
| 0.01 | 0.2 | 0.15 | mod | 31, 72, 118 | 6, 26, 40 | 0, 0, 4 |
| 0.01 | 0.2 | 0.2 | mod | 32, 73, 119 | 7, 27, 41 | 0, 0, 4 |
| 0.01 | 0.25 | 0.1 | mod | 31, 72, 118 | 6, 26, 40 | 0, 0, 4 |
| 0.01 | 0.25 | 0.15 | mod | 31, 73, 119 | 6, 27, 41 | 0, 0, 4 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.25 | 0.2 | mod | 32, 73, 118 | 7, 27, 40 | 0, 0, 4 |
| 0.01 | 0.3 | 0.1 | mod | 31, 73, 118 | 6, 27, 40 | 0, 0, 4 |
| 0.01 | 0.3 | 0.15 | mod | 31, 73, 119 | 6, 27, 41 | 0, 0, 4 |
| 0.01 | 0.3 | 0.2 | mod | 31, 73, 118 | 6, 27, 40 | 0, 0, 4 |
| 0.05 | 0.15 | 0.1 | mod | 32, 77, 127 | 7, 31, 49 | 0, 1, 6 |
| 0.05 | 0.15 | 0.15 | mod | 32, 77, 127 | 7, 31, 49 | 0, 1, 6 |
| 0.05 | 0.15 | 0.2 | mod | 33, 77, 127 | 8, 31, 49 | 0, 1, 6 |
| 0.05 | 0.2 | 0.1 | mod | 33, 77, 129 | 8, 31, 51 | 0, 1, 7 |
| 0.05 | 0.2 | 0.15 | mod | 32, 77, 129 | 7, 31, 51 | 0, 1, 7 |
| 0.05 | 0.2 | 0.2 | mod | 33, 77, 127 | 8, 31, 49 | 0, 1, 6 |
| 0.05 | 0.25 | 0.1 | mod | 32, 77, 128 | 7, 31, 50 | 0, 1, 6 |
| 0.05 | 0.25 | 0.15 | mod | 33, 77, 128 | 8, 31, 50 | 0, 1, 7 |
| 0.05 | 0.25 | 0.2 | mod | 33, 77, 128 | 8, 31, 50 | 0, 1, 7 |
| 0.05 | 0.3 | 0.1 | mod | 32, 77, 129 | 7, 31, 51 | 0, 1, 7 |
| 0.05 | 0.3 | 0.15 | mod | 33, 77, 130 | 8, 31, 52 | 0, 1, 7 |
| 0.05 | 0.3 | 0.2 | mod | 32, 77, 128 | 7, 31, 50 | 0, 1, 6 |
| 0.1 | 0.15 | 0.1 | mod | 35, 77, 131 | 10, 31, 53 | 0, 1, 8 |
| 0.1 | 0.15 | 0.15 | mod | 35, 77, 132 | 10, 31, 54 | 0, 1, 9 |
| 0.1 | 0.15 | 0.2 | mod | 35, 77, 131 | 10, 31, 53 | 0, 1, 8 |
| 0.1 | 0.2 | 0.1 | mod | 35, 78, 132 | 10, 32, 54 | 0, 1, 9 |
| 0.1 | 0.2 | 0.15 | mod | 35, 77, 131 | 10, 31, 53 | 0, 1, 8 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.1 | 0.2 | 0.2 | mod | 35, 78, 132 | 10, 32, 54 | 0, 1, 9 |
| 0.1 | 0.25 | 0.1 | mod | 35, 78, 132 | 10, 32, 54 | 0, 1, 9 |
| 0.1 | 0.25 | 0.15 | mod | 35, 77, 131 | 10, 31, 53 | 0, 1, 8 |
| 0.1 | 0.25 | 0.2 | mod | 35, 77, 132 | 10, 31, 54 | 0, 1, 9 |
| 0.1 | 0.3 | 0.1 | mod | 35, 78, 133 | 10, 32, 55 | 0, 1, 10 |
| 0.1 | 0.3 | 0.15 | mod | 35, 78, 133 | 10, 32, 55 | 0, 1, 10 |
| 0.1 | 0.3 | 0.2 | mod | 35, 78, 132 | 10, 32, 54 | 0, 1, 9 |
| 0.01 | 0.15 | 0.1 | agg | 35, 80, 129 | 10, 34, 51 | 0, 1, 11 |
| 0.01 | 0.15 | 0.15 | agg | 34, 77, 123 | 9, 31, 45 | 0, 1, 7 |
| 0.01 | 0.15 | 0.2 | agg | 35, 79, 127 | 10, 33, 49 | 0, 1, 11 |
| 0.01 | 0.2 | 0.1 | agg | 33, 79, 126 | 8, 33, 48 | 0, 2, 8 |
| 0.01 | 0.2 | 0.15 | agg | 34, 79, 129 | 9, 33, 51 | 0, 1, 10 |
| 0.01 | 0.2 | 0.2 | agg | 35, 78, 132 | 10, 32, 54 | 0, 1, 13 |
| 0.01 | 0.25 | 0.1 | agg | 33, 79, 128 | 8, 33, 50 | 0, 1, 10 |
| 0.01 | 0.25 | 0.15 | agg | 35, 78, 127 | 10, 32, 49 | 0, 1, 10 |
| 0.01 | 0.25 | 0.2 | agg | 35, 81, 130 | 10, 35, 52 | 0, 1, 12 |
| 0.01 | 0.3 | 0.1 | agg | 35, 80, 126 | 10, 34, 48 | 0, 1, 10 |
| 0.01 | 0.3 | 0.15 | agg | 35, 80, 127 | 10, 34, 49 | 0, 1, 10 |
| 0.01 | 0.3 | 0.2 | agg | 34, 78, 129 | 9, 32, 51 | 0, 1, 9 |
| 0.05 | 0.15 | 0.1 | agg | 37, 86, 150 | 12, 40, 72 | 0, 2, 26 |
| 0.05 | 0.15 | 0.15 | agg | 37, 86, 151 | 12, 40, 73 | 0, 2, 26 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.05 | 0.15 | 0.2 | agg | 36, 86, 149 | 11, 40, 71 | 0, 2, 25 |
| 0.05 | 0.2 | 0.1 | agg | 35, 86, 151 | 10, 40, 73 | 0, 2, 27 |
| 0.05 | 0.2 | 0.15 | agg | 37, 86, 150 | 12, 40, 72 | 0, 2, 26 |
| 0.05 | 0.2 | 0.2 | agg | 37, 86, 150 | 12, 40, 72 | 0, 2, 26 |
| 0.05 | 0.25 | 0.1 | agg | 36, 85, 149 | 11, 39, 71 | 0, 2, 25 |
| 0.05 | 0.25 | 0.15 | agg | 37, 86, 151 | 12, 40, 73 | 0, 2, 27 |
| 0.05 | 0.25 | 0.2 | agg | 35, 89, 151 | 10, 43, 73 | 0, 2, 26 |
| 0.05 | 0.3 | 0.1 | agg | 37, 86, 150 | 12, 40, 72 | 0, 2, 26 |
| 0.05 | 0.3 | 0.15 | agg | 36, 85, 150 | 11, 39, 72 | 0, 2, 26 |
| 0.05 | 0.3 | 0.2 | agg | 37, 87, 153 | 12, 41, 75 | 0, 2, 28 |
| 0.1 | 0.15 | 0.1 | agg | 39, 93, 163 | 14, 47, 85 | 0, 2, 37 |
| 0.1 | 0.15 | 0.15 | agg | 39, 92, 166 | 14, 46, 88 | 0, 2, 41 |
| 0.1 | 0.15 | 0.2 | agg | 39, 92, 162 | 14, 46, 84 | 0, 2, 36 |
| 0.1 | 0.2 | 0.1 | agg | 39, 92, 163 | 14, 46, 85 | 0, 2, 38 |
| 0.1 | 0.2 | 0.15 | agg | 40, 94, 164 | 15, 48, 86 | 0, 2, 39 |
| 0.1 | 0.2 | 0.2 | agg | 40, 93, 166 | 15, 47, 88 | 0, 2, 40 |
| 0.1 | 0.25 | 0.1 | agg | 39, 93, 169 | 14, 47, 91 | 0, 2, 43 |
| 0.1 | 0.25 | 0.15 | agg | 40, 93, 166 | 15, 47, 88 | 0, 2, 40 |
| 0.1 | 0.25 | 0.2 | agg | 39, 93, 164 | 14, 47, 86 | 0, 2, 39 |
| 0.1 | 0.3 | 0.1 | agg | 40, 92, 168 | 15, 46, 90 | 0, 2, 41 |
| 0.1 | 0.3 | 0.15 | agg | 40, 94, 169 | 15, 48, 91 | 0, 2, 43 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.1 | 0.3 | 0.2 | agg | 40, 93, 172 | 15, 47, 94 | 0, 2, 45 |

Results for *Phi* = 50k and 25%, 50% and 75% of data: According to current model: HUI in 100% data = 111, HUI in 25% data = 23, HUI in 50% data = 39, HUI in 75% data = 64 According to predictive UIM model:

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.15 | 0.1 | con | 27, 53, 92 | 4, 14, 28 | 0, 0, 0 |
| 0.01 | 0.15 | 0.15 | con | 28, 52, 91 | 5, 13, 27 | 0, 0, 0 |
| 0.01 | 0.15 | 0.2 | con | 27, 52, 91 | 4, 13, 27 | 0, 0, 0 |
| 0.01 | 0.2 | 0.1 | con | 28, 52, 91 | 5, 13, 27 | 0, 0, 0 |
| 0.01 | 0.2 | 0.15 | con | 27, 52, 91 | 4, 13, 27 | 0, 0, 0 |
| 0.01 | 0.2 | 0.2 | con | 28, 52, 92 | 5, 13, 28 | 0, 0, 0 |
| 0.01 | 0.25 | 0.1 | con | 28, 54, 92 | 5, 15, 28 | 0, 0, 0 |
| 0.01 | 0.25 | 0.15 | con | 27, 53, 90 | 4, 14, 26 | 0, 0, 0 |
| 0.01 | 0.25 | 0.2 | con | 28, 53, 92 | 5, 14, 28 | 0, 0, 0 |
| 0.01 | 0.3 | 0.1 | con | 29, 53, 91 | 6, 14, 27 | 0, 0, 0 |
| 0.01 | 0.3 | 0.15 | con | 28, 53, 91 | 5, 14, 27 | 0, 0, 0 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.3 | 0.2 | con | 28, 53, 91 | 5, 14, 27 | 0, 0, 0 |
| 0.05 | 0.15 | 0.1 | con | 29, 63, 101 | 6, 24, 37 | 0, 1, 2 |
| 0.05 | 0.15 | 0.15 | con | 29, 62, 99 | 6, 23, 35 | 0, 1, 2 |
| 0.05 | 0.15 | 0.2 | con | 30, 62, 99 | 7, 23, 35 | 0, 1, 1 |
| 0.05 | 0.2 | 0.1 | con | 30, 62, 102 | 7, 23, 38 | 0, 1, 2 |
| 0.05 | 0.2 | 0.15 | con | 30, 62, 102 | 7, 23, 38 | 0, 1, 2 |
| 0.05 | 0.2 | 0.2 | con | 30, 62, 99 | 7, 23, 35 | 0, 1, 1 |
| 0.05 | 0.25 | 0.1 | con | 29, 63, 100 | 6, 24, 36 | 0, 1, 2 |
| 0.05 | 0.25 | 0.15 | con | 30, 63, 103 | 7, 24, 39 | 0, 1, 2 |
| 0.05 | 0.25 | 0.2 | con | 30, 64, 100 | 7, 25, 36 | 0, 1, 2 |
| 0.05 | 0.3 | 0.1 | con | 30, 63, 102 | 7, 24, 38 | 0, 1, 2 |
| 0.05 | 0.3 | 0.15 | con | 30, 63, 102 | 7, 24, 38 | 0, 1, 2 |
| 0.05 | 0.3 | 0.2 | con | 30, 64, 102 | 7, 25, 38 | 0, 1, 2 |
| 0.1 | 0.15 | 0.1 | con | 31, 66, 108 | 8, 27, 44 | 0, 1, 5 |
| 0.1 | 0.15 | 0.15 | con | 31, 66, 107 | 8, 27, 43 | 0, 1, 4 |
| 0.1 | 0.15 | 0.2 | con | 31, 64, 105 | 8, 25, 41 | 0, 1, 4 |
| 0.1 | 0.2 | 0.1 | con | 31, 66, 109 | 8, 27, 45 | 0, 1, 5 |
| 0.1 | 0.2 | 0.15 | con | 31, 67, 106 | 8, 28, 42 | 0, 2, 5 |
| 0.1 | 0.2 | 0.2 | con | 31, 66, 105 | 8, 27, 41 | 0, 1, 3 |
| 0.1 | 0.25 | 0.1 | con | 31, 65, 108 | 8, 26, 44 | 0, 1, 4 |
| 0.1 | 0.25 | 0.15 | con | 31, 65, 108 | 8, 26, 44 | 0, 1, 5 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.1 | 0.25 | 0.2 | con | 31, 65, 107 | 8, 26, 43 | 0, 1, 4 |
| 0.1 | 0.3 | 0.1 | con | 31, 67, 110 | 8, 28, 46 | 0, 1, 7 |
| 0.1 | 0.3 | 0.15 | con | 31, 65, 109 | 8, 26, 45 | 0, 1, 5 |
| 0.1 | 0.3 | 0.2 | con | 31, 66, 107 | 8, 27, 43 | 0, 1, 4 |
| 0.01 | 0.15 | 0.1 | mod | 29, 62, 98 | 6, 23, 34 | 0, 0, 0 |
| 0.01 | 0.15 | 0.15 | mod | 29, 61, 98 | 6, 22, 34 | 0, 0, 0 |
| 0.01 | 0.15 | 0.2 | mod | 29, 61, 98 | 6, 22, 34 | 0, 0, 0 |
| 0.01 | 0.2 | 0.1 | mod | 29, 62, 98 | 6, 23, 34 | 0, 0, 0 |
| 0.01 | 0.2 | 0.15 | mod | 29, 61, 99 | 6, 22, 35 | 0, 0, 0 |
| 0.01 | 0.2 | 0.2 | mod | 29, 62, 99 | 6, 23, 35 | 0, 1, 1 |
| 0.01 | 0.25 | 0.1 | mod | 29, 62, 99 | 6, 23, 35 | 0, 0, 0 |
| 0.01 | 0.25 | 0.15 | mod | 29, 61, 99 | 6, 22, 35 | 0, 0, 0 |
| 0.01 | 0.25 | 0.2 | mod | 29, 62, 98 | 6, 23, 34 | 0, 0, 0 |
| 0.01 | 0.3 | 0.1 | mod | 29, 62, 100 | 6, 23, 36 | 0, 1, 1 |
| 0.01 | 0.3 | 0.15 | mod | 29, 62, 99 | 6, 23, 35 | 0, 0, 0 |
| 0.01 | 0.3 | 0.2 | mod | 30, 61, 99 | 7, 22, 35 | 0, 0, 0 |
| 0.05 | 0.15 | 0.1 | mod | 31, 65, 107 | 8, 26, 43 | 0, 1, 3 |
| 0.05 | 0.15 | 0.15 | mod | 31, 64, 107 | 8, 25, 43 | 0, 1, 3 |
| 0.05 | 0.15 | 0.2 | mod | 31, 65, 108 | 8, 26, 44 | 0, 1, 4 |
| 0.05 | 0.2 | 0.1 | mod | 31, 65, 107 | 8, 26, 43 | 0, 1, 3 |
| 0.05 | 0.2 | 0.15 | mod | 31, 65, 107 | 8, 26, 43 | 0, 1, 3 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.05 | 0.2 | 0.2 | mod | 31, 64, 107 | 8, 25, 43 | 0, 1, 3 |
| 0.05 | 0.25 | 0.1 | mod | 31, 65, 109 | 8, 26, 45 | 0, 1, 4 |
| 0.05 | 0.25 | 0.15 | mod | 31, 65, 106 | 8, 26, 42 | 0, 1, 2 |
| 0.05 | 0.25 | 0.2 | mod | 31, 65, 108 | 8, 26, 44 | 0, 1, 4 |
| 0.05 | 0.3 | 0.1 | mod | 31, 65, 111 | 8, 26, 47 | 0, 1, 5 |
| 0.05 | 0.3 | 0.15 | mod | 31, 66, 108 | 8, 27, 44 | 0, 1, 4 |
| 0.05 | 0.3 | 0.2 | mod | 31, 65, 110 | 8, 26, 46 | 0, 1, 5 |
| 0.1 | 0.15 | 0.1 | mod | 31, 68, 116 | 8, 29, 52 | 0, 1, 9 |
| 0.1 | 0.15 | 0.15 | mod | 31, 68, 117 | 8, 29, 53 | 0, 1, 10 |
| 0.1 | 0.15 | 0.2 | mod | 31, 68, 114 | 8, 29, 50 | 0, 1, 7 |
| 0.1 | 0.2 | 0.1 | mod | 31, 68, 116 | 8, 29, 52 | 0, 1, 9 |
| 0.1 | 0.2 | 0.15 | mod | 31, 69, 117 | 8, 30, 53 | 0, 2, 10 |
| 0.1 | 0.2 | 0.2 | mod | 31, 68, 115 | 8, 29, 51 | 0, 1, 8 |
| 0.1 | 0.25 | 0.1 | mod | 31, 69, 116 | 8, 30, 52 | 0, 1, 9 |
| 0.1 | 0.25 | 0.15 | mod | 31, 68, 116 | 8, 29, 52 | 0, 1, 9 |
| 0.1 | 0.25 | 0.2 | mod | 31, 68, 116 | 8, 29, 52 | 0, 1, 9 |
| 0.1 | 0.3 | 0.1 | mod | 31, 69, 117 | 8, 30, 53 | 0, 1, 10 |
| 0.1 | 0.3 | 0.15 | mod | 31, 69, 117 | 8, 30, 53 | 0, 1, 10 |
| 0.1 | 0.3 | 0.2 | mod | 31, 69, 117 | 8, 30, 53 | 0, 1, 10 |
| 0.01 | 0.15 | 0.1 | agg | 31, 69, 106 | 8, 30, 42 | 0, 1, 7 |
| 0.01 | 0.15 | 0.15 | agg | 32, 68, 108 | 9, 29, 44 | 0, 2, 6 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.01 | 0.15 | 0.2 | agg | 33, 71, 110 | 10, 32, 46 | 0, 2, 8 |
| 0.01 | 0.2 | 0.1 | agg | 33, 70, 106 | 10, 31, 42 | 0, 2, 6 |
| 0.01 | 0.2 | 0.15 | agg | 31, 69, 105 | 8, 30, 41 | 0, 2, 5 |
| 0.01 | 0.2 | 0.2 | agg | 33, 72, 107 | 10, 33, 43 | 0, 2, 6 |
| 0.01 | 0.25 | 0.1 | agg | 32, 69, 107 | 9, 30, 43 | 0, 2, 6 |
| 0.01 | 0.25 | 0.15 | agg | 32, 69, 107 | 9, 30, 43 | 0, 2, 6 |
| 0.01 | 0.25 | 0.2 | agg | 33, 69, 107 | 10, 30, 43 | 0, 1, 6 |
| 0.01 | 0.3 | 0.1 | agg | 32, 70, 109 | 9, 31, 45 | 0, 2, 7 |
| 0.01 | 0.3 | 0.15 | agg | 32, 69, 107 | 9, 30, 43 | 0, 2, 6 |
| 0.01 | 0.3 | 0.2 | agg | 32, 70, 107 | 9, 31, 43 | 0, 2, 6 |
| 0.05 | 0.15 | 0.1 | agg | 33, 78, 125 | 10, 39, 61 | 0, 3, 18 |
| 0.05 | 0.15 | 0.15 | agg | 34, 77, 126 | 11, 38, 62 | 0, 3, 19 |
| 0.05 | 0.15 | 0.2 | agg | 34, 77, 122 | 11, 38, 58 | 0, 3, 15 |
| 0.05 | 0.2 | 0.1 | agg | 33, 77, 124 | 10, 38, 60 | 0, 3, 17 |
| 0.05 | 0.2 | 0.15 | agg | 33, 78, 125 | 10, 39, 61 | 0, 3, 18 |
| 0.05 | 0.2 | 0.2 | agg | 34, 77, 125 | 11, 38, 61 | 0, 3, 18 |
| 0.05 | 0.25 | 0.1 | agg | 34, 77, 123 | 11, 38, 59 | 0, 3, 16 |
| 0.05 | 0.25 | 0.15 | agg | 34, 78, 126 | 11, 39, 62 | 0, 3, 19 |
| 0.05 | 0.25 | 0.2 | agg | 35, 78, 125 | 12, 39, 61 | 0, 3, 18 |
| 0.05 | 0.3 | 0.1 | agg | 34, 78, 125 | 11, 39, 61 | 0, 3, 18 |
| 0.05 | 0.3 | 0.15 | agg | 35, 78, 127 | 12, 39, 63 | 0, 3, 20 |

| Fraction of transactions used in Clustering | Min_aff | Min_uty | Clustering criterion | HUI found (in 25%, 50% and 75% data respectively) | Additional HUI found (in 25%, 50% and 75% data respectively) | Misses (in 25%, 50% and 75% data respectively) |
|---|---|---|---|---|---|---|
| 0.05 | 0.3 | 0.2 | agg | 33, 79, 126 | 10, 40, 62 | 0, 3, 19 |
| 0.1 | 0.15 | 0.1 | agg | 35, 81, 140 | 12, 42, 76 | 0, 4, 33 |
| 0.1 | 0.15 | 0.15 | agg | 35, 82, 139 | 12, 43, 75 | 0, 4, 32 |
| 0.1 | 0.15 | 0.2 | agg | 35, 82, 138 | 12, 43, 74 | 0, 4, 31 |
| 0.1 | 0.2 | 0.1 | agg | 35, 81, 139 | 12, 42, 75 | 0, 4, 32 |
| 0.1 | 0.2 | 0.15 | agg | 35, 83, 141 | 12, 44, 77 | 0, 4, 34 |
| 0.1 | 0.2 | 0.2 | agg | 35, 80, 143 | 12, 41, 79 | 0, 3, 36 |
| 0.1 | 0.25 | 0.1 | agg | 35, 81, 144 | 12, 42, 80 | 0, 4, 37 |
| 0.1 | 0.25 | 0.15 | agg | 35, 82, 142 | 12, 43, 78 | 0, 4, 35 |
| 0.1 | 0.25 | 0.2 | agg | 35, 80, 140 | 12, 41, 76 | 0, 3, 33 |
| 0.1 | 0.3 | 0.1 | agg | 35, 82, 142 | 12, 43, 78 | 0, 4, 35 |
| 0.1 | 0.3 | 0.15 | agg | 35, 83, 145 | 12, 44, 81 | 0, 4, 38 |
| 0.1 | 0.3 | 0.2 | agg | 35, 82, 145 | 12, 43, 81 | 0, 4, 38 |