BAYESIAN MIXTURE MODELS FOR FREQUENT ITEMSET MINING

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2012

By
Ruofei He
School of Computer Science

Contents

Word Count: 999,999

List of Tables

List of Figures

List of Algorithms

Abstract

In binary-transaction data-mining, traditional frequent itemset mining often produces results which are not straightforward to interpret. To overcome this problem, probability models are often used to produce more compact and conclusive results, albeit with some loss of accuracy. Bayesian statistics have been widely used in the development of probability models in machine learning in recent years and these methods have many advantages, including their abilities to avoid overfitting. In this thesis, we develop two Bayesian mixture models with the Dirichlet distribution prior and the Dirichlet process (DP) prior to improve the previous non-Bayesian mixture model developed for transaction dataset mining.

First, we develop a finite Bayesian mixture model by introducing conjugate priors to the model. Then, we extend this model to an infinite Bayesian mixture using a Dirichlet process prior. The Dirichlet process mixture model is a non-parametric Bayesian model which allows for the automatic determination of an appropriate number of mixture components. We implement the inference of both mixture models using two methods: a collapsed Gibbs sampling scheme and a variational approximation algorithm.

Experiments in several benchmark problems have shown that both mixture models achieve better performance than a non-Bayesian mixture model. The variational algorithm is the faster of the two approaches while the Gibbs sampling method achieves a more accurate result. The Dirichlet process mixture model can automatically grow to a proper complexity for a better approximation. However, these approaches also show that mixture models underestimate the probabilities of frequent itemsets. Consequently, these models have a higher sensitivity but a lower specificity.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Acknowledgements

During the 4 years of my PhD, I met many great people. In truth, they deserve greater tributes than the few lines I will take in thanking them here.

It was my great fortune to have been a student of Jonathan Shapiro, my supervisor. Without him, all my work would have been impossible. In the past three years, I have learned so much from him; not just the EM algorithm, the Gaussian quadrature or the Bayesian inference, but also the attitude and the passion required to fuel this work. I still remember the times he taught me about statistics. In his office, he would write equations on the white board, and patiently explain every detail to assuage any confusion I had. I really enjoyed those moments. I was not Jon's student at first but was transferred to him after my first year. In the following three years, his efforts to help me with my work were immeasurable and his encouragement in times of doubt was inspirational. It is hard enough for me to demonstrate my emotions in my own language let alone in another, but I hope this goes some way to expressing my sincere gratitude to him, and my best wishes for his future health and happiness.

I also want to thank my former supervisor Christo for helping me to get the opportunity to study at the University of Manchester. I thank you for all you have done for me.

To my dear friends, Xinkai Wang and Liu Zhou; with you, I feel like I am at home. Living with you both has provided me with my happiest times in the UK. Thank you Geng Li for all the things you have helped me with - I will miss the time that we played and talked together. Thank you Mingjie Zhao, for your invaluable help in maths and Chinese chess. And Richard, do you remember the time that we spent in the steam room at the Aquatic centre? Thank you for your help in the office. Maybe someday in the future I will visit you on some Australian beach and we can talk together in the sunshine. I would also like to thank all my colleagues in MLO group.

Crucially, a great many thanks go to my family. It is no exaggeration to say that without them I would never have been able to complete my work. Your support and encouragement are the powers that continue to propel me forward.

Chapter 1

Introduction

Transaction datasets are binary datasets with rows corresponding to transactions and columns corresponding to items or attributes. Data mining techniques for such datasets have been developed for over a decade. Methods for finding correlations and regularities in transaction data can have many commercial and practical applications, including targeted marketing, recommender systems, more effective product placement and many others.

Retail records and website logs are two examples of transaction datasets. For example, in a retail application, the rows of the data correspond to purchases made by various customers and the columns correspond to different items for sale in the store. This kind of data is often sparse: whilst there may be thousands of items for sale, a typical transaction may contain only a handful of items, as most of the customers buy only a small fraction of the available merchandise. Although transaction data can also contain the numbers of each item purchased (multi-nomial data), in this thesis we will only consider binary transaction data. An important correlation which data mining seeks to elucidate is which items co-occur in purchases and which items are mutually exclusive, never (or rarely) co-occurring in transactions. This information allows the inference of future purchases from past ones.

Frequent itemset mining and association rule mining [AIS93] are the key approaches for finding correlations in transaction data. Frequent itemset mining finds all frequently occurring item combinations along with their frequencies in the dataset with a given minimum frequency threshold. Association rule mining uses the results of frequent itemset mining to find the dependencies between items or sets of items. If we regard the minimum frequency threshold as an importance

standard, then the set of frequent itemsets contains all the "important" information about the correlation of the dataset. The aim of frequent itemset mining is to extract useful information from the kinds of binary datasets which are now ubiquitous in human society. It aims to help people realize and understand the various latent correlations hidden in the data and to assist people in decision making, policy adjustment and the performance of other activities which rely on correct analysis and knowledge of the data.

However, the results of such mining are difficult to use. The threshold, or criterion for mining, is hard to choose for a compact but representative set of itemsets. To prevent the loss of important information, the threshold is often set quite low, resulting in a huge set of itemsets which brings difficulties in interpretation. These properties of large scale and weak interpretability block a wider use of the mining technique and are barriers to a further understanding of the data itself. People therefore need the mining results to be more informative and compact. The development of FIM, as we will summarize in Chapter 2, can be roughly organized into three directions.

1. Looking for more compact but representative forms of the itemsets - in other words, mining compressed itemsets. The research in this direction consists of two types: lossless compression such as closed itemsets [PBTL99], and lossy compression such as maximal itemsets [CG02].

2. Looking for better standards and qualifications for filtering the itemsets so that the results are more "interesting" to users. Work in this direction focuses on how to extract the information which is both useful and unexpected as people want to find a measure that is closest to the ideal of "interestingness". Several objective and subjective measures are proposed such as *lift* [Mac96], χ^2 [BMS97] and the work of [Jar04].

3. Looking for probability models which reveal and describe both the structure and the inner-relationship of the data more accurately, clearly and thoroughly. There are two ways of using probability models in FIM. The first is to build a probability model that can organize and utilize the results of mining, such as the *Maximal Entropy model* [Tat08]. The second is to build a probability model that is directly generated from the data itself which can not only predict the frequent itemsets, but also explain the data. An example of such model is the *Mixture model*.

These three directions influence each other and form the main stream of current FIM research. Of the three, the probability model solution considers the data as a sampling result from the latent system and tries to explain the system in an understandable, structural and quantified way. With a good probability model, we can expect the following advantages in comparison with normal frequent itemset mining:

1. The model can interpret correlations in, and promote understanding of, the dataset whilst frequent itemsets are merely a collection of facts awaiting interpretation. A probability model can handle several kinds of probability queries, such as joint, marginal and conditional probabilities, whilst frequent itemset mining and association rule mining focuses only on high marginal and conditional probabilities. The prediction is made easy with a model. However, in order to predict with frequent itemsets, we still need to organize them and build a structured model first.

2. It is easier to observe interesting dependencies between the items, both positive and negative, from the model's parameters than it is to discriminate interesting itemsets or rules from the whole set of frequent itemsets or association rules. In fact, the parameters of the probability model trained from a dataset can be seen as a collection of features of the original data. Normally, the size of a probability model is far smaller than the set of frequent itemsets. Therefore the parameters of the model are highly representative. Useful knowledge can be obtained by simply "mining" the parameters of the model directly.

3. As the scale of the model is often smaller than the original data, it can sometimes serve as a proxy or a replacement for the original data. In real world applications, the original dataset may be huge and involve large time costs in querying or scanning the dataset. In such circumstances, if we just want an approximate estimation, a better choice is obviously to use the model to make the inference. As we will show in this thesis, when we want to predict all frequent itemsets, generating them from the model is much faster than mining them from the original dataset because the model prediction is irrelevant to the scale of the data. And Because the model is independent from the minimum frequency threshold, we only need to train the model once and can do the prediction on multiple thresholds but

consuming less time.

To demonstrate the advantage of the probability model more intuitively, we use an supermarket scenario as an example. Imagining the manager of a supermarket is reading the analysis of all the retailing transactions over the last year. If the analysis is based on the result of frequent itemset mining, it will be really hard to find out useful correlations. Because the frequent itemsets might just be a group of some very frequent but independent items. These kind of frequent itemsets are not informative. For example, bread and biscuit are both very popular items in grocery store, but buying them together very frequently does not mean bread and biscuit have special relationship. Meanwhile, some unexpected correlations of the merchandizes are hiding in the ocean of useless itemsets which makes people very hard to find out. The frequent itemset mining often generate huge number of frequent itemsets because the property of downward closure which we will discuss in Chapter 2. The downward closure property means if an itemset is frequent, all its subsets are frequent. By this property, the number of frequent itemsets grows exponentially which eventually makes the useful information difficult to be realized.

As a comparison, if the analysis is based on a accurate probability model, say the mixture model, we can easily obtain some information that cannot easily be obtained by the FIM. Firstly, as the model is a group of components, we can realize that the transactions can be categorized into different types. It is natural to think that the customers come to store for different targets such as food, clothing, magazines, electronic equipments etc. The model provides a possible explanation of the retailing data. The correlations of the items are also easy to find out. For example, if some items are highly positive correlated, then their conditional probabilities will be all quite high in some components and all quite low the the rest. If two items are repelling each other, then in some components one item's probability will be high and the other one's probability will be low and vice versa. Otherwise the items are not closely correlated. Generally, the size of the model is often smaller than the dataset and the set of frequent itemsets and the model is more organized, which makes people doing further analysis much more conveniently.

A powerful tool for building probability models is Bayesian inference which has experienced rapid developments during recent years. Compared with non-Bayesian machine learning methods, Bayesian approaches have several valuable

advantages. Firstly, Bayesian integration does not suffer from overfitting, because it does not fit parameters to the data; it integrates overall parameters and is weighted by how well they fit the data. Secondly, prior knowledge can be incorporated naturally and all uncertainty is manipulated in a consistent manner. One of the most prominent recent developments in this field is the application of the Dirichlet Process (DP) [Fer73] mixture model, a nonparametric Bayesian technique for mixture modelling, which allows for the automatic determination of an appropriate number of mixture components. Here, the term "nonparametric" means that the number of mixture components can grow automatically to the necessary scale. The DP is an infinite extension of the Dirichlet distribution which is the prior distribution for finite Bayesian mixture models. Therefore the DP mixture model can contain as many components as necessary to describe an unknown distribution. By using a model with an unbounded complexity, underfitting is mitigated, whilst the Bayesian approach of computing or approximating the full posterior over parameters mitigates over-fitting.

1.1 Objectives and Contributions

Our goal is to build Bayesian mixture models in order to improve the accuracy of current non-Bayesian mixture models for frequent itemset mining. As discussed, the Bayesian inference can improve the quality of the probability models for frequent itemset mining. More specifically, the main contributions of this work are the following:

1. Evolving the non-Bayesian mixture model to a Bayesian mixture model. The assumption and the structure of the Bayesian model is proposed. The corresponding algorithms for inference via sampling and variational approximation are also described. For the sampling approach, we implemented Gibbs sampling algorithm for the finite Bayesian mixture model (GSFBM) which is a multi-variant Markov Chain Monte Carlo (MCMC) sampling scheme. For the variational approximation, we implement the variational EM algorithm for the finite Bayesian mixture model (VFBM) by approximating the true posterior with a factorized distribution function.

2. Extending the finite Bayesian mixture model to the infinite. The Dirichlet process prior is introduced to the model so that the model obtains the ability

to fit a proper complexity itself. This model solves the problem of finding the proper number of components used in traditional probability models. For this model, we also implement two algorithms. The first one is Gibbs sampling for the Dirichlet Process mixture model (GSDPM). The second one is the truncated variational EM algorithm for the Dirichlet Process mixture model (VDPM). The word "truncated" means we approximate the model with a finite number of components.

3. The two main inference algorithms for Bayesian inference, including the variational Expectation Maximization algorithm and the Gibbs sampling, are applied and compared. Their accuracy, degree of under-estimation and time costs, both training time cost and the time cost for frequent itemset generation, are compared and analyzed.

1.2 Overview of the thesis

This thesis is organized as follows:

Chapter 2 introduces the related background to our work. First we review frequent itemset mining including basic concepts and terminologies, important mining algorithms and recent developments. In the second section we introduce the non-Bayesian mixture model and the Expectation-Maximization algorithm for inference. In Section 2.3 we describe the concept and key related properties of the Dirichlet distribution and the Dirichlet process. These two distributions are important for the Bayesian mixture models which will be introduced in Chapters 3 and 4. In the final section, we briefly introduce Monte Carlo Markov Chain (MCMC) sampling and its multivariant version, Gibbs sampling.

Chapter 3 introduces the finite Bayesian mixture model, including contrasts with the non-Bayesian model, assumptions and inference algorithms. For the finite Bayesian model, we apply both the Gibbs sampling algorithm and the variational approximation algorithm.

Chapter 4 introduces the Dirichlet process mixture model. Based on the finite Bayesian mixture model, we further extend it to the infinite. We describe how the Dirichlet process prior makes the mixture model grow automatically. As in Chapter 3, we apply two algorithms: the Gibbs sampling and the variational approximation algorithms.

Chapter 5 reports and analyzes the empirical results of the proposed models. We test the four algorithms and the non-Bayesian mixture model on five synthetic datasets and nine real datasets. The synthetic datasets are generated from mixture models. We use them to validate the performance of the inference algorithms and then run the five algorithms on the nine real datasets to see how well the models fit the data. Analysis and discussion feature in the final section.

Chapter 6 presents a discussion of the results. In this chapter, we draw some conclusions from the work described here and discuss some possible directions for future work.

Appendix describes in detail some of the computation within Chapters 3 and 4.

Chapter 2

Related Background

2.1 Introduction

In this chapter we introduce the background related to our research and organize it as follows. In Section 2.2, we briefly review the basic concept, algorithms and recent developments in frequent itemset mining. In Section 2.3, we focus on the non-Bayesian mixture model for binary datasets and introduce its inference algorithm: the Expectation-Maximization algorithm. In Section 2.4, we discuss the Dirichlet distribution and the Dirichlet process as the foundations of Chapters 3 and 4. In Section 2.5, we briefly introduce Markov Chain Monte Carlo (MCMC) sampling, another powerful inference tool we will use in our modeling. In the last section we briefly review the content of this chapter.

2.2 Frequent Itemsets Data Mining

Frequent itemsets mining was once a step within association rules mining and was first introduced by Agrawal in 1993 [AIS93]. The idea of association rules came from a supermarket model. Consider a supermarket selling a collection of items. Each customer buys several items and the managers want to understand customers' behaviour by analyzing each purchase transaction. Association rules describe how frequently the items are purchased together. For example, "90% of people who bought bread also bought milk" is an association rule which can be written as "bread⇒milk (90%)". This kind of information can help with business decisions such as optimal merchandise placement on the shelves or how to design promotional schemes etc.

In the process of association rules mining, frequent itemset mining plays an essential role as the base step. It discovers exactly how many times an item combination occurs in the transaction dataset and prunes infrequently occurring ones. Through research over the past decade, frequent itemset mining has become an important data mining technique. It can be used variously in classification, clustering, sequences and correlations [HCXY07, Goe03]. Various methods and algorithms for frequent itemset mining and its applications have been established. In this section, a brief introduction to frequent itemset mining will be given, including a formal definition, some common algorithms and current problems in this research area.

2.2.1 Definitions, Terminologies and Notations

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_D\}$ be the set of items, where D is the number of items. Set $X = \{i_{m_1}, i_{m_2}, \ldots, i_{m_k}\} \subseteq \mathcal{I}$ is called an *itemset* with length k, or a *k-itemset*.

A transaction dataset $\mathcal{T}$ is a set of transactions. We define transaction t as the following. A transaction t is an itemset with a unique number *tid* in the dataset as its primary key. Without ambiguity, we also use t to represent the itemset of the transaction.

A transaction dataset can also be seen as a binary matrix over the set of items. With this representation, transactions are binary vectors. For a certain transaction, the corresponding position is one if an item is purchased or zero if otherwise.

A transaction t is said to *support* an itemset X if and only if $X \subseteq t$. Then the *support* of an itemset is defined as

$$support(X, \mathcal{T}) := |\{t | X \subseteq t, t \in \mathcal{T}\}| \tag{2.1}$$

The *frequency* of an itemset is its *support* divided by the total number of transactions in the dataset:

$$f(X, \mathcal{T}) := \frac{support(X, \mathcal{T})}{|\mathcal{T}|} \tag{2.2}$$

An itemset is frequent if its support meets the minimal support threshold s_{min}. The minimal support threshold is an integer number between 0 and $|\mathcal{T}|$ and it is set by the users. The collection of all the frequent itemsets $\mathcal{F}(\mathcal{T}, s_{min})$

is defined as following.

$$\mathcal{F}(\mathcal{T}, s_{min}) := \{X | X \subseteq \mathcal{I}, support(X, \mathcal{T}) \geq s_{min}\} \tag{2.3}$$

Sometimes we also use relative minimal frequency threshold f_{min} and the frequencies of itemsets for frequent itemsets mining. In many cases *frequency* is more convenient than *support*. f_{min} can be directly calculated by

$$f_{min} = \frac{s_{min}}{|\mathcal{T}|} \tag{2.4}$$

With f_{min}, we can also define $\mathcal{F}$ as

$$\mathcal{F}(\mathcal{T}, f_{min}) := \{X | X \subseteq \mathcal{I}, frequency(X, \mathcal{T}) \geq f_{min}\} \tag{2.5}$$

Definition 2.1 (Frequent Itemset Mining)**.** *Given a transaction dataset $\mathcal{T}$ over a set of items $\mathcal{I}$, and a minimal support threshold s_{min}, find $\mathcal{F}(\mathcal{T}, s_{min})$ and the support of each frequent itemset.*

Normally, we are also interested in the exact support or the frequency values of the frequent itemsets.

An *association rule* is an expression of form $X \Rightarrow Y$, where X and Y are itemsets, and $X \cap Y = \emptyset$. An association rule expresses how likely that it would be if a transaction contains the itemset X, it also contains the itemset Y. An association rule has two attributes: *support* and *confidence*. The *support* is the *support* of $X \cup Y$ and the *confidence* is

$$confidence(X \Rightarrow Y, \mathcal{T}) := \frac{support(X \cup Y, \mathcal{T})}{support(X, \mathcal{T})} \tag{2.6}$$

The rule is called *confident* if it meets a minimal confidence threshold c_{min}. The collection of all the association rules $\mathcal{R}(\mathcal{T}, s_{min}, c_{min})$ is defined as following.

$$\begin{aligned}\mathcal{R}(\mathcal{T}, s_{min}, c_{min}) := \{X \Rightarrow Y | X, Y \subseteq \mathcal{I}, X \cap Y = \emptyset, X \cup Y \in \mathcal{F}(\mathcal{T}, s_{min}),\\ confidence(X \Rightarrow Y, \mathcal{T}) \geq c_{min}\}\end{aligned} \tag{2.7}$$

Definition 2.2 (Association Rule Mining)**.** *Given a transaction dataset $\mathcal{T}$ over a set of items $\mathcal{I}$, a minimal support threshold s_{min} and a minimal confidence threshold c_{min}, find $\mathcal{R}(\mathcal{T}, s_{min}, c_{min})$ and the support and the confidence of each*

association rule.

In the following part of this section, we use an example to demonstrate the concepts described above.

Example 2.1 (Transaction Dataset)**.** *Consider the transaction dataset in Table 2.1 over the set of items*

$$\mathcal{I} = \{Apple, Bread, Coke, Diaper, Eggs, Fish\}$$

Mine all the frequent itemsets with respect to a minimal support threshold of 2

tid	Itemset	Apple	Bread	Coke	Diaper	Eggs	Fish
1	{Apple, Bread}	1	1	0	0	0	0
2	{Apple, Coke, Diaper, Eggs}	1	0	1	1	1	0
3	{Bread, Coke, Diaper, Fish}	0	1	1	1	0	1
4	{Apple, Bread, Coke, Diaper}	1	1	1	1	0	0
5	{Apple, Bread, Coke, Fish}	1	1	1	0	0	1

Table 2.1: Example Transaction Dataset

and all association rules with respect to a minimal confidence threshold of 60%.

The right side of Table 2.1 shows the binary matrix representation of the transaction dataset. In the matrix, transactions are represented as row vectors. Table 2.2 shows all the frequent itemsets of Example 2.1. The searching strategy and algorithms will be discussed in later sections. Table 2.3 shows all the association rules of Example 2.1. We do not plan to discuss the searching of association rules as the process based on the results of frequent itemset mining is quite straightforward and there is little work that can be done to improve it [BMS97].

In frequent itemsets mining, we do not need to consider the item ordering in the transactions. However, for the sake of clarity and convenience, the items in the transactions and itemsets are often sorted by a fixed given order. The order could be item frequency descending order or lexicographic order. Whether or not the items are sorted does not alter the result of the mining, but a fixed order would make the mining process simpler and faster.

The relationship between frequent itemset mining and association rule mining depends on the researchers' interest. Frequent itemset mining can be viewed as the main step in association rule mining while association rule mining can be

Itemset	tids	*support*	*frequency*
{Apple}	{1,2,4,5}	4	0.8
{Bread}	{1,3,4,5}	4	0.8
{Coke}	{2,3,4,5}	4	0.8
{Diaper}	{2,3,4}	3	0.6
{Fish}	{3,5}	2	0.4
{Apple, Bread}	{1,4,5}	3	0.6
{Apple, Coke}	{2,4,5}	3	0.6
{Apple, Diaper}	{2,4}	2	0.4
{Bread, Coke}	{3,4,5}	3	0.6
{Bread, Diaper}	{3,4}	2	0.4
{Bread, Fish}	{3,5}	2	0.4
{Coke, Diaper}	{2,3,4}	3	0.6
{Coke, Fish}	{3,5}	2	0.4
{Apple, Bread, Coke}	{4,5}	2	0.4
{Apple, Coke, Diaper}	{2,4}	2	0.4
{Bread, Coke, Diaper}	{3,4}	2	0.4
{Bread, Coke, Fish}	{3,5}	2	0.4

Table 2.2: Frequent Itemsets of Example 2.1

viewed as an extension of frequent itemset mining. Generally, the research on frequent itemset mining is more fundamental and extendable. After decades of development, the research area has moved far beyond the restrictions of "frequent" and "itemsets" to a wide range of measurements and objectives such as χ^2 [BMS97] and *lift* [Mac96], as replacements for *support* and sequential mining [AS95], and graph mining [WM03] as an extension of itemsets mining.

To close this section, we will talk more about the main sources and backgrounds of such transaction datasets in the real world. Retail records and website logs are two examples of transaction datasets. For example, in a retail application, the rows of data correspond to purchases made by various customers, whilst the columns correspond to different items for sale in the store. This kind of data is often sparse; i.e. while there may be thousands of items for sale, a typical transaction may only contain a small fraction of the available merchandise. Another important source of such transaction datasets is discrete or discretized classification datasets which are transformed into the form of binary data. A strong characteristic of this kind of dataset is that some attributes or items are mutually exclusive; i.e. some groups of items never co-occur in any transactions. Datasets in this class are often relatively denser than web log or retail datasets.

Rule	*support*	*confidence*
$\{Apple\} \Rightarrow \{Bread\}$	3	75%
$\{Bread\} \Rightarrow \{Apple\}$	3	75%
$\{Apple\} \Rightarrow \{Coke\}$	3	75%
$\{Coke\} \Rightarrow \{Apple\}$	3	75%
$\{Diaper\} \Rightarrow \{Apple\}$	2	67%
$\{Bread\} \Rightarrow \{Coke\}$	3	75%
$\{Coke\} \Rightarrow \{Bread\}$	3	75%
$\{Diaper\} \Rightarrow \{Bread\}$	2	67%
$\{Fish\} \Rightarrow \{Bread\}$	2	100%
$\{Coke\} \Rightarrow \{Diaper\}$	3	75%
$\{Diaper\} \Rightarrow \{Coke\}$	3	100%
$\{Fish\} \Rightarrow \{Coke\}$	2	100%
$\{Bread, Coke\} \Rightarrow \{Apple\}$	2	67%
$\{Apple, Coke\} \Rightarrow \{Bread\}$	2	67%
$\{Apple, Bread\} \Rightarrow \{Coke\}$	2	67%
$\{Coke, Diaper\} \Rightarrow \{Apple\}$	2	67%
$\{Apple, Diaper\} \Rightarrow \{Coke\}$	2	100%
$\{Diaper\} \Rightarrow \{Apple, Coke\}$	2	67%
$\{Apple, Coke\} \Rightarrow \{Diaper\}$	2	67%
$\{Coke, Diaper\} \Rightarrow \{Bread\}$	2	67%
$\{Bread, Diaper\} \Rightarrow \{Coke\}$	2	100%
$\{Diaper\} \Rightarrow \{Bread, Coke\}$	2	67%
$\{Bread, Coke\} \Rightarrow \{Diaper\}$	2	67%
$\{Coke, Fish\} \Rightarrow \{Bread\}$	2	100%
$\{Bread, Fish\} \Rightarrow \{Coke\}$	2	100%
$\{Fish\} \Rightarrow \{Bread, Coke\}$	2	100%
$\{Bread, Coke\} \Rightarrow \{Fish\}$	2	67%

Table 2.3: Association Rules of Example 2.1

An important correlation which data mining seeks to elucidate is which items co-occur in purchases and which items are mutually exclusive, never (or rarely) co-occurring in transactions. This information allows for the performance of prediction or further analysis.

2.2.2 Frequent Itemset Mining Algorithms

In this section we introduce some well known and commonly used algorithms for frequent itemset mining. They are Apriori [AS94], Eclat [Zak00] and FP-growth [HPY00]. As the ideas expressed by Apriori and Eclat are also useful in the

process of frequent itemsets generation by probability models, we will give these more detailed introductions.

Apriori Algorithm

Apriori is the earliest and most fundamental algorithm. Proposed by Agrawal in 1994 [AS94], this algorithm follows the simple but powerful Apriori principle. This principle's key idea is that *support* is monotone, decreasing with respect to the extension of an itemset. We express it in a more formal way.

Proposition 2.1 (Apriori Principle)**.** *Given a transaction dataset $\mathcal{T}$ over the set of items $\mathcal{I}$, let $X, Y \subseteq \mathcal{I}$ be two itemsets. If $X \subseteq Y$, then $support(X) \geq support(Y)$*

Proof. The transactions containing Y always contain X because X is a subset of Y.

$$\begin{aligned} X \subseteq Y &\Rightarrow \{t | X \subseteq t, t \in \mathcal{T}\} \supseteq \{t' | Y \subseteq t', t' \in \mathcal{T}\} \\ &\Rightarrow support(X) \geq support(Y) \end{aligned}$$

□

With proposition 2.1, we can immediately conclude that: a k-itemset is frequent only if all its $k-1$ sub-itemsets are frequent. In other words, if an itemset is infrequent, all its super itemsets are infrequent. This property could greatly reduce the searching space and accelerate the speed of mining. We follow a step-by-step way of mining frequent itemsets. First, we scan the dataset to find all frequent singletons. Then we use all the frequent singletons to generate candidate frequent pairs and scan the dataset again to check and obtain the frequent pairs. Next, we use all the frequent pairs to generate candidate frequent triples and scan the dataset for validation. This process iterates until no further candidate k-itemset can be generated by frequent $k-1$-itemsets. This is the key idea of the Apriori algorithm and its alternatives. We use the Apriori algorithm to mine the dataset in Example 2.1 to demonstrate more clearly.

Example 2.2 (Apriori Algorithm)**.** *Use the Apriori algorithm to mine the transaction dataset shown in Table 2.1.*

The first step should be to find out all frequent singletons. All singletons with their *support* are shown in Table 2.4. The *support* of "Eggs" is 1, so we discard this item from further mining.

Itemset	tids	*Support*
{Apple}	{1,2,4,5}	4
{Bread}	{1,3,4,5}	4
{Coke}	{2,3,4,5}	4
{Diaper}	{2,3,4}	3
~~{Eggs}~~	~~{2}~~	~~1~~
{Fish}	{3,5}	2

Table 2.4: Candidate and Frequent Singletons in Example 2.1

The second iteration is to mine all pairs. We combine all the frequent singletons to generate candidate pairs and count their support. Table 2.5 shows all the candidate and frequent pairs. The number of candidate pairs is $\binom{5}{4} = 10$. Two of them are infrequent and are thus filtered.

Itemset	tids	*Support*
{Apple,Bread}	{1,4,5}	3
{Apple,Coke}	{2,4,5}	3
{Apple,Diaper}	{2,4}	2
~~{Apple,Fish}~~	~~{5}~~	~~1~~
{Bread,Coke}	{3,4,5}	3
{Bread,Diaper}	{3,4}	2
{Bread,Fish}	{3,5}	2
{Coke,Diaper}	{2,3,4}	3
{Coke,Fish}	{3,5}	3
~~{Diaper,Fish}~~	~~{3}~~	~~1~~

Table 2.5: Candidate and Frequent Pairs in Example 2.1

The third iteration is for all triples. When generating candidate itemsets longer than 2, the potential searching space grows rapidly. The Apriori algorithm prunes the candidate itemsets by applying Proposition 2.1. More specifically, the candidate generation has two steps: the *join* step and the *prune* step. Assume we are generating candidate k-itemsets by the set of frequent $k-1$-itemsets $\mathcal{F}_{k-1}$. In the *join* step we first find out all frequent $k-1$-itemset pairs in which the two itemsets of pairs share the identical first $k-2$ items except the last ones. Union operation is then applied to the itemsets of each pair to form a candidate

itemset. In the *prune* step, we check every candidate itemset for whether or not all its $k-1$ subsets are frequent. If not, we remove it from the candidate set. The *prune* step aims to decrease the workloads involved in scanning the dataset and counting as much as possible. After the *prune* step, we finally scan the dataset to obtain the frequent k-itemsets and their *support*.

In our example, the *join* step will generate seven candidate triples. Two of them are removed in the *prune* step because their sub-itemset $\{Diaper, Fish\}$ is infrequent. Table 2.6 shows the candidate and frequent itemsets of this iteration. The last two itemsets were pruned in the *prune* step without counting their *support*.

Itemset	tids	*support*
{Apple, Bread, Coke}	{4,5}	2
~~{Apple, Bread, Diaper}~~	~~{4}~~	~~1~~
{Apple, Coke, Diaper}	{2,4}	2
{Bread, Coke, Diaper}	{3,4}	2
{Bread, Coke, Fish}	{3,5}	2
~~{Bread, Diaper, Fish}~~		
~~{Coke, Diaper, Fish}~~		

Table 2.6: Candidate and Frequent Triples in Example 2.1

For quadruples we can still generate the itemset $\{Bread, Coke, Diaper, Fish\}$ by the *join* step, but since $\{Coke, Diaper, Fish\}$ and $\{Bread, Diaper, Fish\}$ are not frequent, this itemset is discarded in the *prune* step. As we cannot generate any other candidate itemsets, the mining process is therefore complete.

The example above showed the entire process of the Apriori algorithm. To summarize Apriori, the pseudo code [Goe03] is given by Algorithm 2.1.

Eclat Algorithm

The Equivalence CLAss Transformation (Eclat) algorithm was proposed by Zaki [Zak00]. Its main features are its use of the vertical data format and lattice decomposing. We continue to use Example 2.1 to describe these concepts and the Eclat algorithm.

As we showed in Table 2.1, a transaction dataset can be expressed as a binary matrix with each row vector recording each transaction and each column vector recording the occurrence situation of each item. We call these column vectors the *tid_list*s of the items. Moreover, every itemset has its *tid_list*. However, for the

Algorithm 2.1 Apriori Itemset-Mining

Input: $\mathcal{T}, s_{min}$
Output: $\mathcal{F}(\mathcal{T}, s_{min})$
  $C_1 := \{\{i\} | i \in \mathcal{I}\}$
  $k := 1$
  while $C_k \neq \emptyset$ **do**
    // Compute the supports of all candidate itemsets
    for all transaction $t \in \mathcal{T}$ **do**
      for all candidate itemsets $X \in C_k$ **do**
        if $X \subseteq t$ **then**
          $X.support := X.support + 1$
        end if
      end for
    end for
    // Extract all frequent itemsets
    $\mathcal{F}_k := \{X | X.support \geq s_{min}, X \in C_k\}$
    // Generate new candidate itemsets
    $C_{k+1} := \emptyset$
    for all $X, Y \in \mathcal{F}_k, X[i] = Y[i]$ for $1 \leq i \leq k-1$, and $X[k] < Y[k]$ **do**
      $I = X \cup \{Y[k]\}$
      if $\forall J \subset I, |J| = k : J \in \mathcal{F}_k$ **then**
        $C_{k+1} = C_{k+1} \cup I$
      end if
    end for
    $k = k + 1$
  end while
  $\mathcal{F}(\mathcal{T}, s_{min}) := \bigcup \mathcal{F}_k$

k-itemsets with $k > 1$, their *tid_list*s cannot be explicitly obtained by the origin dataset. We use a set expression to render the concept of *tid_list* more formally.

$$L(X) = \{tid | (tid, t) \subset \mathcal{T}, X \subseteq t\}$$

Proposition 2.2. *Given itemsets* $X, Y \subseteq \mathcal{I}$,

$$L(X \cup Y) = L(X) \cap L(Y)$$

Proof. First we prove $L(X \cup Y) \subseteq L(X) \cap L(Y)$.

$$\begin{aligned} X \subseteq X \cup Y &\Rightarrow \{t | X \subseteq t, t \in \mathcal{T}\} \supseteq \{t' | X \cup Y \subseteq t', t' \in \mathcal{T}\} \\ &\Rightarrow L(X) \supseteq L(X \cup Y) \end{aligned}$$

For the same reason, we have $L(Y) \supseteq L(X \cup Y)$. Therefore,

$$L(X) \cap L(Y) \supseteq L(X \cup Y) \tag{2.8}$$

Next we prove $L(X \cup Y) \supseteq L(X) \cap L(Y)$.

$$\begin{aligned} \forall t \in L(X) \cap L(Y) &\Rightarrow X \subseteq t, Y \subseteq t \Rightarrow X \cup Y \subseteq t \Rightarrow t \in L(X \cup Y) \\ &\Rightarrow L(X) \cap L(Y) \subseteq L(X \cup Y) \end{aligned} \tag{2.9}$$

With Equation 2.8, 2.9, we have $L(X \cup Y) = L(X) \cap L(Y)$. □

Proposition 2.2 provides a new method of support counting by maintaining and operating with *tid_list*s instead of scanning the dataset multiple times. When generating a new candidate itemset, a union operation is applied to two chosen frequent itemsets. Meanwhile an intersection operation is applied to their *tid_list*s to obtain the new candidate itemset's *tid_list*. The cardinality of the *tid_list* is the support of the candidate itemset.

Another of the improvements of Eclat in comparison with Apriori is the way in which it organizes the process of candidate itemsets generation. Eclat uses a lattice of itemsets to demonstrate the structure of the set of all itemsets. Figure 2.1, is a graphic presentation of the itemset lattice of Example 2.1. In Eclat, a new candidate k-itemset is generated by joining two $k-1$-itemsets which share the same $k-2$-length prefix. The lines in 2.1 represent the process of candidate generation. Obviously, if one of the parent itemsets is infrequent, we do not base a candidate itemset on it in practice. In Figure 2.1 we use a dashed line for such unnecessary generations. In Eclat, there is no *prune* step. Therefore, in comparison with Apriori, this operation certainly generates more candidate itemsets. However, the benefit of this operation is that it makes the mining process more flexible. For example, when we generate the itemset "ABC" by itemsets "AB" and "AC", we do not need to consider the situation of "BC". Thus the generation of "ABC" only involves itemsets within the A-prefix lattice. Additionally, in the generation of itemset "BCDF", we do not need to check its

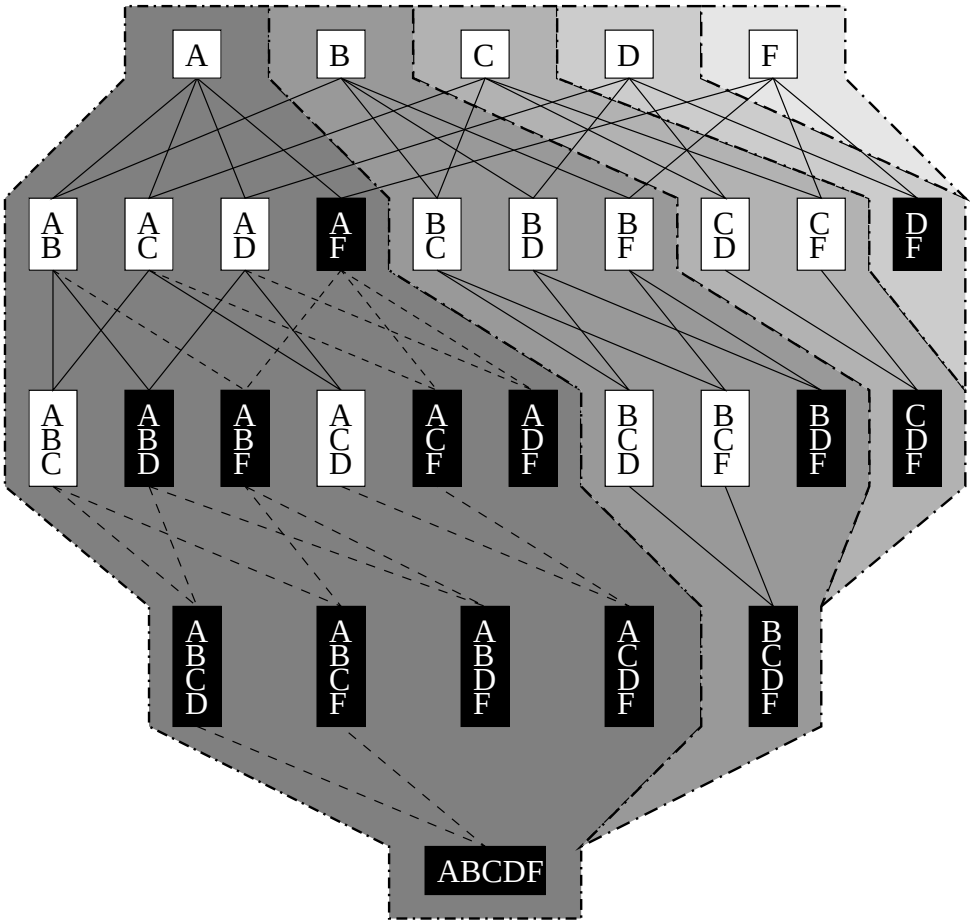


Figure 2.1: Itemset Lattice of Example 2.1. The white boxes are frequent itemsets and the black boxes are infrequent ones. The lines connecting the itemsets in different layers show how the longer itemsets are generated by shorter ones. The regions of different colors show how the whole lattice is decomposed into smaller sub-lattices.

sub-itemsets outside of the BC-prefix lattice. Furthermore, the mining process based on this operation forms a way to decompose the lattice into smaller sub-lattices, thus making a depth-first search possible.

The Eclat process follows a recursive depth-first search strategy. The whole itemsets lattice is decomposed into smaller sub-lattices and the sub-lattices continue to decompose until no further decomposition can take place. Thus, in each recursive mining step, the output is the frequent itemsets of the given sub-lattice or, in other words, those with the same prefix. Here we use $\mathcal{F}[I](\mathcal{T}, s_{min})$ to denote the set of frequent itemsets with the prefix I in dataset $\mathcal{T}$ with the given minimum support threshold s_{min}.

The pseudo code of Eclat [Goe03] is given by Algorithm 2.2. As the process is recursive, we only need to describe the mining process of a given sub-lattice. The final mining result can be written as $\mathcal{F}[\emptyset](\mathcal{T}, s_{min})$ therefore it is merely a special case of $\mathcal{F}[I](\mathcal{T}, s_{min})$.

Algorithm 2.2 Eclat

Input: $\mathcal{T}, s_{min}$
Output: $\mathcal{F}[I](\mathcal{T}, s_{min})$
  $\mathcal{F}[I](\mathcal{T}, s_{min}) := \emptyset$
  for all $i \in \mathcal{I}$ occurring in $\mathcal{T}$ **do**
    $\mathcal{F}[I] := \mathcal{F}[i] \cup \{I \cup \{i\}\}$
    //Create sub-lattice $\mathcal{T}^i$
    $\mathcal{T}^i := \emptyset$
    for all $j \in \mathcal{I}$ occurring in $\mathcal{T}$ such that $j > i$ **do**
      $tid\_list(I \cup \{i, j\}) = tid\_list(I \cup \{i\}) \cap tid\_list(I \cup \{j\})$
      if $|tid\_list(I \cup \{i, j\})| > s_{min}$ **then**
        $\mathcal{T}^i = \mathcal{T}^i \cap \{(j, tid\_list(I \cup \{i, j\}))\}$
      end if
    end for
    //Depth-first recursion
    Compute $\mathcal{F}[I \cup \{i\}](\mathcal{T}^i, s_{min})$
    $\mathcal{F}[I] := \mathcal{F}[i] \cup \mathcal{F}[I \cup \{i\}](\mathcal{T}^i, s_{min})$
  end for

Other Algorithms and Extensions

Another important algorithm for frequent itemset mining is FP-growth [HPY00]. Its key feature is that the process does not need candidate generation and validation. The algorithm can output the frequent itemset directly. More specifically, FP-growth uses the FP-tree structure to compress all the information of the frequent items and then generates the frequent itemsets by a set of well-designed tree operations.

There are many alternatives and extensions of the above three algorithms: For Apriori, these include hashing technique [PCY95], partition technique [SON95], sampling approach [Toi96] and dynamic itemset counting [BMUT97]. For Eclat there is dEclat [ZG03] and for FP-growth there is H-mine [PHMA+01].

Frequent itemset mining is a fact-extraction process. Given the dataset and the minimum support threshold, the result of all data mining algorithms should be the same. The differences between different algorithms lie in their memory

and time cost. Generally, given the significant differences among datasets, no algorithm's performance overwhelms all other algorithms in all cases. However, among the three mainstream algorithms, Eclat and FP-growth achieve a better average time performance than Apriori with some extra memory cost [Goe03].

2.2.3 Problems and Developments

The aim of frequent itemset mining is to discover correlations in the datasets. In practice, however, the results do not fully meet people's expectations. There are several problems which influence the quality and the utility of the result. In this section we briefly introduce the current problems and techniques developed for solving them.

Mining Compressed Itemsets

One of the most significant problems is that the result set is often of huge scale, causing difficulties in distinguishing the useful information. According to the Apriori principle, each of a frequent itemset's sub-itemsets are frequent, which means a large frequent itemset contains an exponential number of smaller, frequent itemsets. For example, if a frequent itemset has a length of 10, all its $2^{10} - 2$ sub-itemsets are frequent. For some dense datasets, the cardinality of the set of frequent itemsets is more than 1 million. Without a proper quantizing standard, it is impossible to filter out useless itemsets. In recent years, people have proposed several methods for dealing with this issue, such as mining closed itemsets [PBTL99, PHM00, ZjH02, GZ03, LLLY03], maximal itemsets [Bay98, BCG01] and compressed or approximate itemsets [CG02, CG05, GLSS06, WHM+05, PDZH02, AGM04, XHYC05, YCHX05, WP06]. The following is a brief introduction to these works.

An itemset X is a *closed frequent itemset* in a dataset $\mathcal{T}$ if X is frequent in $\mathcal{T}$ and there exists no proper super-itemset Y such that Y has the same *support* as X in $\mathcal{T}$. By definition, if a frequent itemset X is not closed, then there exists at least one of its super-itemsets which has the same *support* as X. For example, in Example 2.1 the itemsets $\{Fish\}$, $\{Bread, Fish\}$ and $\{Coke, Fish\}$ are non-closed frequent itemsets because their *support*s are all 3 and their super-itemset $\{Bread, Coke, Fish\}$ also has a *support* of 3. In this case, we can use

the *support* of $\{Bread, Coke, Fish\}$ to deduce the *support* of its non-closed sub-itemsets. The remaining frequent itemsets are all closed itemsets. It has been shown that if we have a set of closed frequent itemsets, we can obtain the whole set of frequent itemsets without loss [PBTL99]. The set of closed frequent itemsets is a lossless compression of the normal set of frequent itemsets. Mining closed frequent itemsets can reduce the number of frequent itemsets. However, the definition of closed frequent itemsets implies that a non-closed frequent itemset needs to have a very strong correlation with one of its super-itemsets, which is very rare in most data mining tasks. Therefore the reduction degree of the result scale is limited. Despite this, better scalability and interpretability is achieved with closed itemset mining [HCXY07].

An itemset X is a *maximal frequent itemset* in a dataset $\mathcal{T}$ if X is frequent in $\mathcal{T}$ and there exists no super-itemset Y such that $X \subset Y$ and Y are frequent. In Example 2.1, the itemsets $\{Apple, Bread, Coke\}$, $\{Apple, Coke, Diaper\}$, $\{Bread, Coke, Diaper\}$ and $\{Bread, Coke, Fish\}$ are maximal frequent itemsets. According to the Apriori principle, with the set of maximal frequent itemsets we can obtain all frequent itemsets except their *support*s. Obviously all maximal frequent itemsets are closed frequent itemsets. The set of maximal itemsets is far more compact than the set of closed itemsets though the information regarding *support* is lost. Research in maximal frequent itemsets improves the theoretical analysis of the complexity of frequent itemset mining [Yan04] and the distribution of the frequent itemsets [RMZ03]. However, as the *support* information is lost, it can hardly be used in practical data mining tasks.

Mining compressed itemsets has been forwarded as one possible means of reducing the huge set of frequent itemsets while maintaining high quality in the results. Generally, mining compressed itemsets can be categorized into two types: lossless compression and lossy compression - these are in terms of the information that the result set contains compared with the whole set of frequent itemsets. For example, mining closed itemsets is a method of lossless compression since all the frequent itemsets can be derived from the closed itemsets. Alternatively, mining maximal itemsets is lossy because the *support* information is lost during the mining process.

Lossless compression tries to find the most compact form or structure to contain all the information of the frequent itemsets and to prune the redundant information as much as possible. [CG02, CG05] tries to mine all non-derivable

frequent sets. Similarly with the closed itemsets, the whole set of frequent itemsets can be derived by the result set of these methods. [LLWH06] proposed the concepts of "frequent generator" and "positive border", being two types of special frequent itemset. From the frequent generators and the positive border, one can judge whether an itemset is frequent and calculate its *support* if this is the case.

Lossy compression aims to find a representation that is both compact and representative while tolerating some accuracy loss. The top-k most frequent closed itemsets proposed by [WHM+05] mine the most frequent k closed itemsets of length no less than min_l using an algorithm called **TFP**. However, due to uneven frequency distribution among the itemsets, the top-k most frequent closed itemsets are usually not the most representative k itemsets.

Another approach is to summarize the frequent itemsets as k representatives which cover the whole set of (closed) frequent itemsets. This summarization provides a compact compression of the collection of the frequent itemsets, making it easy to interpret and use. [AGM04] proposed using k itemsets to approximate a whole collection of frequent itemsets. The measure of the quality of the approximation is the size of the whole collection covered by the k itemsets and their spanning. It is a summarization of the maximal itemsets for representing as many itemsets as possible with a fixed size collection of itemsets. In this approach, the support information of the itemsets is ignored for the sake of compactness.

[XHYC05] tried to take the support information into account to make the result more representative. A distance measure for measuring the distance of the closed itemsets by analyzing the degree of overlapping among their supporting transactions was proposed. This idea is inspired by relaxing the concept of closed itemsets for achieving a more compact solution. In the definition of the closed itemsets, an itemset is non-closed if one of its super-itemsets has exactly the same *support*. In other words, their support transaction sets are fully overlapped. In this case, the non-closed itemset is redundant. From this point of view, the distance of the frequent itemsets can be defined by how they are supported by the transactions - the magnitude of overlapping in their supporting transactions. Based on the distance measure, they define that an itemset X can be represented by another itemset Y if $X \subset Y$ and the distance between X and Y is not greater than a given threshold. Furthermore, **RPglobal** and **RPlocal** algorithms are developed to generate compressed itemsets from the closed frequent itemsets.

Based on the definition of the distance measure of the closed frequent itemsets, [YCHX05] proposed a profile-based summarization method to summarize the (closed) frequent itemsets into k representatives. They further developed the definition of the distance between two itemsets as the KL divergence of the probability distributions of the two itemsets. The concept of the "profiles" is also relaxed from the itemsets. A profile not only contains a combination of the items, called *master pattern*, and a *support*, but also assigns a probability for each item contained in the *master pattern*. Compared with [XHYC05], [YCHX05] give more detailed information about the distributions of the items in each profile. The k representatives can be regarded as a probability model based on the frequent itemsets.

Another summarization approach is to build *Markov Random Field* from the frequent itemsets [WP06]. This model summarizes the frequent itemsets via a graphical model called *Markov Random Field*. The idea is to use frequent itemsets in a level-wise fashion to adjust the correlation represented by the model. Ultimately, all frequent itemsets can be approximately inferred by the model while the model is much more compact and interpretable than the original collection of the frequent itemsets.

Measurement of Interestingness

The above work about compression and summarization are all based on the objective property of the frequent itemsets. Another direction of improving the quality and reducing the quantity of the frequent itemsets are developing new "interestingness" measures such as [BMS97, AY98, RJBAG99, Omi03, Jar04, JS05, Web07, Tat08]. The rest of this section will give a brief introduction of these work.

Other than *support* and *confidence*, some useful simple measures are *lift* [Mac96] and *leverage* [PS91]. The *lift* of an association rule $X \rightarrow Y$ is the ratio of the confidence and the frequency of Y:

$$lift(X \rightarrow Y) = \frac{confidence(X \rightarrow Y)}{frequency(Y)} = \frac{confidence(X \rightarrow Y)}{confidence(\emptyset \rightarrow Y)}$$

lift is a comparison between Y's conditional probability given X with its marginal probability to show whether X has a positive or negative influence to Y. *lift* is greater than 1 if X has positive influence to Y, is smaller than 1 if X has

negative influence and is equal to 1 if X and Y are independent. The *leverage* of an association rule is defined as following:

$$\begin{aligned} leverage(X \to Y) &= support(X \cup Y) - \frac{support(X) \times support(Y)}{N} \\ &= N(frequency(X \cup Y) - frequency(X) \times frequency(Y)) \end{aligned}$$

leverage tries to show the difference between the empirical frequency of $X \cup Y$ and the expectation frequency of $X \cup Y$ assuming X and Y are independent. Similar to *lift*, *leverage* tries to capture the strength of the correlation between X and Y by comparing the empirical situation with the independent assumption.

A more complicated interestingness measure is *Chi-squared statistics* proposed by [BMS97]. It used the *Chi-squared Test* as a measure of independence and developed an efficient algorithm to discover all the itemsets that can both pass the *support* condition and the *Chi-squared Test*. The *Chi-squared Test* for a given itemset X is firstly to build the corresponding **contingency table** R by counting the *support* counts of all possible values (0 or 1) of the items occurring in X. Then for each cell r of R, the value χ^2 is computed by:

$$\chi^2 = \sum_{r \in R} \frac{(f(r) - E(r))^2}{E(r)}$$

where $f(r)$ is the *support* count of the cell r and $E(r)$ is the expected value of r calculated under the assumption of full independence. Compared to the support-confidence framework, the *Chi-squared Test* is more solidly grounded in statistical theory. A further good property of χ^2 is upward closure which means for a certain itemset X, if $\chi^2_X \geq \chi^2_{min}$, then $\chi^2_Y \geq \chi^2_{min}$ for all $Y \subseteq X$.

Aggarwal and Yu reviewed the *support-confidence* framework and proposed the *strongly collective itemset model* [AY98]. This model tried to develop a good measurement for the correlations but more computational efficiency than χ^2. The idea is to classify the relationship between the itemsets and the transactions into two categories: 1) itemsets fully occurring or fully not occurring in a transaction; 2) itemsets partly occurring in a transaction. The first category implies a possible positive correlation while the second category implies a possible negative correlation. The idea is further described as the concepts of *violation* and *violation rate*. An itemset X is said to be in *violation* of a transaction if some of the items are present in the transaction and others are not. The *violation rate* of an itemset

X is denoted by $v(X)$ and is the fraction of violation of the itemset X over all transactions. From a perspective of trying to establish high correlation among the corresponding items in an itemset, the violation of an itemset in a transaction can be understood as a "bad event". Its opposite situation is therefore seen as a "good event". The *collective strength* of an itemset X is defined as the following:

$$CS(X) = \frac{1 - v(X)}{1 - E[v(X)]} \cdot \frac{E[v(X)]}{v(X)}$$

where $E[v(X)]$ is the expected value of $v(X)$ calculated assuming the items are statistically independent. The *collective strength* ranges from 0 to inf with 0 meaning perfect negative correlation and inf meaning perfect positive correlation. They also proved that the *collective strength* has a closure property similar to *support*. Finally, they developed a very fast approximate algorithm to mine the strongly collective itemsets.

Bayardo, Agrawal and Gunopulos proposed the concept of *improvement* [RJBAG99] of an association rule in order to reduce the redundancy of association rules such as $\{pregnant, female\} \rightarrow \{oedema\}$ where $\{female\}$ is redundant. The *improvement* is defined as:

$$improvement(X \rightarrow Y) = confidence(X \rightarrow Y) - \max_{Z \subset X}(confidence(Z \rightarrow Y))$$

A *minimum improvement* is introduced to filter the redundant association rules. Webb [Web07] reviewed the concept and used the term *productive* to denote rules with positive *improvement*. A training-testing framework is also introduced to the significant rule discovery process by using part of the data to explore the candidate rules and using the rest of the data for statistical evaluation of the candidate rules. This framework is put in place to prevent false discovery of the rules, or in other words, overfitting.

[Omi03] reviewed the concept of *confidence* and developed two new metrics; *All-confidence* and *Bond*. Using the measure of *All-confidence*, an itemset is interesting if **all** the rules produced from this itemset have a *confidence* greater or equal to the minimal confidence threshold. We use the concept of *violation rate* defined in [AY98] to describe *Bond*. For an itemset X, its *Bond* is the ratio

of its *frequency* and its *frequency* plus its *violation rate*:

$$Bond(X) = \frac{frequency(X)}{frequency(X) + v(X)}$$

A key contribution both of *All-confidence* and *Bond* is that they also hold the property of downward closure along with *support*, solving the problem that *confidence* cannot efficiently be used in searching algorithms.

[Jar04] demonstrated a method of incorporating background knowledge to discover unexpected interesting itemsets. They used a Bayesian network, which is a directed graph representing the conditional dependencies of the items, to represent the prior expectations. The interestingness of an itemset is therefore defined as the absolute difference of the frequency of the itemset estimated from the data and the estimation of this itemset from the Bayesian network. In this paper, the definition of an itemset was slightly different from the common asymmetric one which only counts the appearance of the items. By their definition, an itemset is a collection of attributes in which each attribute can be both 0 and 1. Furthermore, they defined the interestingness of an attribute set as the maximal interestingness in all possible value combinations of the attributes. In [JS05], this interestingness definition was further developed in the problem of finding the n most interesting attribute set and finding the n approximate most interesting attribute set. A series of corresponding algorithms was also proposed for mining these interesting attribute sets.

N. Tatti proposed that the significance of the itemsets can be ranked by a *Maximal Entropy model* [Tat08]. A *Maximal Entropy model* is a distribution of the items which maximizes the entropy with the frequent itemsets as the constraints. For a given frequent itemset family F_G, in which every itemset $X \in F_G$ satisfies $X \subset G$, a distribution of G is said to satisfy the constraint of F_G if for each itemset $X \in F_G$ and its frequency θ_X:

$$p(X) = \theta_X$$

For all the distributions satisfying the constraint of F_G, we denote the distribution with the maximal entropy as p^*. On the other hand, the *empirical distribution* q_G of G is defined by the frequencies of all possible value combinations of items in G in the dataset. Then the significance rank of G is measured by the Kullback-Leibler divergence of p^* and q_G. The algorithm for solving the *Maximal Entropy*

model has exponential time complexity.

Probability Models

The idea that a dataset can be regarded as a sampling result from an underlying distribution encourages people to find proper probability models to interpret the transaction datasets. In previous sections, we have mentioned some probability models together with research into compressed itemsets, such as the *profile-based itemset summarization* [YCHX05], *Markov Random Field* [WP06] and *Maximal Entropy model* [Tat08].

Generally, these probability models are created based on the results of frequent itemset mining. The itemsets serve as base information or constraints. The purpose of these models is to make frequent itemsets more interpretable or informative. Other types of probability models are directly generated from the dataset itself such as the *Independent model*, the *Multivariant Tree Distribution model* [CL68], the *Mixture model* [EH81], and another *Maximal Entropy model* based on transaction statistics [TM10].

The simplest and most intuitive model is the *Independent model*. It assumes that the items are independent from each other. The probabilities of the itemsets are productions of the probabilities of the corresponding items. This model is obviously too simple to describe the correlation and association between items, but it is the start point and base line of many more effective models.

The *Multivariant Tree Distribution model* [CL68], also called the *Chow-Liu Tree*, assumes that there are only pairwise dependencies between the variables, and that the dependency graph on the attributes has a tree structure. There are three steps in building the model: computing the pairwise marginals of the attributes, computing the mutual information between the attributes and applying Kruskal's algorithm [Kru56] to find the minimum spanning tree of the full graph, whose nodes are the attributes and the weights on the edges are the mutual information. With the tree, the marginal probability of an itemset can be first decomposed to a production of factors via the chains rule and then calculated with the standard belief propagation algorithm [Pea88].

The *Mixture model* [EH81] is based on the assumption that there are latent types controlling the distribution of the items. Within each type, the items are independent. In other words, the items are conditionally independent. This assumption is a natural extension of the *Independent model*. The *Mixture model*

is itself a widely used model for statistical and machine learning tasks. The idea is to use a series of simple distributions to approximate a more complex distribution. As our work is mainly focused on this model, more discussion of the model's assumptions and algorithms will be given in later sections.

The *Maximal Entropy model*, as described in the last section, tries to find a distribution that maximizes the entropy within the constraints of frequent itemsets [PMS03, Tat08] or other statistics [TM10]. The algorithm for solving the *Maximal Entropy model* is the *Iterative Scaling* algorithm. The *Iterative Scaling* algorithm is a process of finding the probability of a given itemset query. The algorithm starts from an "ignorant" initial state and updates the parameters by enforcing them satisfying the related constraints iteratively until convergence. Finally the probability of the given query can be calculated via the parameters.

The criteria for probability models are accuracy, memory cost, offline time cost (time cost for building the model from data) and online time cost (time cost for answering a query by the model). The accuracy of a query X is often measured by:

$$e(X) = \frac{|p_M(X) - f(X)|}{f(X)} \tag{2.10}$$

where $p_M(X)$ is the probability predicted by the model and $f(X)$ is the actual frequency [PMS03, YCHX05]. Here X can be an itemset or a transaction. For a collection of queries, the accuracy is:

$$\hat{E} = \frac{1}{N_X} \sum_{j=1}^{N_X} e(X_j) \tag{2.11}$$

where N_X is the number of queries. In most cases, the collection of queries which measure the accuracy of the model are chosen to be the set of frequent itemsets or the top k frequent itemsets [YCHX05, TM10]. For models that are in their construction process, the frequent itemsets are constraints and therefore the frequent itemsets' prediction from the model is guaranteed to achieve some accuracy level such as [WP06, Tat08]. In [YCHX05], this error value is called the restoration error for evaluating the quality of summarization.

Another criterion for measuring accuracy is a comparison between the set of frequent itemsets predicted by the model and the true set of frequent itemsets. In this scheme, the information of itemsets' frequencies are ignored. The difference between the two sets is measured by calculating the false negative rate (F^-) and

the false positive rate (F^+):

$$F^- = \frac{N_M}{N_M + N_C} \tag{2.12}$$

$$F^+ = \frac{N_F}{N_F + N_C} \tag{2.13}$$

where N_M is the number of itemsets that the model failed to predict, N_F is the number of itemsets that the model falsely predicted and N_C is the number of itemsets that the model predicted correctly. This criterion is used in the research of representative itemsets in [AGM04].

The time and memory costs of several important models have been analyzed and surveyed by [PMS03]. We directly cite the conclusion in Table 2.7 at the end of this section. In Table 2.7, D is the number of items, N is the number of transactions, n_Q is the length of the query itemset, n_j is the length of j-th frequent itemset, N_F is the number of frequent itemsets, $N_{1's}(i)$ is the number of "1"s in the i-th transaction, f is the memory required to store a float number with a given precision and N_c is the number of components in the *Mixture model*.

Model	Memory Cost	Online Time Cost	Offline Time Cost
Full Data	$f\sum_{i=1}^{N} N_{1's}(i)$	$O(Nn_Q\sum_{i=1}^{N} N_{1's}(i))$	$O(1)$
Independence	fD	$O(n_Q)$	$O(DN)$
Mixture	$f(1+D)N_c$	$O(n_QN_c)$	$O(DNN_c)$
Chow-Liu	fD^2	$O(Dn_Q)$	$O(D^2N)$
Max-Entropy	$f(\sum_{j=1}^{N_F} n_j + N_F)$	$O(N_Fn_Q2^{n_Q})$	$O(NN_F)$

Table 2.7: Worst-case memory cost, online time cost and offline time cost for various models.

2.3 Mixture Model and EM Algorithm

The idea of the *Mixture model* is to present a complex distribution through a series of simpler atomic distributions. In the *Mixture model*, the unknown distribution is a weighted sum of some known distributions or a linear combination of known distributions. In theory, any distribution can be well approximated by a family of well chosen base distributions in such a mixture framework. The concept of the *Mixture model* was first described formally by B. Everitt and D. J. Hand in

their book *Finite Mixture Distributions* [EH81]. A powerful tool for finding the right mixtures and their weights for an unknown distribution is the *Expectation-Maximization (EM) algorithm* proposed by A. P. Dempster, N. M. Laird and D. B. Rubin in 1977 [DLR77]. In this section, we will introduce the assumption and the structure of the *Mixture model* and how to use the EM algorithm to perform the inference of the mixture model.

2.3.1 Assumption and Structure of Mixture Model

A default assumption of most probability models is that the data points of a dataset are independently identically distributed. If we regard the transaction dataset as a sampling result from an unknown distribution, this assumption means for arbitrary transactions A and B, the situation of transaction A does not interfere with the distribution of transaction B, and A and B follow the same distribution.

The fundamental assumption of the mixture model is the conditionally independent assumption. It assumes that for each data point there is a latent variable ranging from a finite set controlling the distribution of the data point. For a D-dimension dataset, the distributions of all attributes are independent from each other when the latent variable is chosen. Under the transaction background, if we think of the latent variable as an indicator of type, then the assumption of the mixture model means that once the type of transaction is fixed, the presence or absence of probability of items are independent from each other.

Before moving to the formal mathematical description of mixture model, we demonstrate this assumption of this model by a practical scenario. Consider a grocery store or an internet shopping website such as Amazon or Ebay. The default assumption is that the customers buy goods by their freewill and have no significant influence to each other. As there is no further knowledge about the customers, we assume that their purchasing follows an unknown but identical distribution. The assumption of the conditional independence is based on a simple observation: the customers can be categorized into different types. People have different interests. Each time they go shopping, they might focus on different "topics", such as books, electronics, sports, clothes etc. Under a certain "topic", the merchandizes are chosen randomly and independently. Different from total randomness, the correlation of the items are presented by the "topics". If several items are strongly correlated, for example, PCs and computer accessories, they

tend to have high probabilities in one type and have very low probabilities in others. This assumption obviously reflects this scenario much more naturally than the total independence assumption.

We define this assumption more formally. We denote the dataset as $\mathcal{T}$. It is a D-dimension binary dataset with N transactions. The μ-th transaction is denoted as $\mathbf{X}^\mu$. Every transaction $\mathbf{X}^\mu$ is a D-dimension binary vector with its i-th value denoted as x_i^μ, $x_i^\mu \in \{0, 1\}$. Suppose there are K components; then, each transaction is generated by one of the K components following a multinomial distribution with parameter $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$, where $\sum_{k=1}^{K} \pi_k = 1$. Here we introduce a component indicator $\mathcal{Z} = \{z^\mu\}_{\mu=1}^{N}$ indicating which components the transactions are generated from: $z^\mu = k$ if the μ-th transaction $\mathbf{X}^\mu$ is generated from the kth component. According to the model assumption, once the component is selected, the probabilities of the items are independent of each other. That is, for transaction $\mathbf{X}^\mu$:

$$p(\mathbf{X}^\mu | z^\mu, \boldsymbol{\Theta}) = \prod_{i=1}^{D} p(x_i^\mu | z^\mu, \boldsymbol{\Theta}), \tag{2.14}$$

where $\boldsymbol{\Theta}$ represents all the parameters of the model. Thus, the probability of a transaction given by the mixture model is:

$$p(\mathbf{X}^\mu | \boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k \prod_{i=1}^{D} p(x_i^\mu | z^\mu, \boldsymbol{\Theta}) \tag{2.15}$$

Since the transactions are binary vectors, we assume the conditional probability of each item follows a Bernoulli distribution with parameter ϕ_{ik}:

$$p(x_i^\mu | z^\mu, \boldsymbol{\Theta}) = \phi_{iz^\mu}^{x_i^\mu} (1 - \phi_{iz^\mu})^{1 - x_i^\mu} \tag{2.16}$$

Notice in Equation (2.16) that the parameters ϕ are labelled by the component indicator z^μ representing the z^μ-th component chosen.

A graphic representation of this model is shown in Figure 2.2 where circles denote random variables, arrows denote dependence, and plates denote replication. In Figure 2.2, the distribution of each transaction $\mathbf{X}^\mu$ depends on the selection of z^μ and model parameter $\boldsymbol{\phi}$, and z^μ depends on $\boldsymbol{\pi}$. This process will repeat N times to generate the whole dataset.

The Bernoulli mixture is a hierarchical model. If we want to sample from this

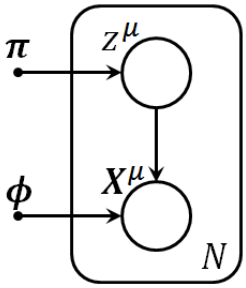


Figure 2.2: Graphic Representation of Mixture Model for Transaction Data Set

model, we need first to choose the components according to the multinomial distribution. Subsequently, based on the component or the sub-distribution selected, we can sample the transaction. We can describe the process as following:

1. Choose a K-dimension vector $\boldsymbol{\pi}$ satisfying $\sum_{k=1}^{K} \pi_k = 1$ as the proportion vector of the components.

2. For each item $\{i\}$ and component C_k, choose $\phi_{ik} \in [0, 1]$ as the conditionally probability of each item given the component C_k where $i \in \{1, \ldots, D\}$ and $k \in \{1, \ldots, K\}$.

3. For each transaction $\mathbf{X}^\mu$

   (a) Choose a component indicator $z^\mu \sim$Multinomial($\boldsymbol{\pi}$), where

   $$p(z^\mu = k|\boldsymbol{\pi}) = \pi_k \tag{2.17}$$

   (b) Then we can generate data by:

   $$p(\mathbf{X}^\mu|z^\mu, \boldsymbol{\phi}) = \prod_{i=1}^{D} p(x_i^\mu|z^\mu, \phi_{iz^\mu}) = \prod_{i=1}^{D} \phi_{iz^\mu}^{x_i^\mu}(1 - \phi_{iz^\mu})^{1-x_i^\mu} \tag{2.18}$$

In this model, we need to estimate π_k and ϕ_{ik} from the data. That is the proportion of each component and the conditional probabilities of the items. The estimation of the parameters for a probability model should follow the maximum likelihood principle. We need to find proper π_k and ϕ_{ik} to maximize the likelihood of the data. In optimization, the likelihood is often replaced by the log-likelihood for convenience. The problem of mixture model parameter estimation is described

as following:

$$\begin{aligned} &\text{Find proper } \boldsymbol{\Theta}^*, \\ &\text{s.t. } \boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \ln \mathcal{L}(\boldsymbol{\Theta}|\mathcal{T}) \end{aligned} \tag{2.19}$$

where $\ln \mathcal{L}(\boldsymbol{\Theta}|\mathcal{T})$ is the log-likelihood of the data given the model. The log-likelihood can be further written as:

$$\ln \mathcal{L}(\boldsymbol{\Theta}|\mathcal{T}) = \ln p(\mathcal{T}|\boldsymbol{\Theta}) = \sum_{\mu=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \prod_{i=1}^{D} \phi_{ik}^{x_i^\mu} (1-\phi_{ik})^{1-x_i^\mu} \right] \tag{2.20}$$

It is a problem of parameter estimation with hidden variable $\boldsymbol{\pi}$. Traditional multivariable optimization methods cannot handle such a problem because the log-likelihood contains the log of the sum which makes the terms hard to simplify and solve analytically. The most widely used algorithm for this kind of parameter estimation problem is the EM algorithm which we introduce next.

2.3.2 Expectation-Maximization Algorithm

The *Expectation-Maximization algorithm* was first introduced by A. P. Dempster, N. M. Laird and D. B. Rubin in their famous paper *Maximum Likelihood from Incomplete Data via the EM Algorithm* in 1977 [DLR77]. In this paper, the EM algorithm was proposed as a general method of finding the maximum-likelihood estimation of the parameters from a dataset when the dataset is incomplete or has missing data. The EM algorithm can be used when a dataset really has missing data, or when the optimization of the likelihood function is analytically intractable such as with our mixture model. When applying the EM algorithm to the mixture model, the mixing proportion $\boldsymbol{\pi}$ is regarded as the "missing value" of the dataset. In this section we use the EM algorithm to estimate the parameters in our mixture model.

Basic EM Algorithm

Before applying the EM algorithm to our Bernoulli mixture model, we will briefly introduce the basic process of the EM algorithm from a "missing data" viewpoint. We assume the data $\mathcal{T}$ is observed and is generated from some unknown distribution. We also assume that the observed data $\mathcal{T}$ is incomplete. The complete

dataset exists $\mathcal{D} = (\mathcal{T}, \mathcal{Z})$ and the joint probability function given the parameters $\mathbf{\Theta}$:

$$p(\mathbf{d}|\mathbf{\Theta}) = p(\mathbf{X}, \mathbf{z}|\mathbf{\Theta}) = p(\mathbf{z}|\mathbf{X}, \mathbf{\Theta})p(\mathbf{X}|\mathbf{\Theta}) \tag{2.21}$$

By Equation (2.21), we can define a new likelihood function:

$$\mathcal{L}(\mathcal{D}|\mathbf{\Theta}) = \mathcal{L}(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta}) = p(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta}) \tag{2.22}$$

This is called the complete-data likelihood. In this function $\mathcal{Z}$ is an unknown random variable governed by some underlying distribution. This means the complete-data likelihood $\mathcal{L}(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta})$ can be seen as a function of $\mathcal{Z}$. The idea of the EM algorithm is to find the expectation of this function with respect to the missing data $\mathcal{Z}$ and to maximize it. Differing from traditional optimization methods, the EM algorithm maximizes the expected value of the likelihood function instead of the value itself as there is an unknown random variable involved.

If we temporarily think of the current estimation of parameters $\mathbf{\Theta}^{(i-1)}$ as constants, the expected value of the complete-data log-likelihood function $\mathcal{L}(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta})$ with respect to $\mathcal{Z}$ given the observed data $\mathcal{T}$ and current estimated model parameter $\mathbf{\Theta}^{(i-1)}$ is:

$$Q(\mathbf{\Theta}, \mathbf{\Theta}^{(t-1)}) = E\left[\log \mathcal{L}(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta})|\mathcal{T}, \mathbf{\Theta}^{(t-1)}\right] \tag{2.23}$$

where $\mathbf{\Theta}^{(t-1)}$ is the current estimation of parameters we used to evaluate the expectation and $\mathbf{\Theta}$ are the new parameters that we optimize to increase Q. In Equation(2.23), the expected value is the log-likelihood function as a function of random variable $\mathcal{Z}$ given $\mathcal{T}$ and $\mathbf{\Theta}^{(t-1)}$. The probability density function of $\mathcal{Z}$ is therefore the posterior probability density $p(\mathcal{Z}|\mathcal{T}, \mathbf{\Theta}^{(t-1)})$. Then Equation (2.23) can be further expanded as:

$$E\left[\log \mathcal{L}(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta})|\mathcal{T}, \mathbf{\Theta}^{(t-1)}\right] = \int_{\mathcal{Z}\in\mathbf{\Omega}} \log p(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta})p(\mathcal{Z}|\mathcal{T}, \mathbf{\Theta}^{(t-1)})d\mathcal{Z} \tag{2.24}$$

where $\mathbf{\Omega}$ is the space that $\mathcal{Z}$ can take values from. This step is called the *Expectation Step* (*E-Step*). Notice in the expectation, $\mathcal{T}$ and $\mathbf{\Theta}^{(t-1)}$ are constants and $\mathcal{Z}$ is integrated out. Thus the expectation is deterministic and can be maximized via normal differential methods. The next step is called the *Maximization Step* (*M-Step*). It functions to maximize the "Q-function" with respect of $\mathbf{\Theta}$. That is,

we find:

$$\boldsymbol{\Theta}^{(t)} = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(t-1)}) \tag{2.25}$$

The E-step and M-step are repeated as necessary. Each iteration is guaranteed to increase the log-likelihood and the algorithm is guaranteed to converge to a local maximum of the likelihood function. The above is an abstract framework of the EM algorithm. We introduce its application on the Bernoulli mixture model in the next section.

EM Algorithm for Bernoulli Mixture Model

In this part, we use the EM algorithm to optimize the Bernoulli mixture model. Recall the likelihood function of the transaction dataset in Equation (2.20):

$$\ln \mathcal{L}(\boldsymbol{\Theta}|\mathcal{T}) = \ln p(\mathcal{T}|\boldsymbol{\Theta}) = \sum_{\mu=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \prod_{i=1}^{D} \phi_{ik}^{x_i^\mu} (1-\phi_{ik})^{1-x_i^\mu} \right]$$

This function is hard to optimize analytically because it contains the log of the sum. According to the EM algorithm, we can assume that the transaction dataset is incomplete and that there is a latent variable $\mathcal{Z}$ indicating which component the transaction is generated from. Merging the observed data $\mathcal{T}$ and the unobserved data $\mathcal{Z}$, the complete-data log-likelihood function is:

$$\begin{aligned} \ln \mathcal{L}(\boldsymbol{\Theta}|\mathcal{T}, \mathcal{Z}) = \ln p(\mathcal{T}, \mathcal{Z}|\boldsymbol{\Theta}) &= \sum_{\mu=1}^{N} \ln \left[p(\mathbf{X}|z, \boldsymbol{\Theta}) p(z|\boldsymbol{\Theta}) \right] \\ &= \sum_{\mu=1}^{N} \ln [\pi_z p(\mathbf{X}|\boldsymbol{\phi}_z)] \end{aligned} \tag{2.26}$$

where we omit the μ label of z^μ and $\mathbf{X}^\mu$ for the sake of simplicity. In subsequent equations, we will also omit this label when no ambiguity is caused. This likelihood is much simpler than Equation (2.20) and can be optimized via normal differential methods. Obviously, the problem is that we do not know the values of $\mathcal{Z}$. According to the EM algorithm, we optimize the expectation of this likelihood function with respect to $\mathcal{Z}$ instead. First, we need to induce the posterior

distribution of $\mathcal{Z}$. Using the Bayes' rule, the posterior distribution is:

$$\tau_k^\mu = p(z=k|\mathbf{X},\boldsymbol{\Theta}) = \frac{p(\mathbf{X}|z=k,\boldsymbol{\Theta})p(z=k|\boldsymbol{\Theta})}{p(\mathbf{X}|\boldsymbol{\Theta})} = \frac{\pi_z p(\mathbf{X}|\boldsymbol{\phi}_k)}{\sum_{k=1}^K \pi_{k'} p(\mathbf{X}|\boldsymbol{\phi}_{k'})} \tag{2.27}$$

and

$$p(\mathcal{Z}|\mathcal{T},\boldsymbol{\Theta}) = \prod_{\mu=1}^N p(z^\mu|\mathbf{X}^\mu,\boldsymbol{\Theta}) \tag{2.28}$$

Here we use τ_k^μ to denote the posterior probability of component k given the μ-th transaction and the current parameters. With Equation (2.27) and (2.28), the expectation of the log-likelihood function is:

$$\begin{aligned}
Q(\boldsymbol{\Theta},\boldsymbol{\Theta}^{(t-1)}) &= \sum_{\mathcal{Z}} \ln \mathcal{L}(\boldsymbol{\Theta}|\mathcal{T},\mathcal{Z}) p(\mathcal{Z}|\mathbf{X},\boldsymbol{\Theta}^{(t-1)}) \\
&= \sum_{\mathcal{Z}} \sum_{\mu=1}^N \ln(\pi_{z^\mu} p(\mathbf{X}^\mu|\boldsymbol{\phi}_{z^\mu})) \prod_{\mu'=1}^N p(z^{\mu'}|\mathbf{X}^{\mu'},\boldsymbol{\Theta}) \\
&= \sum_{z^1=1}^K \sum_{z^2=1}^K \cdots \sum_{z^N=1}^K \sum_{\mu=1}^N \ln(\pi_{z^\mu} p(\mathbf{X}^\mu|\boldsymbol{\phi}_{z^\mu})) \prod_{\mu'=1}^N p(z^{\mu'}|\mathbf{X}^{\mu'},\boldsymbol{\Theta}) \\
&= \sum_{\mu=1}^N \sum_{z^\mu=1}^K \ln(\pi_{z^\mu} p(\mathbf{X}^\mu|\boldsymbol{\phi}_{z^\mu})) p(z^\mu|\mathbf{X}^\mu,\boldsymbol{\Theta}) \sum_{\mathcal{Z}/\{z^\mu\}} \prod_{\mu'\neq\mu}^N \prod_{\mu=1}^N p(z^{\mu'}|\mathbf{X}^{\mu'},\boldsymbol{\Theta}) \\
&= \sum_{\mu=1}^N \sum_{k=1}^K \tau_k^\mu \ln(\pi_k p(\mathbf{X}^\mu|\boldsymbol{\phi}_k))
\end{aligned} \tag{2.29}$$

In Equation (2.29), we use the fact that $\sum_{\mathcal{Z}/z^\mu} \prod_{\mu'\neq\mu}^N p(z^{\mu'}|\mathbf{X}^{\mu'},\boldsymbol{\Theta}) = 1$. The sum of the joint probabilities of all $z^{\mu'}$s except the chosen z^μ equals 1. In the last step, we change the label of parameters in the equation from z to k as there is no difference when summing them up. Now we can maximize the "Q-function" with respect to π_k and $\boldsymbol{\phi}_k$.

Firstly, we find the expression for π_k with the constraint of $\sum_{k=1}^K \pi_k = 1$. We introduce the Lagrange multiplier λ and solve the following equation:

$$\frac{\partial}{\partial \pi_k}\left[\sum_{\mu=1}^N \sum_{k=1}^K \tau_k^\mu \ln(\pi_k p(\mathbf{X}^\mu|\boldsymbol{\phi}_k)) + \lambda(\sum_{k=1}^K \pi_k - 1)\right] = 0 \tag{2.30}$$

or

$$\sum_{\mu=1}^{N} \frac{1}{\pi_k} \tau_k^\mu + \lambda = 0 \tag{2.31}$$

It is easy to pick $\lambda = -N$ and find:

$$\pi_k = \frac{1}{N} \sum_{\mu=1}^{N} \tau_k^\mu \tag{2.32}$$

The $\boldsymbol{\phi}_k$ is a series of parameters $\{\phi_{1k}, \phi_{2k}, \ldots, \phi_{Dk}\}$. According to Equation (2.18):

$$p(\mathbf{X}^\mu | \boldsymbol{\phi}_k) = \prod_{i=1}^{D} \phi_{ik}^{x_i^\mu} (1 - \phi_{ik})^{1 - x_i^\mu}$$

Then, for ϕ_{ik} we need to solve:

$$\frac{\partial}{\partial \phi_{ik}} \left[\sum_{\mu=1}^{N} \sum_{k=1}^{K} \tau_k^\mu \ln(\pi_k \prod_{i=1}^{D} \phi_{ik}^{x_i^\mu} (1 - \phi_{ik})^{1 - x_i^\mu}) \right] = 0 \tag{2.33}$$

or

$$\sum_{\mu=1}^{N} \tau_k^\mu \left[\frac{x_i^\mu}{\phi_{ik}} - \frac{1 - x_i^\mu}{1 - \phi_{ik}} \right] = 0 \tag{2.34}$$

It turns out:

$$\phi_{ik} = \frac{\sum_{\mu=1}^{N} x_i^\mu \tau_k^\mu}{\sum_{\mu'}^{N} \tau_k^{\mu'}} \tag{2.35}$$

In summary, the Equations (2.27), (2.32) and (2.35) form the framework of the EM algorithm. For the E-step, we calculate the posterior distribution of the latent variables:

$$\tau_k^\mu = \frac{\pi_k^{(t-1)} p(\mathbf{X}^\mu | \boldsymbol{\phi}_k^{(t-1)})}{\sum_{k'}^{K} \pi_{k'}^{(t-1)} p(\mathbf{X}^\mu | \boldsymbol{\phi}_{k'}^{(t-1)})} \tag{2.36}$$

With this posterior distribution, in the M-step we can maximize the likelihood function and optimize the parameters:

$$\pi_k = \frac{1}{N} \sum_{\mu=1}^{N} \tau_k^\mu \tag{2.37}$$

$$\phi_{ik} = \frac{\sum_{\mu=1}^{N} x_i^\mu \tau_k^\mu}{\sum_{\mu'}^{N} \tau_k^{\mu'}} \tag{2.38}$$

Algorithm 2.3 EM algorithm for Bernoulli Mixtures

initialize π_k and ϕ_{ik}
repeat
  for $\mu = 1$ to N **do**
    for $k = 1$ to K **do**
      $\tau_k^\mu = \frac{\pi_k \prod_{i=1}^{D} \phi_{ik}^{x_i^\mu} (1-\phi_{ik})^{1-x_i^\mu}}{\sum_{k'=1}^{K} \pi_{k'} \prod_{i'=1}^{D} \phi_{i'k'}^{x_{i'}^\mu} (1-\phi_{i'k'})^{1-x_{i'}^\mu}}$
    end for
  end for
  $\pi_k = \frac{1}{N} \sum_{\mu=1}^{N} \tau_k^\mu$
  $\phi_{ik} = \frac{\sum_{\mu=1}^{N} x_i^\mu \tau_k^\mu}{\sum_{\mu'}^{N} \tau_k^{\mu'}}$
until convergence

Algorithm 4.2 shows the process of the EM algorithm for the Bernoulli mixture model.

An Alternative Explanation of the EM Algorithm

Here, we will introduce another explanation of the EM algorithm from [Bis07] as this explanation can be a base for the deduction of the variational EM algorithm for Bayesian mixture model. Return to the likelihood of the given dataset $\mathcal{T}$. As we proposed, the $\mathcal{Z}$ are random variables with unknown distribution. Assume $\mathcal{Z}$ follows a distribution $q(\mathcal{Z})$. Then, for any $q(\mathcal{Z})$ the following equation holds:

$$\ln p(\mathcal{T}|\mathbf{\Theta}) = \sum_{\mathcal{Z}} q(\mathcal{Z}) \ln p(\mathcal{T}|\mathbf{\Theta}) = L(q, \mathbf{\Theta}) + KL(q||p), \tag{2.39}$$

where $\sum_{\mathcal{Z}} \sim \sum_{z^1=1}^{K} \cdots \sum_{z^\mu=1}^{K} \cdots \sum_{z^N=1}^{K}$ and

$$L(q, \mathbf{\Theta}) = \sum_{\mathcal{Z}} q(\mathcal{Z}) \ln \frac{p(\mathcal{T}, \mathcal{Z}|\mathbf{\Theta})}{q(\mathcal{Z})} \tag{2.40}$$

$$KL(q||p) = -\sum_{\mathcal{Z}} q(\mathcal{Z}) \ln \frac{p(\mathcal{Z}|\mathcal{T}, \mathbf{\Theta})}{q(\mathcal{Z})} \tag{2.41}$$

In Equation (2.39), $KL(q||p)$ is the Kullback-Leibler divergence between $q(\mathcal{Z})$ and the true posterior distribution $p(\mathcal{Z}|\mathcal{T}, \mathbf{\Theta})$. Recall that for any distribution, the Kullback-Leibler divergence $KL(q||p) \geq 0$, with equality if and only if $p(\mathcal{Z}) = p(\mathcal{Z}|\mathcal{T}, \mathbf{\Theta})$. Therefore based on Equation (2.39), we have $\ln p(\mathcal{T}|\mathbf{\Theta}) \geq L(q, \mathbf{\Theta})$. Thus, $L(q, \mathbf{\Theta})$ can be regarded as a lower bound of the log-likelihood. We can

maximize the likelihood by maximizing $L(q, \mathbf{\Theta})$. For $q(\mathcal{Z})$ we assume it follows a multinomial distribution form:

$$q(\mathcal{Z}) = \prod_{\mu=1}^{N} q(z^\mu), \text{where } q(z^\mu) \sim \text{Multinomial}(\boldsymbol{\tau}^\mu),$$
$$\boldsymbol{\tau}^\mu = (\tau_1^\mu, \ldots, \tau_K^\mu), \sum_{k=1}^{K} \tau_k^\mu = 1 \tag{2.42}$$

Thus, we could expand $L(q, \mathbf{\Theta})$:

$$L(q, \mathbf{\Theta}) = \sum_{\mathcal{Z}} q(\mathcal{Z}) \ln p(\mathcal{T}|\mathcal{Z}, \mathbf{\Theta}) + \sum_{\mathcal{Z}} q(\mathcal{Z}) \ln p(\mathcal{Z}|\mathbf{\Theta}) - \sum_{\mathcal{Z}} q(\mathcal{Z}) \ln q(\mathcal{Z}) \tag{2.43}$$

All the terms involve standard computations in the exponential family, and the following optimization of the parameters could be solved by a classic multivariate function maximization with constraints. We treat these terms one by one:

$$\begin{aligned}
&\sum_{\mathcal{Z}} q(\mathcal{Z}) \ln p(\mathcal{Z}|\mathcal{T}, \mathbf{\Theta}) \\
&= \sum_{\mathcal{Z}} \left[\prod_{\mu'=1}^{N} \tau_{z^{\mu'}}^{\mu'} \right] \sum_{\mu=1}^{N} \sum_{i=1}^{D} \ln[\phi_{iz^\mu}^{x_i^\mu} (1 - \phi_{iz^\mu})^{1-x_i^\mu}] \\
&= \sum_{\mathcal{Z} \setminus \{z^\mu\}} \left[\prod_{\mu' \neq \mu}^{N} \tau_{z^{\mu'}}^{\mu'} \right] \sum_{\mu=1}^{N} \sum_{i=1}^{D} \sum_{z^\mu=1}^{K} \tau_{z^\mu}^\mu \ln[\phi_{iz^\mu}^{x_i^\mu} (1 - \phi_{iz^\mu})^{1-x_i^\mu}] \\
&= \sum_{\mu=1}^{N} \sum_{i=1}^{D} \sum_{k=1}^{K} \tau_k^\mu \left[x_i^\mu \ln \frac{\phi_{ik}}{1 - \phi_{ik}} + \ln(1 - \phi_{ik}) \right]
\end{aligned} \tag{2.44}$$

Similarly, we have:

$$\sum_{\mathcal{Z}} q(\mathcal{Z}) \ln p(\mathcal{Z}|\mathbf{\Theta}) = \sum_{\mu=1}^{N} \sum_{k=1}^{K} \tau_k^\mu \ln \pi_k \tag{2.45}$$

$$-\sum_{\mathcal{Z}} q(\mathcal{Z}) \ln q(\mathcal{Z}) = -\sum_{\mu=1}^{N} \sum_{k=1}^{K} \tau_k^\mu \ln \tau_k^\mu \tag{2.46}$$

We can use differential method to maximize Equation (2.43) with respect to τ_k^μ, π_k and ϕ_{ik}. The result is the same as Equation (2.27), (2.32) and (2.35). Equations

(2.32) and (2.35) depend on τ_k^μ and Equation (2.27) depends on π_k and ϕ_{ik}. Hence the optimizing process alternates between two phases. After the model initialization, first we compute τ_k^μ according to Equation (2.27). This step is the E-step. In this step $q(\mathcal{Z})$ is set to equal $p(\mathcal{Z}|\mathcal{T}, \mathbf{\Theta}^{old})$, causing the lower bound $L(q, \mathbf{\Theta}^{old})$ to increase to the same value as the log-likelihood function $\ln p(\mathcal{T}|\mathbf{\Theta}^{old})$ by vanishing the Kullback-Leibler divergence $KL(q||p)$. Then we compute π_k and ϕ_{ik} according to Equation (2.32) and (2.35). This step is the M-step. In this step, $q(\mathcal{Z})$ is fixed and the lower bound $L(q, \mathbf{\Theta}^{old})$ is maximized by altering $\mathbf{\Theta}^{old}$ to $\mathbf{\Theta}^{new}$. As the KL divergence is always non-negative, the log-likelihood function $\ln p(\mathcal{T}|\mathbf{\Theta})$ increases at least as much as the lower bound does. The E-Step and M-Step iterate until the log-likelihood meets a local maximum.

Prediction via the Bernoulli Mixture Model

Once the parameters of the model are trained, the probability of the itemsets can be easily predicted. For any itemset I, its probability is calculated by only taking the items occurring in I into account and ignoring the items which are not in I:

$$p(I|\mathbf{\Theta}) = \sum_{k=1}^{K} \pi_k \prod_{i_m \in I} \phi_{mk} \tag{2.47}$$

The number of free parameters used for prediction is $K(D+1)-1$. The searching algorithm of the whole set of frequent itemsets is the same as traditional frequent itemset mining algorithms, the only difference being that the frequencies of the itemsets are calculated instead of scanning the original dataset and counting in a data mining algorithm. With a prepared model, both time and memory cost can be greatly reduced with some accuracy loss since the frequency counting process has been replaced by a simple calculation of summation and multiplication.

Review of the Bernoulli Mixture Model

The Bernoulli mixture model is an easy and efficient probability model for use with transaction datasets. It tries to cluster the transactions into different probability "prototypes" or "profiles". The form of the component is quite similar to the "profiles" in the profile-based summarization [YCHX05], although the structure and inferences of the two models are different. If we think of these components as the basis vectors of a space, then the posterior distribution of each transaction

$\boldsymbol{\tau}^{\mu}$ can be mapped to a point on the $(K-1)$-dimension simplex. The proportion vector of the components $\boldsymbol{\pi}$ is the central point of all the points mapped from the posterior distributions of the transactions. From this view, the distribution of the posterior distributions is represented simply by the mean value of these distributions. When replacing a distribution by a single point, some information may be lost, which can result in some under-fitting.

From a Bayesian view, there is no prior for the parameters involved. The inferences of the parameters are all based on the given data. In practice, this might cause an over-fitting problem, especially when the number of components assigned is much greater than necessary. The smoothing of the model is also a problem. In practice, if the parameters are not smoothed, it is very likely that in the testing data some transactions will appear, while the probability predicted by the model is exactly 0. This is another kind of over-fitting.

The last problem we want to discuss is the assignment of "K", the number of components. In clustering algorithms, finding proper K is always a difficult task. People have to choose K by guessing, experience or previous experiment results. There is no formal method for finding "K". Clustering and probability models such as mixture models are unsupervised learning methods, so it is hard to prove that there exists a value of K that is the best value.

In recent years, with rapid development in Bayesian and nonparametric modelling, the problems of the mixture model mentioned are diminishing. The over-fitting and smoothing problems are mitigated by introducing Bayesian priors to the parameters. The problem of finding "K", in other words the potential under-fitting problem, is mitigated by using nonparametric methods: the Dirichlet process mixture model.

We will give a background introduction to both the Dirichlet distribution and Dirichlet process in the next section. The application of the Bayesian mixture model and the Dirichlet process mixture model in our work will be introduced in Chapters 3 and 4.

2.4 Dirichlet Distribution and Dirichlet Process

Bayesian inference views the model parameters as random variables. The parameters' distribution $p(\boldsymbol{\Theta})$ is the *prior probability distribution*. If the posterior distribution $p(\boldsymbol{\Theta}|Data)$ is in the same family as the prior probability distribution,

the prior and the posterior are then called *conjugate distributions*, and the prior is called a *conjugate prior* for the likelihood. For example, the *Gaussian* family conjugates with itself with respect to a Gaussian function likelihood. That means if the likelihood function is a Gaussian function, and if we choose a Gaussian as the prior, the posterior distribution is also a Gaussian. Other examples are that the conjugate prior for the Bernoulli distribution is the *Beta distribution* and the conjugate prior for multinomial distribution is the *Dirichlet distribution*.

For a given likelihood function, choosing its conjugate prior makes mathematical manipulation easier. In the mixture model, we need to handle two kinds of distribution: a multinomial distribution for component selection and a series of Bernoulli distributions for sampling each attribute. When we transform this model to a Bayesian model, we will choose the Dirichlet distribution as the prior to the multinomial distribution and the Beta distributions as the priors to the Bernoulli distributions.

In this section, we introduce the Dirichlet distribution. Then, as we plan to extend the number of components of the mixture model to infinity, or as many as necessary, we introduce the corresponding prior distribution: the *Dirichlet process*. The mathematical concepts in this section is the theoretical foundation of our work in Chapter 3 and Chapter 4.

2.4.1 Dirichlet Distribution

Here, we introduce the definition and some important properties of the Dirichlet distribution. Then we discuss the related posterior distributions and explain the Polya Urn model for the Dirichlet distribution.

Definition and Basic Properties

The Dirichlet distribution is a multi-parameter generalization of the Beta distribution. Its definition [Fer73] is as follows:

Definition 2.3. *Let $Z_1, \ldots, Z_K$ be independent random variables with $Z_k \sim G(\alpha_k, 1)$ where $G(\alpha, \beta)$ is the* ***Gamma distribution****:*

$$G(z|\alpha, \beta) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} e^{-z/\beta} z^{\alpha-1} \tag{2.48}$$

With the shape parameter $\alpha \geq 0$, the scale parameter $\beta > 0$ and z is always

positive. The ***Dirichlet distribution*** *with parameter* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$*, with all* $\alpha_k \geq 0$ *and some of* $\alpha_k > 0$*, denoted by* $Dir(\boldsymbol{\alpha})$*, is defined as the distribution of* $(Y_1, \ldots, Y_K)$*, where:*

$$Y_k = \frac{Z_k}{\sum_{k'=1}^{K} Z_{k'}} = \frac{Z_k}{Z} \tag{2.49}$$

By this definition, the joint probability of the Dirichlet distribution is:

$$\begin{aligned} p(Y_1, \ldots, Y_K | \alpha_1, \ldots, \alpha_K) &= \int_0^\infty |J(\mathbf{Z}, \mathbf{Y})| p_Z(ZY_1, \ldots, ZY_K | \alpha_1, \ldots, \alpha_K) dZ \\ &= \int_0^\infty Z^{K-1} \prod_{k=1}^{K} \frac{1}{\Gamma(\alpha_k)} e^{-ZY_k} (ZY_k)^{\alpha_k - 1} dZ \\ &= \int_0^\infty Z^{K-1} \frac{1}{\prod_{k=1}^{K} \Gamma(\alpha_k)} e^{-Z} Z^{\alpha - K} \prod_{k=1}^{K} Y_k^{\alpha_k - 1} dZ \\ &= \frac{1}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} Y_k^{\alpha_k - 1} \int_0^\infty e^{-Z} Z^{\alpha - 1} dZ \\ &= \frac{\Gamma(\alpha)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} Y_k^{\alpha_k - 1} \end{aligned} \tag{2.50}$$

where $Z = \sum_{k=1}^{K} Z_k$, $\alpha = \sum_{k=1}^{K} \alpha_k$, $\sum_{k=1}^{K} Y_k = 1$ and we use the definition of the Gamma function:

$$\Gamma(\alpha) = \int_0^\infty e^{-t} t^{\alpha - 1} dt$$

Note that Y_K is in fact an abbreviation of the term $1 - \sum_{k=1}^{K-1} Y_k$ since $Y_1 + \cdots + Y_K = 1$.

An important property of the Dirichlet distribution is the additive property.

Property 2.1. *If* $(Y_1, \ldots, Y_K) \sim Dir(\alpha_1, \ldots, \alpha_K)$ *and* $r_1, \ldots, r_l$ *are integers such that* $0 < r_1 < \cdots < r_l = K$*, then:*

$$(\sum_{1}^{r_1} Y_k, \sum_{r_1+1}^{r_2} Y_k, \ldots, \sum_{r_{l-1}+1}^{r_l} Y_k) \sim Dir(\sum_{1}^{r_1} \alpha_k, \sum_{r_1+1}^{r_2} \alpha_k, \ldots, \sum_{r_{l-1}+1}^{r_l} \alpha_k) \tag{2.51}$$

This follows directly from the additive property of the Gamma distribution: if $Z_1 \sim G(\alpha_1, 1)$ and $Z_2 \sim G(\alpha_2, 1)$, then $Z_1 + Z_2 \sim G(\alpha_1 + \alpha_2, 1)$. Let $Z = Z_1 + Z_2$, then:

$$p(Z | \alpha_1, \alpha_2) = \int_0^Z p(Z_1 | \alpha 1) p(Z - Z_1 | \alpha_2) dZ_1$$

$$
\begin{aligned}
&= \int_0^Z \frac{1}{\Gamma(\alpha_1)} e^{-Z_1} Z_1^{\alpha_1-1} \cdot \frac{1}{\Gamma(\alpha_2)} e^{-Z+Z_1} (Z - Z_1)^{\alpha_2-1} dZ_1 \\
&= \frac{1}{\Gamma(\alpha_1)\Gamma(\alpha_2)} e^{-Z} \int_0^Z Z_1^{\alpha_1-1} (Z - Z_1)^{\alpha_2-1} dZ_1 \\
&= \frac{1}{\Gamma(\alpha_1)\Gamma(\alpha_2)} e^{-Z} \cdot \frac{\Gamma(\alpha_1)\Gamma(\alpha_2)}{\Gamma(\alpha_1+\alpha_2)} Z^{\alpha_1+\alpha_2-1} \\
&= \frac{1}{\Gamma(\alpha_1+\alpha_2)} e^{-Z} Z^{\alpha_1+\alpha_2-1} \\
&\sim G(\alpha_1+\alpha_2, 1)
\end{aligned} \tag{2.52}
$$

In Dirichlet distribution, the label of the $\{Y_k\}$ is exchangeable, which means this additive property is valid for arbitrary partition of the $\{Y_k\}$.

$(Y_1, \ldots, Y_K)$ are often used to describe a multinomial distribution. Let $(Y_1, \ldots, Y_K)$ be the probability distribution for the event space $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_K\}$, s.t. $p(X = \mathcal{X}_k) = Y_k$ where X is a random variable in space $\mathcal{X}$. Property 2.1 means if $(Y_1, \ldots, Y_K)$ follow Dirichlet distribution, then for any random partition $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}, M \leq K$ over the space $\mathcal{X}$ s.t. $\mathcal{B}_1 \cap \mathcal{B}_2 = \emptyset \ \forall \mathcal{B}_1, \mathcal{B}_2 \in \mathcal{B}$ and $\cup_{j=1}^{M} \mathcal{B}_j = \mathcal{X}$, the probability of this partition still follows a Dirichlet distribution:

$$
p(\mathcal{B}_1), \ldots, p(\mathcal{B}_M) \sim Dir(\mathcal{B}(\boldsymbol{\alpha})) \tag{2.53}
$$

where we use $\mathcal{B}(\boldsymbol{\alpha})$ denoting that the parameters are reorganized according to the partition. Alternatively, we can write Equation (2.53) as follows, for the discussion of the Dirichlet process in later sections.

$$
\mathbf{Y}(\mathcal{B}_1), \ldots, \mathbf{Y}(\mathcal{B}_M) \sim Dir(\boldsymbol{\alpha}(\mathcal{B}_1), \ldots, \boldsymbol{\alpha}(\mathcal{B}_M)) \tag{2.54}
$$

Here $\mathbf{Y}(\mathcal{B}_i)$ is a measure of the partition over the event space.

With this additive property, we can easily get the marginal distribution of a given Y_k. It is simply partitioning Y_k and the rest into two parts. Hence, this is a Beta distribution:

$$
\begin{aligned}
p(Y_k|\boldsymbol{\alpha}) &= \frac{\Gamma(\alpha)}{\Gamma(\alpha_k)\Gamma(\alpha - \alpha_k)} Y_k^{\alpha_k-1} (1 - Y_k)^{\alpha-\alpha_k-1} \\
&\sim Beta(\alpha_k, \alpha - \alpha_k)
\end{aligned} \tag{2.55}
$$

where $\alpha = \sum_{k=1}^{K} \alpha_k$. The expectation and the second raw moment of a given Y_k

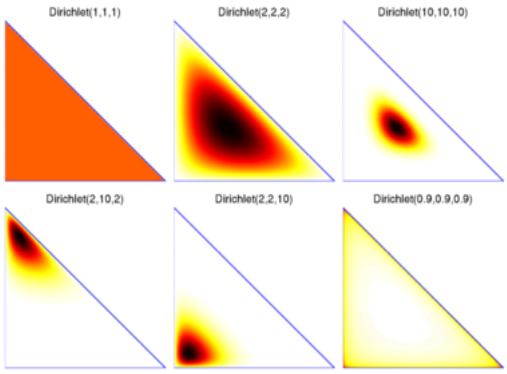


Figure 2.3: Several images of the probability density of the Dirichlet distribution

is:

$$E[Y_k] = \frac{\alpha_k}{\alpha} \tag{2.56}$$

$$E[Y_k^2] = \frac{\alpha_k(\alpha_k + 1)}{\alpha(\alpha + 1)} \tag{2.57}$$

For easier explanation of the Dirichlet process in the following section, we will discuss the relationship between Y_k and α_k a little further. As Y_k is normalized, we can also normalize α_k by letting $g_k = \alpha_k/\alpha$. Here $\alpha = \sum_{k=1}^{K} \alpha_k$ is often called the concentration parameter. Figure 2.3[1] and shows the differences of the distributions between different concentration parameters. Recall that the Y_ks are often viewed as a multinomial distribution over the event space $\mathcal{X}$. Since g_ks are also normalized and they are in fact the expectations of the Y_ks, we can also think that the g_ks form a base distribution or measure over $\mathcal{X}$. The random measure $\mathbf{Y}$ is sampled from the Dirichlet distribution with the base measure $\mathbf{G}_0 = (g_1, \ldots, g_k)$ and the concentration α as parameters.

Posterior Distributions and the Polya Urn Model

Consider that $\mathbf{Y} = (Y_1, \ldots, Y_K)$ describes a mulitnomial distribution with $p(X = \mathcal{X}_k) = Y_k$. We want to know the posterior distribution of $(Y_1, \ldots, Y_K)$ with an observation of X or several observations $X_1, \ldots, X_N$. Using the Bayes rule,

$$p(\mathbf{Y}|X = \mathcal{X}_k) = \frac{p(X = \mathcal{X}_k|\mathbf{Y})p(\mathbf{Y})}{p(X = \mathcal{X}_k)}$$

[1]This image is from `http://projects.csail.mit.edu/church/w/images/thumb/7/73/Dirichlet.png/400px-Dirichlet.png`

$$
\begin{aligned}
&= \frac{1}{C_1} Y_k \cdot C_2 \prod_{k'=1}^{K} Y_{k'}^{\alpha_{k'}-1} \\
&\sim Dir(\alpha_1, \ldots, \alpha_k + 1, \ldots, \alpha_K)
\end{aligned} \tag{2.58}
$$

For multiple observations $X_1, \ldots, X_N$, assuming that the number of event $\mathcal{X}_k$ is n_k, $\sum_{k=1}^{K} n_k = N$. Then following Equation (2.58), we have:

$$
p(\mathbf{Y}|X_{1:N}) \sim Dir(\alpha_1 + n_1, \ldots, \alpha_K + n_K) \tag{2.59}
$$

Notice that the posterior distribution given some observations is still a Dirichlet distribution.

Reversely, we are also concerned with the distribution of the observations. That is, given the prior parameters $\boldsymbol{\alpha}$, what is the probability of the random variable X? We can integrate over $\mathbf{Y}$:

$$
\begin{aligned}
p(X = \mathcal{X}_k) &= \int_{\mathbf{Y}} Y_k p(\mathbf{Y}|\boldsymbol{\alpha}) d\mathbf{Y} \\
&= E[Y_k] \\
&= \frac{\alpha_k}{\alpha}
\end{aligned} \tag{2.60}
$$

where the integral is over the whole $(K-1)$-dimension simplex.

Furthermore, the distribution of the $(N+1)$-th observation given previous N observations is:

$$
\begin{aligned}
p(X_{N+1} = \mathcal{X}_k|X_{1:N}) &= \int_{\mathbf{Y}} Y_k p(\mathbf{Y}|\boldsymbol{\alpha}, X_{1:N}) d\mathbf{Y} \\
&= E[Y_k|X_{1:N}] \\
&= \frac{\alpha_k + n_k}{\alpha + N}
\end{aligned} \tag{2.61}
$$

The posterior distribution of an observation given previous observations forms a *Polya Urn* model [JK77] which is proposed as an intuitive explanation for how the Dirichlet distribution works as a prior to the multinomial distribution.

Consider a bag containing α balls of which initially α_k are the colour of k, $1 \leq k \leq K$ (assuming for now that all α_ks are integers). We draw balls from the bag and record the colour of the balls drawn each time. Each time we draw a ball we replace it with two balls with the same colour and put them back in the

bag. If we denote the probability of obtaining a ball of a colour k at the μ-th step as $p(X_\mu = k)$, then the probabilities of the first draw, the second draw and the $(N+1)$-th draw are:

$$p(X_1 = k) = \frac{\alpha_k}{\alpha} \tag{2.62}$$

$$p(X_2 = k|X_1) = \frac{\alpha_k + \delta(X_1 - k)}{\alpha + 1} \tag{2.63}$$

$$p(X_{N+1} = k|X_{1:N}) = \frac{\alpha_k + \sum_{\mu=1}^{N} \delta(X_\mu - k)}{\alpha + N} = \frac{\alpha_k + n_k}{\alpha + N} \tag{2.64}$$

where

$$\delta(X_\mu - k) = \begin{cases} 1 & \text{if } X_\mu = k \\ 0 & \text{if } X_\mu \neq k \end{cases} \tag{2.65}$$

The distribution described by the Polya Urn model in Equation (2.64) is the same as the posterior distribution of an observation given previous observations in Equation (2.61). From the Polya Urn model we discover the cluster effect of the Dirichlet distribution. Unlike the normal drawing balls model, the probability of a colour k will increase if balls with this colour are drawn, which means larger clusters have the advantage of growing themselves in comparison with smaller clusters. This effect will cause the data points to cluster around a smaller number of clusters, which may help mitigate against over-fitting. The Polya Urn model is useful for the inference via sampling methods such as Gibbs sampling, which we will discuss in later chapters.

2.4.2 Dirichlet Process

The Dirichlet Process (DP) is the cornerstone of Bayesian nonparametric statistics. Here, the word "nonparametric" means the number of parameters of the model may grow in the inference process. It is an infinite extension of the Dirichlet distribution. It extends the number of components from a given finite number to a number that grows by the process and can grow as large as is necessary. In some senses, the number of components in the Dirichlet process is a countable infinity.

The Dirichlet process was first proposed in Ferguson's paper *A Bayesian Analysis of Some Nonparametric Problems* [Fer73] in 1973. In the same year, Blackwell

and MacQueen proposed the Polya Urn model for the Dirichlet process [BM73]. In 1994, Sethuraman provided a more explicit characterization of the DP in terms of a stick-breaking construction [Set94]. Since the Dirichlet distribution is used as the prior to mixture models, the DP is also used as the nonparametric prior to mixture models [Ant74]. A mixture model with DP prior is called a *Dirichlet process mixture model*.

Here, we review previous work on the Dirichlet process. First we introduce the concept of the DP. Then we give a brief introduction to the Polya Urn model and the stick-breaking construction as they are the foundations for the Gibbs sampling method and the truncated variational approximation method for DP inference.

Dirichlet Process via Polya Urn Model

The Dirichlet process extends the Dirichlet distribution by extending the event space $\mathcal{X}$ to a continuous space. In the discussion of Dirichlet distribution, we assume the event space contains K events and the $\mathbf{Y}$ is a measure over the space sampled from the Dirichlet distribution with a base measure $\mathbf{G}_0$ and a concentration α as parameters. For any partition $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}, M \leq K$, the distribution of measure $\mathbf{Y}$ over the partition $\mathcal{B}$ still follows a Dirichlet distribution with a base distribution $\mathbf{G}_0$ over the partition $\mathcal{B}$ and a concentration α as parameters.

When we extend the event space to a continuous space, if a distribution which is a distribution of the measures over the space, satisfying that for any random partition over the space, the measures over the partition sampled from this distribution follow a Dirichlet distribution, then this distribution is called a *Dirichlet process*. More formally, we have the following definition [Fer73]:

Definition 2.4. *Let $\mathcal{X}$ be an event space and $\mathcal{A}$ be a σ-field of subsets defined on $\mathcal{X}$. Let $\mathbf{G}_0$ be a non-null finite measure (non-negative and finitely additive) measure on $(\mathcal{X}, \mathcal{A})$ and α be a concentration scalar. We say $\mathbf{Y}$ is a Dirichlet process $DP_{\alpha,\mathbf{G}_0}(\cdot)$ on $(\mathcal{X}, \mathcal{A})$ with parameter $\boldsymbol{\alpha}$ if for every $K = 1, 2, \ldots,$ and measurable partition $(\mathcal{B}_1, \ldots, \mathcal{B}_K)$ of $\mathcal{X}$, the distribution of $(\mathbf{Y}(\mathcal{B}_1), \ldots, \mathbf{Y}(\mathcal{B}_K))$ is Dirichlet, $Dir(\alpha\mathbf{G}_0(\mathcal{B}_1), \ldots, \alpha\mathbf{G}_0(\mathcal{B}_K))$*

Most properties of the Dirichlet distribution are still held in the Dirichlet process. An important property of the Dirichlet process is its discreteness, as shown

in Blackwell's work [Bla73]. The discreteness of DP means that the distribution consists of a countably infinite point of probability masses. It is difficult to understand the Dirichlet process and its discreteness as its joint distribution has no explicit closed form. We do not plan to give details about the mathematical proof or demonstrate this property, but we can show it intuitively via the Polya Urn model. As the Polya Urn model of Dirichlet distribution demonstrates the distributions of the observations, the Polya Urn model for DP also tries to show the distributions of the observations with the intermediate measure $\mathbf{Y}$ integrated out.

Consider a bag of balls with colours as a continuum, the distribution of the colour following a continuous base distribution $\mathbf{G}_0$ and an initial number of balls α. When we draw the first ball from the bag, we have:

$$p(X_1 = c_1) \sim \mathbf{G}_0(\cdot) \tag{2.66}$$

As the same in the Dirichlet distribution, we replace the ball with two balls with the same color c_1 and put them back into the bag. For the second draw, the number of balls becomes $\alpha + 1$ and the distribution is:

$$p(X_2|X_1) = \begin{cases} \frac{1}{\alpha+1}, & \text{if } X_2 = c_1 \\ \frac{\alpha}{\alpha+1}, & \text{if } X_2 \sim \mathbf{G}_0(\cdot) \end{cases} \tag{2.67}$$

For the $(N+1)$-th ball, the posterior is:

$$p(X_{N+1}|X_{1:N}) = \begin{cases} \frac{n_k}{\alpha+N}, & \text{if } X_{N+1} = c_k, k \leq K \\ \frac{\alpha}{\alpha+N}, & \text{if } X_{N+1} \sim \mathbf{G}_0(\cdot) \end{cases} \tag{2.68}$$

Here n_k is the number of balls drawn with the colour c_k and at this stage there are K distinct colours drawn from the bag. Notice that although the base distribution is continuous, the DP itself is discrete with probability one. At any stage, the values observed from a DP have a non-zero probability of occurring again. Compared with the posterior in Dirichlet distribution, in Equation (2.64), we notice that in the DP the probabilities of existed c_k do not contain α_k and there is an extra chance that the new ball is drawn from the base distribution. That is because the base distribution in the Dirichlet distribution is finite and discrete. $g_k = \alpha_k/\alpha$ is the probability mass for each event. However, in DP the

base distribution becomes continuous and the probability for an exact point (or colour in the Polya Urn model) is 0. Therefore, there is no such g_k in DP. In fact, we can also rewrite Equation (2.64) to a form similar to Equation (2.68) to illustrate the differences and connections of the Dirichlet distribution and the DP more clearly.

$$p(X_{N+1}|X_{1:N}) = \begin{cases} \frac{n_k}{\alpha+N}, & \text{if } X_{N+1} = c_k, k \leq K \\ \frac{\alpha}{\alpha+N}, & \text{if } X_{N+1} \sim Multinomial(g_1, \ldots, g_K) \end{cases} \tag{2.69}$$

In Equation (2.69), the number K is a fixed finite positive integer given by the base distribution $\mathbf{G}_0$. In the DP, the K in an unlimited integer as the base distribution is continuous.

By Equation (2.68), we can observe that the DP also shows a cluster effect as it does in the Dirichlet distribution. If a ball of colour c_k is drawn, the probability of drawing this colour again is increased. The probability of drawing balls with new colour decreases by the process of drawing and replacing. It is easy to show that the expectation of the number of colours K at the N-th step is $\sum_{i=1}^{N-1} \frac{\alpha}{\alpha+i}$.

$$\begin{aligned} E_1(K) &= 1 \\ E_2(K) &= \frac{1}{\alpha+1} E_1(K) + \frac{\alpha}{\alpha+1}[E_1(K)+1] = 1 + \frac{\alpha}{\alpha+1} \\ &\vdots \\ E_N(K) &= \frac{N-1}{\alpha+N-1} E_{N-1}(K) + \frac{\alpha}{\alpha+N-1}[E_{N-1}(K)+1] \\ &= E_{N-1}(K) + \frac{\alpha}{\alpha+N-1} \\ &= \sum_{i=1}^{N-1} \frac{\alpha}{\alpha+i} \approx \alpha \ln N \end{aligned} \tag{2.70}$$

The term $\sum_{i=1}^{N-1} \frac{\alpha}{\alpha+i}$ is the $(N-1)$-th partial sum of a *Harmonic series* which does not converge but increases very slowly. If we set $\alpha = 1$, then we need $N > 10^{43}$ to make the expectation of K reach 100 [BW71].

One should notice that this estimation of K is only estimated based on the Dirichlet process. We do not consider the situation of a specific dataset. In other words, this is only the estimation of the prior. The likelihood of the data is not included. In practice, the estimation and the exact result of K always combines

the situation both on the prior side and the likelihood side.

In summary, we list three important properties established in [Bla73, BM73], which are useful in Bayesian nonparametric inference.

Property 2.2. *Dirichlet process is a probability measure on the space of probability measures on* $(\mathcal{X}, \mathcal{A})$.

Property 2.3. *Dirichlet process gives a probability of one to the subset of all discrete probability measures on* $(\mathcal{X}, \mathcal{A})$.

Property 2.4. *The posterior distribution* $DP_{\alpha,\mathbf{G}_0}(\cdot|X_{1:N})$ *is the Dirichlet measure* $DP_{\alpha+N,\mathbf{G}_0+\delta_{\mathbf{X}}}(\cdot)$ *where* $\delta_{\mathbf{X}}$ *is the probability measure degenerate at* $\mathbf{X} = (X_1, \ldots, X_N)$ *and* $\mathbf{G}_0+\delta_{\mathbf{X}}$ *is an abbreviation of the probability measure* $\frac{\alpha}{\alpha+N}\mathbf{G}_0(\cdot)+\frac{N}{\alpha+N}\delta_{\mathbf{X}}(\cdot)$.

The Polya Urn model of DP is important as it provides the foundation for inference of DP by sampling. The details of this model used in the inference of Dirichlet process mixture model will be shown in Chapter 4.

Dirichlet Process via Stick-Breaking Construction

Another important work on the Dirichlet process is the stick-breaking construction proposed by Sethuraman in 1994 [Set94]. In his work, Sethuraman provided another definition of the DP and proved that his definition and Ferguson's definition are equivalent. He also showed that the stick-breaking construction also satisfies the Property 2.2, 2.3 and 2.4. In this part we describe the content of this definition whilst we continue to omit its mathematical proof about it.

As the same in Ferguson's definition, at the beginning we have a concentration scalar α and a base distribution $\mathbf{G}_0$. Consider two infinite collections of independent variables, $v_k \sim Beta(1, \alpha) = \frac{1}{\Gamma(\alpha)} v_k(1-v_k)^\alpha$ and $c_k \sim \mathbf{G}_0(\cdot)$ for $k = 1, 2, \ldots$. The stick-breaking construction process is as follows:

$$\pi_k(\mathbf{v}) = v_k \prod_{l=1}^{k-1}(1 - v_l) \tag{2.71}$$

$$DP_{\alpha,\mathbf{G}_0}(\cdot) = \sum_{k=1}^{\infty} \pi_k(\mathbf{v})\delta_{c_k} \tag{2.72}$$

This representation of the DP clearly demonstrates its discreteness. By the stick-breaking representation, DP consists of countably infinite masses. The probability mass of each point is π_k. Notice that the difference between 1 and the sum of π_k is 0:

$$\begin{aligned}
1 - \sum_{k=1}^{\infty} \pi_k(\mathbf{v}) &= 1 - \lim_{K\to\infty} \sum_{k=1}^{K} \pi_k(\mathbf{v}) \\
&= \lim_{K\to\infty} \left[1 - \sum_{k=1}^{K} v_k \prod_{l=1}^{k-1} (1 - v_l) \right] \\
&= \lim_{K\to\infty} \left[1 - v_1 - (1 - v_1)v_2 - \dots \right] \\
&= \lim_{K\to\infty} \left[(1 - v_1)(1 - v_2) - (1 - v_1)(1 - v_2)v_3 - \dots \right] \\
&= \lim_{K\to\infty} \prod_{k=1}^{K} (1 - v_k) = 0
\end{aligned} \tag{2.73}$$

On the other hand the expectation of π_k is:

$$\begin{aligned}
E[\pi_k(\mathbf{v})] &= E\left[v_k \prod_{l=1}^{k-1} (1 - v_l) \right] \\
&= E[v_k] \prod_{l=1}^{k-1} E[(1 - v_l)] \\
&= \left(\frac{\alpha}{\alpha + 1} \right)^{k-1} \frac{1}{\alpha + 1} \xrightarrow{k\to\infty} 0
\end{aligned} \tag{2.74}$$

Equation (2.73) and (2.74) show that for a K large enough, the DP can be well approximated by a sum of finite K masses. This conclusion is useful for the truncated approximation of the DP. Further use of the stick-breaking construction in the Dirichlet process mixture model will be discussed in Chapter 4.

2.5 Markov Chain Monte Carlo and Gibbs Sampling

Here, we introduce the *Markov Chain Monte Carlo* (MCMC) sampling method and its special case the *Gibbs* sampler [GG84] for multi-variant sampling. First,

we briefly introduce the *Monte Carlo* method, *Markov Chain* and the *Metropolis-Hasting Algorithm* [MU49, MRR+53, Has70] for MCMC. Subsequently, we introduce the *Gibbs* sampler as a special case of the *Metropolis-Hasting Algorithm*.

2.5.1 Markov Chain Monte Carlo

The famous *Monte Carlo* approach was first developed to use random generation for computing integrals. For a given function $h(x)$, if it can be decomposed to a production of a function and a probability density function: $h(x) = f(x)p(x)$ defined on the interval $[a, b]$, then the integral of $h(x)$ over $[a, b]$ is:

$$\int_a^b h(x)dx = \int_a^b f(x)p(x)dx = E_{p(x)}[f(x)] \tag{2.75}$$

By the law of large numbers, if we draw a large number of $x_1, x_2, \ldots, x_N$ from $p(x)$, then the expectation of $f(x)$ can be approximated by the mean:

$$\int_a^b h(x)dx = E_{p(x)}[f(x)] \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n) \tag{2.76}$$

For Bayesian inference, Monte Carlo integration is often used to approximate posterior or marginal distributions as most of these distributions in Bayesian analysis are analytically intractable. For example, a marginal density of y can be approximated by

$$I(y) = \int f(y|x)p(x)dx \approx \frac{1}{N}\sum_{n=1}^{N} f(y|x_n)$$

where x_n are drawn from $p(x)$.

Sometimes, if we want to calculate the expectation of a function with an inconvenient distribution, such as only knowing the distribution proportionally, we can use a roughly approximate distribution as a replacement:

$$\int f(x)q(x)dx = \int f(x)\left(\frac{q(x)}{p(x)}\right)p(x)dx \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n)\left(\frac{q(x_n)}{p(x_n)}\right) \tag{2.77}$$

where x_n are drawn from $p(x)$. This form of sampling is called the *Important Sampling*.

Markov Chain is a probability model used to describe a sequence of random variables. Let $\{X_1, \ldots, X_N\}$ denote a sequence of random variables, $S = s_1, \ldots, s_M$ denote the state space referring to the range of possible X values. At this stage, we assume the state space is finite for convenience. It can be easily extended to become continuous at a later point. The Markov chain assumes that the distribution of t-th variable only depends on the state of the $(t-1)$-th variable:

$$p(X_t|X_1, \ldots, X_{t-1}) = p(X_t|X_{t-1}) \tag{2.78}$$

The conditional probability of X_t given X_{t-1} is called the *transition probability* or the *transition kernel*. In our MCMC background, we assume the Markov chain is time-homogenous, meaning that the transition probability is the same for every t. The transition probability is often written as:

$$P(i,j) = P(i \to j) = p(X_t = s_j|X_{t-1} = s_i) \tag{2.79}$$

For the finite state space S, we can define the transition probability matrix $\mathbf{P}$ as the $M \times M$ matrix whose (i,j)-th element is $P(i,j)$. Consider the marginal distribution of the t-th variable. We use $\pi_i(t)$ denoting the probability that the t-th variable is at the i-th state and use $\boldsymbol{\pi}(t) = (\pi_1(t), \ldots, \pi_M(t))$ for the distribution of the t-th variable. Then the following equation holds:

$$\pi_j(t) = \sum_{i=1}^{M} P(i,j)\pi_i(t-1) \tag{2.80}$$

Alternatively, it can be written in the vector form:

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(t-1)\mathbf{P} \tag{2.81}$$

Equations (2.80) and (2.81) are called the Chapman-Kolomogrov equation. The Markov chain may reach a stationary distribution that there is a $\boldsymbol{\pi}^*$, satisfying:

$$\boldsymbol{\pi}^* = \boldsymbol{\pi}^*\mathbf{P} \tag{2.82}$$

When the Markov chain reaches the stationary distribution, the probability of any state is independent of its initial probability.

For a given distribution $\boldsymbol{\pi}$, if we want to choose a transition matrix $\mathbf{P}$ to

satisfy Equation (2.82), it is sufficient if:

$$P(i,j)\pi_i = P(j,i)\pi_j \tag{2.83}$$

This condition is called the *reversibility condition*. It is easy to show that Equation (2.83) implies $\boldsymbol{\pi}^* = \boldsymbol{\pi}^*\mathbf{P}$:

$$(\boldsymbol{\pi}\mathbf{P})_j = \sum_i \pi_i P(i,j) = \sum_i \pi_j P(j,i) = \pi_j \tag{2.84}$$

For continuous state space, the transition matrix becomes a function $P(x,y) = P(x \to y)$ with:

$$\int P(x,y)dy = 1 \tag{2.85}$$

The Chapman-Kolomogrov equation becomes:

$$\pi_t(y) = \int P(x,y)\pi_{t-1}(x)dx \tag{2.86}$$

and Equation (2.82) and (2.83) become:

$$\pi^*(y) = \int P(x,y)\pi_*(x)dx \tag{2.87}$$

$$P(x,y)\pi_x = P(y,x)\pi_y \tag{2.88}$$

The *Metropolis-Hasting* algorithm is a method that can sample from complex distributions by constructing a transition kernel satisfying the reversibility condition to form a Markov chain with the unique stationary distribution as the target distribution. Suppose the target distribution is $p(x) = f(x)/Z$ where Z is the normalization constant and is unknown and hard to compute. We can generate a sequence of draws from $p(x)$ using the following steps:

1. Start from any x_0 satisfying $f(x_0) > 0$.

2. Using the current sample value x, generate a candidate point x^* from a user specific jumping distribution $q(x, x^*)$, which is the probability of returning x^* given x. This distribution is also called the proposal or candidate-generating function.

3. Given the candidate point x^* and the previous point x, calculate the acceptance rate α:
$$\alpha = \min\left(\frac{f(x^*)q(x^*,x)}{f(x)q(x,x^*)}, 1\right)$$

4. Draw a number $\beta \in [0,1]$ from a uniform distribution. If $\beta < \alpha$ accept the new point as a sample, else reject the new point and keep the old point. Go to step 2 until convergence.

This is the *Metropolis-Hasting* algorithm. If the jumping function is symmetric with $q(x,x^*) = q(x^*,x)$, then this algorithm becomes the original *Metropolis* algorithm. The sequence $(x_0, x_1, \dots)$ forms a Markov chain. We will show that the stationary distribution of this Markov chain is our goal distribution $p(x)$. To show this, we only need to show that the reversibility condition is satisfied:

$$\begin{aligned} & P(x,y)p(x) = P(y,x)p(y) \\ \Rightarrow\ & q(x,y)\alpha(x,y)p(x) = q(y,x)\alpha(y,x)p(y) \text{ for all } x,y \end{aligned} \tag{2.89}$$

As Equation (2.89) is symmetric for x and y, we only need to discuss the following two cases:

1. $q(x,y)p(x) = q(y,x)p(y)$. Here $\alpha(x,y) = \alpha(y,x) = 1$, therefore we have:
$$q(x,y)\alpha(x,y)p(x) = q(y,x)\alpha(y,x)p(y)$$

2. $q(x,y)p(x) > q(y,x)p(y)$. The acceptance rate
$$\alpha(x,y) = \frac{p(y)q(y,x)}{p(x)q(x,y)} \quad \text{and} \quad \alpha(y,x) = 1.$$
Hence:
$$\begin{aligned} q(x,y)\alpha(x,y)p(x) &= q(x,y)\frac{p(y)q(y,x)}{p(x)q(x,y)}p(x) \\ &= q(y,x)p(y) \\ &= q(y,x)\alpha(y,x)p(y) \end{aligned}$$

2.5.2 Gibbs Sampler

Here, we discuss the *Gibbs sampler*. The Gibbs sampler is a special case of the Metropolis-Hasting algorithm wherein the new proposal point is always accepted. It is used for sampling from a multi-variant distribution. For a distribution $p(x_1, \ldots, x_M)$, if we know the conditional distributions $p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_M)$ for $j \in (1, \ldots, M)$, we can sample $p(x_1, \ldots, x_M)$ in the following way:

1. Choose a start point $X^{(0)} = (x_1^{(0)}, \ldots, x_M^{(0)})$.

2. Update the variables of the current point one by one by sampling new values from the conditional distribution until convergence. For the variable $x_j^{(i)}$, the new value is drawn from the distribution:

$$x_j^{(i)} \sim p(x_j^{(i)}|x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \ldots, x_M^{(i-1)})$$

We can show that this scheme satisfies the reversibility condition. In this Markov chain, the step is updating one variable of the current point. If we consider Gibbs sampling in the Metropolis-Hasting algorithm framework, then for each sampling step, the jumping function is the conditional distribution function and the acceptance rate is always 1. Notice that the jumping function $q(x_j^{(i-1)}, x_j^{(i)})$ is irrelevant to the current value $x_j^{(i-1)}$. We have:

$$\begin{aligned} q(x_j^{(i-1)}, x_j^{(i)}) &= p(x_j^{(i)}|x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \ldots, x_M^{(i-1)}) \\ &= \frac{p(x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_j^{(i)}, x_{j+1}^{(i-1)}, \ldots, x_M^{(i-1)})}{p(x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \ldots, x_M^{(i-1)})} \end{aligned} \tag{2.90}$$

$$\begin{aligned} \text{and} \quad q(x_j^{(i)}, x_j^{(i-1)}) &= p(x_j^{(i-1)}|x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \ldots, x_M^{(i-1)}) \\ &= \frac{p(x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_j^{(i-1)}, x_{j+1}^{(i-1)}, \ldots, x_M^{(i-1)})}{p(x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \ldots, x_M^{(i-1)})} \end{aligned} \tag{2.91}$$

By Equation (2.90) and (2.91), we can easily obtain:

$$q(x_j^{(i-1)}, x_j^{(i)})p(\ldots, x_j^{(i-1)}, \ldots) = q(x_j^{(i)}, x_j^{(i-1)})p(\ldots, x_j^{(i)}, \ldots) \tag{2.92}$$

This is the reversibility condition for Gibbs sampling. The Gibbs sampling is very useful in Bayesian statistics. Apart from the variational methods, the Gibbs sampler plays an essential role in the Bayesian inference. In Chapters 3 and 4 we

will introduce how Gibbs sampling is used in our mixture model.

2.6 Summary

In this chapter we reviewed the related background of our work. We use 5 sections to demonstrate them one by one. At the end of this chapter, we reorganize them as a summary.

In Section 2.2, we mainly focus on the introduction of the frequent itemset mining (FIM). We firstly introduced the basic concepts, terminologies and notations of FIM. In the second part of this section we demonstrated two main stream algorithms for FIM: the Apriori algorithm and the Eclat algorithm. The scheme of the Eclat algorithm will be further used in the process of generating the set of frequent itemsets by our proposed models. In the third part of this section, we reviewed the main problem of normal FIM and the recent developments of this research field. We conclude the developments into three categories: mining compressed itemsets, looking for better measurement of interestingness and developing probability models.

In Section 2.3, we introduced the mixture model and the classic algorithm, the Expectation-Maximization (EM) algorithm, to do the parameter inference of the mixture model. We also applied the EM algorithm on the transaction dataset to deduce the non-Bayesian mixture model for our case. We provided two explanations of the EM algorithm as the first one is fundamental and the second one is very useful in our latter work on variational EM algorithm. We also showed the method to do the inference including obtaining the marginal and joint probabilities via a trained model.

In Section 2.4, we focus on the prior distributions of Bayesian mixture models. The Dirichlet distribution is the conjugate prior for the multinomial distribution which controls the component selection. In order to extend the number of components from fixed finite to a state of growing as necessary, we also extend the Dirichlet distribution to the Dirichlet Process (DP). The DP is a discrete distribution and allows new draws from the base distribution. For both distribution we introduced their definitions, some useful properties and their Polya Urn models respectively. The Polya Urn models makes the distributions easier to understand and more importantly, they are the theoretical fundamental for using Gibbs sampling to the inference. We also briefly introduced the stick-breaking construction

of the DP. The stick-breaking representation of the DP makes the truncated variational approximation for the inference of the DP mixture model possible.

In Section 2.5, we described the Markov Chain Monte Carlo (MCMC) sampling and the Gibbs sampling as a special form of the MCMC. We introduced the mechanism of the MCMC sampling and demonstrated how to use the Gibbs sampling for Bayesian inference.

In next chapter, we will focus on evolving the non-Bayesian mixture model to the finite Bayesian mixture model. We will describe the structure of the Bayesian mixture model, and show how to use the Gibbs sampling and the variational EM algorithm to do the parameter inference of the model and how to use the model to obtain the marginal and joint probabilities of a given query.

Chapter 3

Finite Bayesian Mixture Model

In Chapter 2 we introduced the Bernoulli Mixture model for transaction datasets. The Bernoulli Mixture model in Chapter 2 is a non-Bayesian model. In the model inference, we want to obtain the parameters that maximized the likelihood of the model. However, from a Bayesian view, the model parameters are random variables themselves. Instead of finding the value of the parameters, the Bayesian approach tries to find the optimal posterior distribution of the parameters. In comparison with non-Bayesian machine learning methods, Bayesian approaches have several valuable advantages. Firstly, Bayesian integration does not suffer from overfitting, because it does not simply fit parameters to the data. Rather, it integrates over all parameters which are weighted by how well they fit the data. Secondly, prior knowledge can be incorporated naturally and all uncertainty manipulated in a consistent manner. One famous application of the finite Bayesian mixture model is the *Latent Dirichlet Allocation* [BNJ03] developed for natural language processing.

Two main methods for Bayesian inference are MCMC sampling and the variational EM algorithm. Both approaches have their strengths and weaknesses. MCMC sampling is guaranteed to achieve a well-approximated posterior distribution of the components given a large enough number of iterations. However, it can be slow to converge and the convergence hard to diagnose. On the other hand, whilst variational inference is much faster than MCMC sampling, the EM algorithm is often trapped into a local minimum and the approximation assumption of the variation inference makes the variational inference less accurate than MCMC sampling.

In this chapter we introduce the assumption and structure of the finite Bayesian

mixture model and then we introduce both approaches. The experiments and related discussions will be given in Chapter 5.

3.1 Assumption and Structure

The difference between the Bayesian and non-Bayesian mixture models is that the Bayesian mixture tries to form a smooth distribution over the model parameters by introducing proper priors. The origin mixture model introduced in the previous chapter is a two-layer model. The top layer is the multinomial distribution for choosing the mixtures, and the next layer is the Bernoulli distribution for items. Into the Bayesian mixture we introduce a Dirichlet distribution as the prior to the multinomial parameter $\boldsymbol{\pi}$ and Beta distributions as the priors of the Bernoulli parameters $\{\phi_{ik}\}$.

The Dirichlet distribution is the conjugate prior of the multinomial distribution. The probabilities of the components of the mixture model, denoted as $\boldsymbol{\pi}$, are treated as random variables in Bayesian analysis. They are assumed to follow a Dirichlet distribution as the Dirichlet distribution has a similar form as the multinomial distribution so that it is convenient to do further mathematical manipulation. To make this clear, we write the multinomial distribution as the following form:

$$p(z|\boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{\delta(z-k)} \tag{3.1}$$

Comparing it with Equation (2.50), we can notice that the form of the two equations are very similar. By this manner, we can incorporate the likelihood and the prior very easily. The parameters of the Dirichlet distribution are called the hyper-parameters of the model. If we have any prior knowledge that should be considered in the process of learning, we can adjust the hyper-parameters to reflect our prior knowledge. In our case as we have no special preference for a certain component, we set a symmetric prior by a positive scalar α. α is also called the concentration parameter because a larger α implies that the distribution of the parameters will be more concentrated to the expectation value. As the prior is symmetric, the hyper-parameter for each component is α/K where K is the number of components.

The conjugate prior for the Bernoulli distribution is the Beta distribution. The Beta distribution has two parameters for each Bernoulli parameter. As

we have no preference for components, the parameters for the same item but different component can share the same hyper-parameters. It is natural to use the frequency of each item over all dataset as a reference to select the hyper-parameters. The exact settings of the hyper-parameters will be introduced in Chapter 5.

After adding priors to the model, the new model is as follows.

1. Assign $\alpha, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as the hyperparameters of the model, where α, is positive scalar and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_D)$ and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_D)$ where $\beta_i > 0, \gamma_i > 0$ for all $i = 1, \ldots, D$.

2. Choose $\boldsymbol{\pi} \sim$Dir$(\alpha/K, \alpha/K, \ldots, \alpha/K)$ where

$$p(\boldsymbol{\pi}|\alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} \prod_{k=1}^{K} \pi_k^{\alpha/K-1} \tag{3.2}$$

   with $\sum_{k=1}^{K} \pi_k = 1$.

3. For each item and component choose $\phi_{ik} \sim$Beta(β_i, γ_i) where

$$p(\phi_{ik}|\beta_i, \gamma_i) = \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \phi_{ik}^{\beta_i - 1}(1 - \phi_{ik})^{\gamma_i - 1} \tag{3.3}$$

   with $\phi_{ik} \in [0, 1]$ where $i \in \{1, \ldots, D\}, k \in \{1, \ldots, K\}$

4. For each transaction $\mathbf{X}^{\mu}$:

   (a) Choose a component $z^{\mu} \sim$Multinomial$(\boldsymbol{\pi})$, where

$$p(z^{\mu} = k|\boldsymbol{\pi}) = \pi_k \tag{3.4}$$

   (b) Then we can generate data by:

$$p(\mathbf{X}_i|z, \boldsymbol{\phi}) = \prod_{i=1}^{D} \phi_{iz}^{x_i}(1 - \phi_{iz})^{1-x_i} \tag{3.5}$$

   Here we omit μ for the sake of simplicity.

This process can be briefly written as:

$$\boldsymbol{\pi}|\alpha \sim \text{Dir}(\alpha/K, \alpha/K, \ldots, \alpha/K)$$

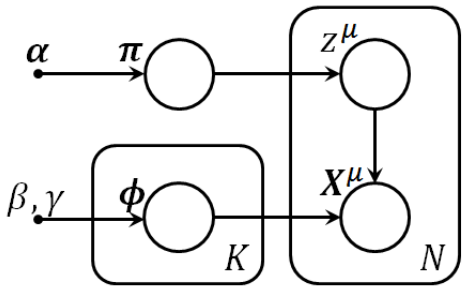


Figure 3.1: finite Bayesian mixture graphic representation

$$\begin{aligned} \boldsymbol{\phi}_k|\boldsymbol{\beta},\boldsymbol{\gamma} &\sim \text{Beta}(\boldsymbol{\beta},\boldsymbol{\gamma}) \\ z^\mu|\boldsymbol{\pi} &\sim \text{Multi}(\boldsymbol{\pi}) \\ \mathbf{X}^\mu|z^\mu,\boldsymbol{\phi} &\sim \text{Ber}(\mathbf{X}^\mu|\boldsymbol{\phi}_{z^\mu}) \end{aligned} \tag{3.6}$$

Figure 3.1 is a graphic representation for Bayesian mixtures. For easier model comparison, we use the same notation in the non-Bayesian, finite Bayesian and the Dirichlet process mixture models in the next chapter when this causes no ambiguity. In contrast with the non-Bayesian mixture model, the Bayesian mixture model treats parameters as random variables and uses hyperparameters to control them. Based on the model assumption, the joint probability of the transaction $\mathbf{X}^\mu$, components indicator z^μ and the model parameters $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ is:

$$p(\mathbf{X}^\mu, z^\mu, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) = p(\mathbf{X}^\mu|z^\mu, \boldsymbol{\phi})p(z^\mu|\boldsymbol{\pi})p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma})p(\boldsymbol{\pi}|\alpha) \tag{3.7}$$

For the whole dataset:

$$p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{\mu=1}^{N}[p(\mathbf{X}^\mu|z^\mu, \boldsymbol{\phi})p(z^\mu|\boldsymbol{\pi})]p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma})p(\boldsymbol{\pi}|\alpha) \tag{3.8}$$

Integrating over $\boldsymbol{\pi}$, $\boldsymbol{\phi}$, summing over $\mathcal{Z}$ and applying a logarithm, we obtain the log-likelihood of the dataset:

$$\ln p(\mathcal{T}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \ln \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) d\boldsymbol{\phi} d\boldsymbol{\pi} \tag{3.9}$$

Here the integral over $\boldsymbol{\pi}$ means an integral over a $(K-1)$-dimension simplex. The integral over $\boldsymbol{\phi}$ means an integral over a $K \times D$ vector $\boldsymbol{\phi} \in [0,1]^{K \times D}$. The summing over $\mathcal{Z}$ is summing over all possible $\mathcal{Z}$ configurations. This integral is

intractable because of the coupling of $\mathcal{Z}$ and $\boldsymbol{\pi}$. The coupling of $\mathcal{Z}$ and $\boldsymbol{\pi}$ also makes it difficult to optimize via a normal EM algorithm.

3.2 Inference via Gibbs Sampling

The idea of performing Bayesian inference via Gibbs sampling considers the parameters of the model as a vector and samples the distribution of the parameters. In our application, as mentioned in the previous section, the difficulty lies in the assignment of the component indicators as the proportion parameters $\boldsymbol{\pi}$ are unknown and cannot be dealt with separately. However, the conditional probabilities $\boldsymbol{\phi}$ follow simple Beta distributions and their value can be easily estimated if the component indicators are fixed.

Therefore, in our case we do not sample all parameters but only the component indicators. This is collapsed Gibbs sampling. According to the Gibbs sampling framework, we consider $\mathcal{Z}$ as the vector we want to sample and sample one variable (z^μ for a transaction) each time by keeping other indicators fixed. To do this, we need to obtain the conditional probability of z^μ given the data and the rest values of indicators $\mathcal{Z}_{-\mu}$. Based on Bayes rule:

$$p(z^\mu = k|\mathcal{Z}_{-\mu}, X^\mu) \propto \underbrace{p(X^\mu|z^\mu = k)}_{\text{likelihood}} \underbrace{p(z^\mu = k|\mathcal{Z}_{-\mu})}_{\text{prior}} \tag{3.10}$$

Here, the prior distribution is controlled by the Dirichlet distribution. As discussed in Chapter 2, if $\boldsymbol{\pi}$ follow Dirichlet distribution and they are parameters for a multinomial distribution, the μ-th observation's distribution given the rest of the observations is Equation (2.64). That is:

$$p(z_\mu = k|\mathcal{Z}_{-\mu}) = \frac{\alpha/K + n_{k/\{\mu\}}}{\alpha + N - 1} \tag{3.11}$$

where $n_{k/\{\mu\}}$ is the number of points assigned to components k except X^μ.

The likelihood is a little complicated as the parameters $\boldsymbol{\phi}$ should be estimated first. We have:

$$p(X^\mu|z^\mu = k) = \int_{\boldsymbol{\phi}_k} p(X^\mu|\boldsymbol{\phi}_k) p(\boldsymbol{\phi}_k|\mathcal{Z}_{-\mu}, \mathcal{T}_{-\mu}) d\boldsymbol{\phi} \tag{3.12}$$

where:

$$
\begin{aligned}
p(\boldsymbol{\phi}_k|\mathcal{Z}_{-\mu},\mathcal{T}_{-\mu}) &= \frac{1}{C}p(\mathcal{Z}_{-\mu}|\boldsymbol{\phi}_k,\mathcal{Z}_{-\mu})p(\boldsymbol{\phi}_k|\beta,\gamma) \\
&= \frac{1}{C}\prod_{\mu'\neq\mu}^{N}\prod_{i=1}^{D}[\phi_{ik}^{x_i^{\mu'}}(1-\phi_{ik})^{1-x_i^{\mu'}}]^{\delta(z^{\mu'}=k)}\cdot\prod_{i=1}^{D}\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)}\phi_{ik}^{\beta_i-1}(1-\phi_{ik})^{\gamma_i-1} \\
&= \frac{1}{C}\prod_{i=1}^{D}\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)}\phi_{ik}^{\eta_{ik}-1}(1-\phi_{ik})^{\nu_{ik}-1} \qquad (3.13)
\end{aligned}
$$

where C is the normalization constant

$$
C=\sum_{k'=1}^{K}p(\mathcal{Z}_{-\mu}|\boldsymbol{\phi}_{k'},\mathcal{T}_{-\mu})p(\boldsymbol{\phi}_{k'}|\boldsymbol{\beta},\boldsymbol{\gamma}) \qquad (3.14)
$$

and

$$
\eta_{ik}=\beta_i+\sum_{\mu'\neq\mu}^{N}\delta(z^{\mu'}-k)x_i^{\mu'} \qquad (3.15)
$$

$$
\nu_{ik}=\gamma_i+n_{k/\{\mu\}}-\sum_{\mu'\neq\mu}^{N}\delta(z^{\mu'}-k)x_i^{\mu'} \qquad (3.16)
$$

where (3.17)

$$
\delta(z^{\mu'}-k)=\begin{cases}1 & \text{if } z^{\mu'}=k\\ 0 & \text{if } z^{\mu'}\neq k\end{cases} \qquad (3.18)
$$

Then we can obtain $p(X^\mu|z^\mu=k)$ by integrating $\boldsymbol{\phi}_k$ out:

$$
\begin{aligned}
p(X^\mu|z^\mu=k) &= \int_{\boldsymbol{\phi}_k}p(X^\mu|\boldsymbol{\phi}_k)p(\boldsymbol{\phi}_k|\mathcal{Z}_{-\mu},\mathcal{T}_{-\mu})d\boldsymbol{\phi} \\
&= \int_{\boldsymbol{\phi}_k}\prod_{i=1}^{D}\phi_{ik}^{x_i^\mu}(1-\phi_{ik})^{1-x_i^\mu}\cdot\frac{1}{C}\prod_{i=1}^{D}\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)}\phi_{ik}^{\eta_{ik}-1}(1-\phi_{ik})^{\nu_{ik}-1}d\boldsymbol{\phi} \\
&= \frac{1}{C}\prod_{i=1}^{D}\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)}\int_0^1\phi_{ik}^{x_i^\mu+\eta_{ik}-1}(1-\phi_{ik})^{1-x_i^\mu+\nu_{ik}-1}d\phi_{ik} \\
&= \frac{1}{C}\prod_{i=1}^{D}\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)}\frac{\Gamma(x_i^\mu+\eta_{ik})\Gamma(1-x_i^\mu+\nu_{ik})}{\Gamma(\eta_{ik}+\nu_{ik}+1)} \\
&= \frac{1}{C}\prod_{i=1}^{D}\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)}\frac{\eta_{ik}^{x_i^\mu}\Gamma(\eta_{ik})\gamma_{ik}^{1-x_i^\mu}\Gamma(\nu_{ik})}{(\eta_{ik}+\nu_{ik})\Gamma(\eta_{ik}+\nu_{ik})}
\end{aligned}
$$

Algorithm 3.1 collapsed Gibbs sampling for finite Bayesian mixture model

initialize α, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and $\mathcal{Z}$
repeat
  for $\mu = 1$ to N **do**
    Update η_{ik}, ν_{ik} by
      $\eta_{ik} = \beta_i + \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'}$
      $\nu_{ik} = \gamma_i + n_{k/\{\mu\}} - \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'}$
    Calculate multinomial probabilities based on
      $p(z^\mu = k) = \frac{\alpha/K + n_{k/\{\mu\}}}{\alpha + N - 1} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{x_i^\mu} \left(\frac{\nu_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{1-x_i^\mu}$
    Normalize $p(z^\mu = k)$
    Sample z^μ based on $p(z^\mu = k)$
  end for
until convergence

$$= \frac{1}{C'} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}}\right)^{x_i^\mu} \left(\frac{\nu_{ik}}{\eta_{ik} + \nu_{ik}}\right)^{1-x_i^\mu} \tag{3.19}$$

where C' is the normalization factor.

Combine Equation (3.11) and (3.19), the posterior of the indicator z^μ is:

$$p(z^\mu = k | \mathcal{Z}_{-\mu}, \mathcal{T}) \propto \frac{\alpha/K + n_{k/\{\mu\}}}{\alpha + N - 1} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}}\right)^{x_i^\mu} \left(\frac{\nu_{ik}}{\eta_{ik} + \nu_{ik}}\right)^{1-x_i^\mu} \tag{3.20}$$

The whole process of the collapsed Gibbs sampling for the finite Bayesian mixture model is shown in Algorithm 3.1.

The time cost for Gibbs sampling is analyzed as follows. The updating of β_{ik}, γ_{ik} is $O(DK)$. The calculation of $p(z^\mu = k)$ is $O(DK)$. If we think of the process of sampling from the first data point to the last as an iteration, then the time cost for an iteration is $O(NDK)$.

With the result of sampling, we can perform the predictive inference. For a proposed transaction $\hat{X} = (\hat{x}_1, \ldots, \hat{x}_D), \hat{x}_i \in \{0, 1\}$, its probability by the model is the following integral:

$$\begin{aligned} p(\hat{X}|X_{1:N}) &= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{k=1}^{K} p(\hat{X}|\hat{z} = k, \boldsymbol{\phi}_k) p(\hat{z}|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\alpha, X_{1:N}) p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma}, X_{1:N}) d\boldsymbol{\pi} d\boldsymbol{\phi} \\ &= \sum_{k=1}^{K} \int_{\boldsymbol{\phi}} p(\hat{X}|\hat{z} = k, \boldsymbol{\phi}_k) \left[\int_{\boldsymbol{\pi}} \pi_k p(\boldsymbol{\pi}|\alpha, X_{1:N}) d\boldsymbol{\pi} \right] p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma}, X_{1:N}) d\boldsymbol{\phi} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=1}^{K} \frac{\alpha/K + n_k}{\alpha + N} \prod_{i=1}^{D} \int_{\phi_{ik}} \phi_{ik}^{\hat{x}_i} (1-\phi_{ik})^{1-\hat{x}_i} p(\phi_{ik}|\boldsymbol{\beta}, \boldsymbol{\gamma}, X_{1:N}) d\phi_{ik} \\
&= \sum_{k=1}^{K} \frac{\alpha/K + n_k}{\alpha + N} \prod_{i=1}^{D} \int_{\phi_{ik}} \frac{\Gamma(\eta_{ik} + \nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} \phi_{ik}^{\hat{x}_i + \eta_{ik} - 1} (1-\phi_{ik})^{1-\hat{x}_i + \nu_{ik} - 1} d\phi_{ik} \\
&= \sum_{k=1}^{K} \frac{\alpha/K + n_k}{\alpha + N} \prod_{i=1}^{D} \frac{\Gamma(\eta_{ik} + \nu_{ik})\Gamma(\hat{x}_i + \eta_{ik})\Gamma(1 - \hat{x}_i + \nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})\Gamma(\eta_{ik} + \nu_{ik} + 1)} \\
&= \sum_{k=1}^{K} \frac{\alpha/K + n_k}{\alpha + N} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{\hat{x}_i} \left(\frac{\nu_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{1-\hat{x}_i}
\end{aligned} \tag{3.21}
$$

where n_k is the number of points assigned to component k, and:

$$\eta_{ik} = \beta_i + \sum_{\mu=1}^{N} \delta(z^\mu - k) x_i^\mu \tag{3.22}$$

$$\nu_{ik} = \gamma_i + n_k - \sum_{\mu=1}^{N} \delta(z^\mu - k) x_i^\mu \tag{3.23}$$

In the integration of Equation (3.45), we use the property of the posterior distribution of Dirichlet and Beta distribution shown by Equation (2.59). Similarly, for an itemset, the problem becomes a marginal probability query. For an itemset I, its probability is:

$$p(I|X_{1:N}) = \sum_{k=1}^{K} \frac{\alpha/K + n_k}{\alpha + N} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{\delta(i \in I)} \tag{3.24}$$

where $\delta(i \in I)$ is the indicator that:

$$\delta(i \in I) = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{if } i \notin I \end{cases} \tag{3.25}$$

The time cost for predicting an itemset is $O(K(N_I + 1))$ where N_I is the length of the itemset. When making a predictive inference, we only need to take care of the value of $\frac{\alpha/K + n_k}{\alpha + N}$ and $\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}}$. Therefore the number of parameters is $K(D+1) - 1$, exactly the same as with the non-Bayesian model.

3.3 Variational Approximation

An alternative approach for Bayesian inference is the variational approximation method [Bea03, Bis07]. The idea of this method is to relax the true posterior distribution to an approximated factorized distribution function and optimize the approximated function via the EM algorithm. Here, we introduce this approach for the finite Bayesian mixture model.

Recall the integral of the likelihood from Section 3.1:

$$p(\mathcal{T}|\alpha,\beta,\gamma) = \int_{\boldsymbol{\pi}}\int_{\phi}\sum_{\mathcal{Z}}\prod_{\mu=1}^{N}[p(\mathbf{X}^{\mu}|z^{\mu},\boldsymbol{\phi})p(z^{\mu}|\boldsymbol{\pi})]p(\boldsymbol{\phi}|\boldsymbol{\beta},\boldsymbol{\gamma})p(\boldsymbol{\pi}|\alpha)d\boldsymbol{\phi}d\boldsymbol{\pi} \quad (3.26)$$

The posterior distribution of the latent variable $\mathcal{Z}$ and the parameters can be calculated by:

$$p(\mathcal{Z},\boldsymbol{\pi},\boldsymbol{\phi}|\mathcal{T},\alpha,\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{p(\mathcal{T},\mathcal{Z},\boldsymbol{\pi},\boldsymbol{\phi}|\alpha,\boldsymbol{\beta},\boldsymbol{\gamma})}{p(\mathcal{T}|\alpha,\boldsymbol{\beta},\boldsymbol{\gamma})} \quad (3.27)$$

However, this distribution is intractable because we cannot solve the integral in (3.26). Notice that the latent variable $\mathcal{Z}$ and parameter $\boldsymbol{\pi}$ are coupled, which makes it impossible to handle the integral over $\boldsymbol{\pi}$ and the summing over $\mathcal{Z}$ separately. We need to move it out of the integral. To solve this problem, we introduce a decoupled approximate distribution and minimize the KL divergence between the true posterior distribution and the approximate one. Assume the approximate distribution has the following form:

$$q(\mathcal{Z},\boldsymbol{\pi},\boldsymbol{\phi}|\boldsymbol{\tau},\boldsymbol{\rho},\boldsymbol{\eta},\boldsymbol{\nu}) = \left[\prod_{\mu=1}^{N} q(z^{\mu}|\boldsymbol{\tau}^{\mu})\right] q(\boldsymbol{\phi}|\boldsymbol{\eta},\boldsymbol{\nu})q(\boldsymbol{\pi}|\boldsymbol{\rho}) \quad (3.28)$$

where:

$$q(z^{\mu}|\boldsymbol{\tau}^{\mu}) \sim \text{Multinomial}(\boldsymbol{\tau}^{\mu}) \quad (3.29)$$

$$q(\boldsymbol{\phi}|\boldsymbol{\eta},\boldsymbol{\nu}) \sim \text{Beta}(\boldsymbol{\eta},\boldsymbol{\nu}) \quad (3.30)$$

$$q(\boldsymbol{\pi}|\boldsymbol{\rho}) \sim \text{Dir}(\boldsymbol{\rho}) \quad (3.31)$$

Each variational parameter corresponds with a hyperparameter or the latent variable as shown in Equations (3.29), (3.30) and (3.31). In this approximation, we break the controlling chain of "$\alpha \rightarrow \boldsymbol{\pi} \rightarrow \mathcal{Z}$" and replace it with two chains:

$\boldsymbol{\rho} \to \boldsymbol{\pi}$ and $\boldsymbol{\tau} \to \mathcal{Z}$. Then the log likelihood is:

$$\begin{aligned}
&\log p(\mathcal{T}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) \\
&= \log \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) d\boldsymbol{\phi} d\boldsymbol{\pi} \\
&= \log \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} \frac{p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})}{q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})} q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}) d\boldsymbol{\phi} d\boldsymbol{\pi} \\
&= \log E_q \left[\frac{p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})}{q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})} \right] \\
&\leq E_q \left[\log \frac{p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})}{q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})} \right] \\
&= E_q[\log p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})] - E_q[\log q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})] \qquad (3.32)
\end{aligned}$$

In Equation (3.32) we use Jensen's inequality to move the log inside the integrals and the sum. Let $\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})$ denote the right-hand side of Equation (3.32) for simplicity. It is easy to show that the difference between the log likelihood and $\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is the KL divergence of $q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu})$ and the true posterior distribution:

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) + KL(q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu}) \parallel p(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\mathcal{T}, \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})) \\
&= E_q[\log p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})] - E_q[\log q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})] \\
&\quad + E_q[\log q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})] - E_q[\log p(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\mathcal{T}, \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})] \\
&= E_q[\log p(\mathcal{T}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})] \\
&= \log p(\mathcal{T}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) \qquad (3.33)
\end{aligned}$$

As demonstrated in Chapter 2, $\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is the lower bound of the log-likelihood because the KL divergence is always non-negative. We maximize the log-likelihood by maximizing its lower bound $\mathcal{L}$. We can further expand $\mathcal{L}$ as:

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) \\
&= E_q[\log p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})] - E_q[\log q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})] \\
&= E_q[\log p(\boldsymbol{\pi}|\alpha)] + E_q[\log p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma})] + E_q[\log p(\mathcal{Z}|\boldsymbol{\pi})] + E_q[\log p(\mathcal{T}|\mathcal{Z}, \boldsymbol{\phi})] \\
&\quad - E_q[\log q(\boldsymbol{\pi}|\boldsymbol{\rho})] - E_q[\log q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu})] - E_q[\log q(\mathcal{Z}|\boldsymbol{\tau})] \qquad (3.34)
\end{aligned}$$

The computation of these terms can be seen in Appendix A.1. In the computation we use the fact that $E[\log \pi_k] = \Psi(\alpha_k) - \Psi(\sum_{k'=1}^{K} \alpha_{k'})$ if $\boldsymbol{\pi} \sim \text{Dir}(\alpha)$ where $\Psi(\cdot)$ is the digamma function:

$$\Psi(x) = \frac{d}{dx} \ln \Gamma(x) \tag{3.35}$$

The computational result is the following.

$$\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) = C + f(\boldsymbol{\rho}, \boldsymbol{\tau}) + g(\boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{\tau}) + h(\boldsymbol{\tau}) \tag{3.36}$$

where:

$$C = \log \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} + K \sum_{i=1}^{D} \log \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \tag{3.37}$$

$$\begin{aligned} f(\boldsymbol{\rho}, \boldsymbol{\tau}) = & \sum_{k=1}^{K} (\frac{\alpha}{K} + \sum_{\mu=1}^{N} \tau_k^\mu - \rho_k)[\Psi(\rho_k) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})] \\ & - \log \frac{\Gamma(\sum_{k=1}^{K} \rho_k)}{\prod_{k=1}^{K} \Gamma(\rho_k)} \end{aligned} \tag{3.38}$$

$$\begin{aligned} g(\boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{\tau}) = & \sum_{i=1}^{D} \sum_{k=1}^{K} (\beta_i + \sum_{\mu=1}^{N} \tau^\mu x_i^\mu - \eta_{ik})[\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\ & + \sum_{i=1}^{D} \sum_{k=1}^{K} (\gamma_i + \sum_{\mu=1}^{N} \tau^\mu (1 - x_i^\mu) - \nu_{ik})[\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\ & - \sum_{i=1}^{D} \sum_{k=1}^{K} \log \frac{\Gamma(\eta_{ik} + \nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} \end{aligned} \tag{3.39}$$

$$h(\boldsymbol{\tau}) = - \sum_{\mu=1}^{N} \sum_{k=1}^{K} \tau_k^\mu \log \tau_k^\mu \tag{3.40}$$

Then we need to maximize $\mathcal{L}$ with respect to τ_k^μ, ρ_k, η_{ik} and ν_{ik} with the constraints $\sum_{k=1}^{K} \tau_k^\mu = 1, \forall \mu$. We can use traditional differential method for this maximization. $\mathcal{L}$ is maximized by:

$$\tau_k^\mu \propto \exp \Big\{ \Psi(\rho_k) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})$$

Algorithm 3.2 Variational EM for Finite Bayesian Bernoulli Mixtures

initialize ρ_k, η_{ik} and ν_{ik}
repeat
  for $\mu = 1$ to N **do**
    for $k = 1$ to K **do**
      Update τ_k^μ according to Equation (3.41)
    end for
    Normalize τ_k^μ to sum to 1
  end for
  $\rho_k = \alpha/K + \sum_{\mu=1}^{N} \tau_k^\mu$
  $\eta_{ik} = \beta_i + \sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu$
  $\nu_{ik} = \gamma_i + \sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu)$
until convergence

$$+ \sum_{i=1}^{D} \left\{ x_i^\mu [\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] + (1 - x_i^\mu)[\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \right\} \Bigg\} \tag{3.41}$$

$$\rho_k = \frac{\alpha}{K} + \sum_{\mu=1}^{N} \tau_k^\mu \tag{3.42}$$

$$\eta_{ik} = \beta_i + \sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu \tag{3.43}$$

$$\nu_{ik} = \gamma_i + \sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu) \tag{3.44}$$

The details of this maximization are shown in Appendix A.2. As in the non-Bayesian EM algorithm, this is a co-ordinate ascent process. After initialization, we perform the E-step by calculating τ_k^μs by Equation (3.41). We then perform the M-step by updating the variational parameters ρ_k, η_{ik} and ν_{ik} by Equation (3.42), (3.43) and (3.44). The E-step and the M-step are iterated until convergence. This process is shown in algorithm 3.2.

The time cost of updating τ_k^μ is $O(NDK)$. The time cost of updating variational parameters is also $O(NDK)$. Therefore the time cost for an iteration is $O(NDK)$.

The predictive inference of variational approximation is to use the $q(\cdot)$ as a substitute of the true posterior distribution. As the approximation function is de-coupled, we can analytically perform the integral. For a transaction $\hat{X}$, its

probability given by the model is the following integral:

$$\begin{aligned} p(\hat{X}|X_{1:N}) &= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\hat{z}} p(\hat{X}|\hat{z}, \boldsymbol{\phi}_k) p(\hat{z}|\boldsymbol{\pi}) q(\boldsymbol{\pi}|\boldsymbol{\rho}, X_{1:N}) q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}, X_{1:N}) d\boldsymbol{\pi} d\boldsymbol{\phi} \\ &= \sum_{k=1}^{K} \frac{\rho_k}{\sum_{k=1}^{K} \rho_{k'}} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{\hat{x}_i} \left(\frac{\nu_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{1-\hat{x}_i} \end{aligned} \tag{3.45}$$

The computation of this integral is similar to the computation in Equation (3.45) for Gibbs sampling. For any itemset I, its predictive probability given by the model is:

$$\begin{aligned} p(I|X_{1:N}) &= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\hat{z}} p(I|\hat{z}, \boldsymbol{\phi}) p(\hat{z}|\boldsymbol{\pi}) q(\boldsymbol{\pi}, \boldsymbol{\phi}|\boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu}, X_{1:N}) d\boldsymbol{\pi} d\boldsymbol{\phi} \\ &= \sum_{k=1}^{K} \frac{\rho_k}{\sum_{k=1}^{K} \rho_{k'}} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{\delta(i \in I)} \end{aligned} \tag{3.46}$$

where $\delta(i \in I)$ is the indicator that

$$\delta(i \in I) = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{if } i \notin I \end{cases} \tag{3.47}$$

The time cost for predicting an itemset is $O(K(N_I + 1))$ where N_I is the length of the itemset. As with the predictive inference of Gibbs sampling, we only need to take care of the value of ρ_k, η_{ik} and ν_{ik} proportionally. Therefore the number of parameters is $K(D + 1) - 1$ and is exactly the same as with the non-Bayesian model.

3.4 Summary

In this chapter we evolved the non-Bayesian mixture model to a finite Bayesian mixture model. To achieve this, we added several hyper-parameters and related prior distribution functions to the model. As we introduced in Chapter 2, we use the Dirichlet distribution as the prior function of the distribution of the component selection parameter $\boldsymbol{\pi}$.

In Section 3.2 we applied Gibbs sampling to do the model inference. The sampling scheme is based on the posterior distribution of the component assignment given by the Polya Urn model of the Dirichlet distribution. We also discussed

the computational time and predictive inference of Gibbs sampling.

In Section 3.3 we demonstrated the variational EM algorithm for the model inference. To solve the problem of parameter coupling, we use an factorized function to approximate the true posterior distribution. By this way, the EM algorithm can be easily applied.

The empirical result of the two proposed algorithms will be presented and discussed in Chapter 5. In next chapter we will extend the finite Bayesian mixture model to infinity. By doing this, we can break the limitation of the fixed number of components.

Chapter 4

Dirichlet Process Mixture

In this chapter we discuss the Dirichlet process mixture model for transaction datasets. As introduced in Chapter 2, the Dirichlet Process (DP) is an infinite extension of the Dirichlet distribution. In Chapter 3 we introduced the finite Bayesian mixture model with a Dirichlet distribution as the prior to the mixture proportions. If we replace the Dirichlet distribution with the DP in the mixture model, the finite Bayesian mixture model becomes the Dirichlet process mixture model.

Since the DP is discrete and infinite, the DP mixture model has the property of "self-growing". This means that the DP mixture model does not require the number of components to be assigned by the user. Instead, it finds the appropriate number automatically which is why the DP mixture model is also called the nonparametric Bayesian inference. This property is valuable because inference or clustering based on a fixed number of components always faces the problem of finding the proper "K". If the K is not large enough, the model tends either to be over simplified or to under-fit the data. Conversely, if the K is too large, it is very possible that the model will over-fit the data. However, the DP can generate new components according to the likelihood of existing components and the prior of the parameter. It improves the mixture model from both sides - through introducing a prior to mitigate over-fitting and by self-growing to mitigate under-fitting. The DP mixture model has a sound foundation in Bayesian nonparametric statistics. Hence, it is a good choice for unsupervised machine learning tasks when faced with datasets of which little prior knowledge is held.

In Chapter 2 we introduced the definition and properties of the DP, including

the Polya Urn model and the stick-breaking representation. As with the Dirichlet distribution, the Polya Urn model is the base for collapsed Gibbs sampling, proposed by [Mac94]. The stick-breaking construction, however, supplies an explicit presentation of the DP and is therefore useful for building a variational approximation of the DP. Blei and Jordan developed a truncated variational inference [BJ05] for the DP mixture model. In this chapter we discuss how the two approaches work for our transaction datasets.

We first introduce the assumption and the structure of the model. Then we introduce both approaches. The experiments and related discussions will be given in Chapter 5.

4.1 Assumption and Structure

As with the finite Bayesian mixture model, we assign priors to the multinomial and Bernoulli distributions. The difference here is that we use the DP instead of the Dirichlet distribution, causing significant changes to both the structure and the presentation of the model.

Recall the Dirichlet process introduced in Chapter 2; the DP is a distribution over the measures of the event space. In our case, the DP is a measure of the multi-variant Bernoulli distributions. In Chapter 2, we used a bag of balls with continuum colours as an example. In the mixture model, the balls are replaced by the mixtures. Each time we draw from the Dirichlet process, it is possible that the result is an existing Bernoulli distribution. It is also possible that we draw a new distribution from the base distribution of $Beta(\boldsymbol{\beta}, \boldsymbol{\gamma})$. In the finite Bayesian mixture, we can express the generation of the component and Bernoulli parameters separately as the number of components is finite and fixed. In the DP, it is more convenient to say that for a certain transaction we directly draw a Bernoulli measure from the DP. The new model can be expressed as follows:

1. Assign $\alpha, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as the hyperparameters of the model, where α, is positive scalar and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_D)$ and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_D)$ where $\beta_i > 0, \gamma_i > 0$ for all $i = 1, \ldots, D$.

2. For each transaction $\mathbf{X}^\mu$,choose the Bernoulli measure from the Dirichlet process.

$$\boldsymbol{\phi} \sim \mathrm{DP}(\alpha, B_0(\boldsymbol{\beta}, \boldsymbol{\gamma})) \tag{4.1}$$

3. For each transaction $\mathbf{X}$:

$$p(X|\boldsymbol{\phi}) = \prod_{i=1}^{D} \phi^{x_i}(1-\phi_i)^{1-x_i} \tag{4.2}$$

This process can be briefly written as:

$$\boldsymbol{\phi}^{\mu}|\alpha, B_0 \sim \mathrm{DP}(\alpha, B_0(\boldsymbol{\beta}, \boldsymbol{\gamma})) \tag{4.3}$$

$$X^{\mu}|\boldsymbol{\phi}^{\mu} \sim \mathrm{Ber}(\boldsymbol{\phi}^{\mu}) \tag{4.4}$$

Since the number of components is not fixed, it is hard to explicitly write the joint probability or the likelihood under this expression of the DP. A more explicit form of the DP via the stick-breaking construction and related variational approximate inference will be introduced in Section 4.3.

4.2 Inference via Gibbs Sampling

As with the collapsed Gibbs sampler we applied in the finite Bayesian inference, we also want to sample the component indicators in the DP mixture. The difference is that the number of components may grow itself.

Again, we start from the posterior distribution by the Bayes rule:

$$p(z^{\mu} = k|\mathcal{Z}_{-\mu}, \mathcal{T}) \propto \underbrace{p(X^{\mu}|z^{\mu}=k)}_{\text{likelihood}} \underbrace{p(z^{\mu}=k|\mathcal{Z}_{-\mu})}_{\text{prior}} \tag{4.5}$$

where $k = 1, \ldots, K', K'+1$ and K' is the current number of components. $k = K'+1$ means the number of components increases by 1. By the Polya urn model introduced in Chapter 2, the μ-th observation's distribution given the rest observations is Equation (2.68).

$$p(z^{\mu}|\mathcal{Z}_{-\mu}) = \begin{cases} \frac{n_{k/\{\mu\}}}{\alpha+N-1}, & \text{if } z^{\mu}=k, k \leq K' \\ \frac{\alpha}{\alpha+N-1}, & \text{if } z^{\mu}=K'+1 \end{cases} \tag{4.6}$$

Another way of understanding this prior is thinking of the DP as the limit of the Dirichlet distribution when we extend the finite event space to infinity. As discussed in Chapter 2, Equation (2.69), the Dirichlet distribution can be written

as the following:

$$p(z^\mu|\mathcal{Z}_{-\mu}) = \begin{cases} \frac{n_k}{\alpha+N-1}, & \text{if } z^\mu = K, k \leq K \\ \frac{\alpha}{\alpha+N-1}, & \text{if } z^\mu \sim Multinomial(g_1, \ldots, g_K) \end{cases} \tag{4.7}$$

where $g(\cdot)$ is the base measure. Although the DP is infinite, at a certain stage it contains finite components. Thus, when we extend the base measure to a continuous distribution, the Equation (4.7) changes to Equation (4.6) naturally.

The inference on the likelihood for the existed components $p(X^\mu|z^\mu = k), k \leq K'$ is the same as we did in Chapter 3. We do not repeat the computation but simply list the result:

$$p(X^\mu|z^\mu = k) = \frac{1}{C'} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{x_i^\mu} \left(\frac{\nu_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{1-x_i^\mu} \tag{4.8}$$

where C' is the normalization constant and:

$$\eta_{ik} = \beta_i + \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'} \tag{4.9}$$

$$\nu_{ik} = \gamma_i + n_{k/\{\mu\}} - \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'} \tag{4.10}$$

The rest part is the likelihood when $k = K' + 1$. We still have the following integral:

$$p(X^\mu|z^\mu = K'+1) = \int_{\boldsymbol{\phi}_k} p(X^\mu|\boldsymbol{\phi}_{K'+1}) p(\boldsymbol{\phi}_{K'+1}|\mathcal{Z}_{-\mu}, \mathcal{T}_{-\mu}) d\boldsymbol{\phi} \tag{4.11}$$

Notice that $p(\boldsymbol{\phi}_{K'+1}|\mathcal{Z}_{-\mu}, \mathcal{T}_{-\mu})$ is in fact the prior distribution of $\boldsymbol{\phi}$ since it is a new component drawn from the base distribution. Therefore we have:

$$\begin{aligned} p(X^\mu|z^\mu = K'+1) &= \int_{\boldsymbol{\phi}_k} p(X^\mu|\boldsymbol{\phi}_{K'+1}) p(\boldsymbol{\phi}_{K'+1}|\boldsymbol{\beta}, \boldsymbol{\gamma}) d\boldsymbol{\phi} \\ &= \int_{\boldsymbol{\phi}_{K'+1}} \prod_{i=1}^{D} \phi^{x_i^\mu} (1-\phi)^{1-x_i^\mu} \cdot \frac{1}{C} \prod_{i=1}^{D} \frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \phi^{\beta_i - 1} (1-\phi)^{\gamma_i - 1} d\boldsymbol{\phi} \\ &= \frac{1}{C} \prod_{i=1}^{D} \frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \int_0^1 \phi^{x_i^\mu + \beta_i - 1} (1-\phi)^{1-x_i^\mu + \gamma_i - 1} d\phi \end{aligned}$$

Algorithm 4.1 collapsed Gibbs sampling for Dirichlet process mixture model

initialize α, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$
repeat
  for $\mu = 1$ to N **do**
    Update η_{ik}, ν_{ik} by
      $\eta_{ik} = \beta_i + \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'}$
      $\nu_{ik} = \gamma_i + n_{k/\{\mu\}} - \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'}$
    Calculate multinomial probabilities based on
    $$p(z^\mu = k | \mathcal{Z}_{-\mu}, \mathcal{T}) \propto \begin{cases} \frac{n_{k/\{\mu\}}}{\alpha+N-1} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{x_i^\mu} \left(\frac{\nu_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{1-x_i^\mu}, & \text{if } z^\mu = K', k \leq K' \\ \frac{\alpha}{\alpha+N-1} \prod_{i=1}^{D} \left(\frac{\beta_i}{\beta_i+\gamma_i}\right)^{x_i^\mu} \left(\frac{\gamma_i}{\beta_i+\gamma_i}\right)^{1-x_i^\mu}, & \text{if } z^\mu = K'+1 \end{cases}$$
    Normalize $p(z^\mu = k)$
    Sample z^μ based on $p(z^\mu = k)$
  end for
until convergence

$$= \frac{1}{C} \prod_{i=1}^{D} \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \frac{\Gamma(x_i^\mu + \beta_i)\Gamma(1 - x_i^\mu + \gamma_i)}{\Gamma(\beta_i + \gamma_i + 1)}$$
$$= \frac{1}{C'} \prod_{i=1}^{D} \left(\frac{\beta_i}{\beta_i + \gamma_i}\right)^{x_i^\mu} \left(\frac{\gamma_i}{\beta_i + \gamma_i}\right)^{1-x_i^\mu} \tag{4.12}$$

Combining the Equations (4.6), (4.8) and (4.12), we can obtain the sampling formula for the DP mixture model:

$$p(z^\mu = k | \mathcal{Z}_{-\mu}, \mathcal{T}) \propto \begin{cases} \frac{n_{k/\{\mu\}}}{\alpha+N-1} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{x_i^\mu} \left(\frac{\nu_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{1-x_i^\mu}, & \text{if } z^\mu = K', k \leq K' \\ \frac{\alpha}{\alpha+N-1} \prod_{i=1}^{D} \left(\frac{\beta_i}{\beta_i+\gamma_i}\right)^{x_i^\mu} \left(\frac{\gamma_i}{\beta_i+\gamma_i}\right)^{1-x_i^\mu}, & \text{if } z^\mu = K'+1 \end{cases} \tag{4.13}$$

where $n_{k/\{\mu\}}$ is the number of data points assigned to component k except X^μ and:

$$\eta_{ik} = \beta_i + \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'} \tag{4.14}$$

$$\nu_{ik} = \gamma_i + n_{k/\{\mu\}} - \sum_{\mu' \neq \mu}^{N} \delta(z^{\mu'} - k) x_i^{\mu'} \tag{4.15}$$

The whole process of the collapsed Gibbs sampling for the finite Bayesian mixture model is shown in Algorithm 4.1.

The difference between the inference of the finite Bayesian mixture and the DP mixture is that the scale of the DP mixture model varies in sampling. Generally, the model only grows, but it is possible that some components vanish during sampling because all their data points move to other components. The model's changes in scale add extra time cost to the algorithm. The time cost of adding or deleting a component is $O(D)$. The varying number of components also makes the estimation of the time cost of the algorithm difficult. At a certain stage, when the number of components is K', the time cost of updating parameters and sampling is $O(K'D)$ and $O((K'+1)D)$ respectively.

The predictive inference of the DP mixture model is similar to the finite Bayesian mixture model since the outcome of Algorithm 4.1 is always finite. The slight difference is that we should take the base distribution into account, so the joint probability, given transaction X, and the marginal probability, given itemset I, are as follows:

$$\begin{aligned} p(\hat{X}|X_{1:N}) = &\sum_{k=1}^{K} \frac{n_k}{\alpha+N} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{\hat{x}_i} \left(\frac{\nu_{ik}}{\eta_{ik}+\nu_{ik}}\right)^{1-\hat{x}_i} \\ &+ \frac{\alpha}{\alpha+N} \prod_{i=1}^{D} \left(\frac{\beta_i}{\beta_i+\gamma_i}\right)^{x_i^{\mu}} \left(\frac{\gamma_i}{\beta_i+\gamma_i}\right)^{1-x_i^{\mu}} \end{aligned} \tag{4.16}$$

$$\begin{aligned} p(I|X_{1:N}) = &\sum_{k=1}^{K} \frac{n_k}{\alpha+N} \prod_{i=1}^{D} \left(\frac{\eta_{mk}}{\eta_{mk}+\nu_{mk}}\right)^{\delta(i\in I)} \\ &+ \frac{\alpha}{\alpha+N} \prod_{i=1}^{D} \left(\frac{\beta_i}{\beta_i+\gamma_i}\right)^{\delta(i\in I)} \end{aligned} \tag{4.17}$$

where n_k is the number of points assigned to component k, K is the final number of components contained in the model, and:

$$\eta_{ik} = \beta_i + \sum_{\mu=1}^{N} \delta(z^{\mu}-k) x_i^{\mu} \tag{4.18}$$

$$\nu_{ik} = \gamma_i + n_k - \sum_{\mu=1}^{N} \delta(z^{\mu}-k) x_i^{\mu} \tag{4.19}$$

The time cost of predicting an itemset is $O(K(N_I+1))$ where N_I is the length of the itemset. The number of parameters is $K(D+1)-1$. Here K is not assigned by user but automatically generated by the algorithm.

4.3 Truncated Variational Approximation

As well as sampling methods, variational methods are also important for the inference of the mixture model as they are generally faster. The stick-breaking presentation of the DP provides a possible way to apply variational methods. In the stick-breaking representation, an unknown random distribution is represented as a sum of countably infinite atomic distributions. In the transaction dataset background, the target distribution is the distribution of the transaction $p(\mathbf{X}^\mu)$ and the atomic distributions are the conditional distributions, such as $p(\mathbf{X}^\mu|z^\mu)$.

According to the stick-breaking construction, the DP mixture model for the transaction dataset can be expressed as follows. First we sample an infinite series of i.i.d. real numbers $v_1, \ldots, v_k, \ldots$ where $v_k \sim Beta(1, \alpha)$. Each time we sample a v_k, we sample a multi-variant Bernoulli distribution with parameters $\boldsymbol{\phi}_k$ from the base distribution $Beta(\boldsymbol{\beta}, \boldsymbol{\gamma})$. Then the DP mixture model is the infinite weighted sum of the Bernoulli distributions where the weight $\pi_k = v_k \prod_{l=1}^{k-1}(1 - v_l)$:

$$DP(\alpha, Beta(\boldsymbol{\beta}, \boldsymbol{\gamma})) = \sum_{k=1}^{\infty} \pi_k Ber(\boldsymbol{\phi}_k) \tag{4.20}$$

Based on the stick-breaking representation, the Dirichlet process mixture model is as follows.

1. Assign $\alpha, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as the hyperparameters of the model, where α, is positive scalar and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_D)$ and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_D)$ where $\beta_i > 0, \gamma_i > 0$ for all $i = 1, \ldots, D$.

2. Choose $v_k \sim \text{Beta}(1, \alpha), k = 1, \ldots$

3. For each item and component choose $\phi_{ik} \sim$Beta(β_i, γ_i) where:

$$p(\phi_{ik}|\beta_i, \gamma_i) = \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \phi_{ik}^{\beta_i - 1}(1 - \phi_{ik})^{\gamma_i - 1} \tag{4.21}$$

4. For each transaction $\mathbf{X}^\mu$:

   (a) Choose a component $z^\mu \sim$Multinomial$(\boldsymbol{\pi}(\mathbf{v}))$ where:

$$\pi_k(\mathbf{v}) = v_k \prod_{l=1}^{k-1}(1 - v_l) \tag{4.22}$$

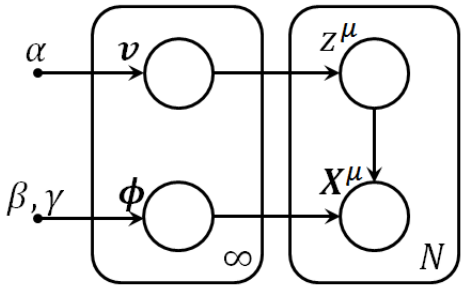


Figure 4.1: Graphic representation of DP mixture in stick-breaking representation

(b) Then we can generate data by:

$$p(\mathbf{X}^\mu|z^\mu,\boldsymbol{\phi}) = \prod_{i=1}^{D} \phi_{iz^\mu}^{x_i^\mu}(1-\phi_{iz^\mu})^{1-x_i^\mu} \tag{4.23}$$

The stick-breaking construction for the DP mixture is depicted in Figure 4.1. With the model assumption, the joint probability of the dataset $\mathcal{T}$, component indicators $\mathcal{Z}$ and the model parameters $\mathbf{v}$ and $\boldsymbol{\phi}$ is:

$$\begin{aligned} &p(\mathcal{T},\mathcal{Z},\mathbf{v},\boldsymbol{\phi}|\alpha,\boldsymbol{\beta},\boldsymbol{\gamma}) \\ &\quad= \prod_{\mu=1}^{N}[p(\mathbf{X}^\mu|z^\mu,\boldsymbol{\phi})p(z^\mu|\mathbf{v})]p(\boldsymbol{\phi}|\boldsymbol{\beta},\boldsymbol{\gamma})p(\mathbf{v}|\alpha) \end{aligned} \tag{4.24}$$

Integrating over $\boldsymbol{\pi}$, $\boldsymbol{\phi}$, summing over $\mathcal{Z}$ and applying a logarithm, we obtain the log-likelihood of the dataset:

$$\ln p(\mathcal{T}|\alpha,\boldsymbol{\beta},\boldsymbol{\gamma}) = \ln \int_{\mathbf{v}}\int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} p(\mathcal{T},\mathcal{Z},\mathbf{v},\boldsymbol{\phi}|\alpha,\boldsymbol{\beta},\boldsymbol{\gamma}) d\boldsymbol{\phi} d\mathbf{v} \tag{4.25}$$

Here, the integral over $\mathbf{v}$ means the integral over a vector $\mathbf{v} \in [0,1]^\infty$. The integral over $\boldsymbol{\phi}$ means the integral over a $\infty \times D$ vector $\boldsymbol{\phi} \in [0,1]^{\infty\times D}$. The summing over $\mathcal{Z}$ is the summing over all possible $\mathcal{Z}$ configurations. This integral is intractable because of the integral over infinity dimensions and the coupling of $\mathcal{Z}$ and $\mathbf{v}$.

Notice the following limit with a given truncation K:

$$\lim_{T\to\infty}[1-\sum_{k=1}^{K}\pi_k(\mathbf{v})] = \lim_{T\to\infty}\prod_{k=1}^{K}(1-v_k) = 0 \tag{4.26}$$

Equation (4.26) shows that for a large enough truncation level T, all the components beyond the Tth component can be ignored as the sum of their proportion is very close to 0. This means that it is possible to approximate the infinite situation by a finite number of components. Where this differs with the finite Bayesian model is that in the finite Bayesian mixture, the number of components is finite. In the truncated DP mixture, whilst the number of components is infinite, we only use a finite distribution to approximate it. Therefore, we can use a finite and fully decoupled function as the approximation of true posterior distribution. We propose the following factorized family of variational distribution:

$$\begin{aligned} &q(\mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\boldsymbol{\tau}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu}) \\ &\quad = \left[\prod_{\mu=1}^{N} q(z^\mu|\boldsymbol{\tau}^\mu)\right] \left[\prod_{k=1}^{K}\prod_{i=1}^{D} q(\phi_{ik}|\eta_{ik}, \nu_{ik})\right] \prod_{k=1}^{K-1} q(v_k|\rho_{1k}, \rho_{2k}) \end{aligned} \tag{4.27}$$

where

$$\begin{aligned} &q(z^\mu|\boldsymbol{\tau}^\mu) \sim \text{Multinomial}(\boldsymbol{\tau}^\mu) \\ &q(\phi_{ik}|\eta_{ik}, \nu_{ik}) \sim \text{Beta}(\eta_{ik}, \nu_{ik}) \\ &q(v_k|\rho_{1k}, \rho_{2k}) \sim \text{Beta}(\rho_{1k}, \rho_{2k}) \end{aligned}$$

Here $\boldsymbol{\rho}_1$, $\boldsymbol{\rho}_2$, $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ are free variational parameters corresponding to the hyperparameters 1, α, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, and $\boldsymbol{\tau}$ is the multinomial parameter for decoupling $\mathbf{v}$ and $\mathcal{Z}$. As we are assuming the proportion of the components beyond T is 0, the value of v_K in the approximation is always 1. We use this $q(\cdot)$ function to approximate the true posterior distribution of the parameters. To achieve this, we need to estimate the values of $\boldsymbol{\rho}_1$, $\boldsymbol{\rho}_2$, $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$. Similar to the non-Bayesian mixture EM, we expand the log-likelihood and optimize its lower bound.

$$\begin{aligned} &\ln p(\mathcal{T}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) \\ =&\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) + KL(q(\mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\boldsymbol{\tau}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu}) \parallel p(\mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\mathcal{T}, \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})) \\ =&E_q\left[\ln \frac{p(\mathcal{T}, \mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})}{q(\mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\boldsymbol{\tau}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu})}\right] - E_q\left[\ln \frac{p(\mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\mathcal{T}, \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})}{q(\mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\boldsymbol{\tau}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu})}\right] \end{aligned} \tag{4.28}$$

With Equations (4.24) and (4.27), the lower bound of the log-likelihood can be

further expanded to:

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) \\
=&E_q[\log p(\mathcal{T}, \mathcal{Z}, \mathbf{v}, \boldsymbol{\phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})] - E_q[\log q(\mathcal{Z}, \mathbf{v}, \boldsymbol{\phi})] \\
=&E_q[\log p(\mathbf{v}|\alpha)]+E_q[\log p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma})]+E_q[\log p(\mathcal{Z}|\mathbf{v})]+E_q[\log p(\mathcal{T}|\mathcal{Z}, \boldsymbol{\phi})] \\
&-E_q[\log q(\mathbf{v}|\boldsymbol{\rho}_1, \boldsymbol{\rho}_2)]-E_q[\log q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu})]-E_q[\log q(\mathcal{Z}|\boldsymbol{\tau})] \qquad (4.29)
\end{aligned}$$

The calculation of the terms is similar to the calculation performed in the finite Bayesian mixture except $E_q[\log p(\mathcal{Z}|\mathbf{v})]$. This term can be expanded as follows.

$$\begin{aligned}
E_q[\log p(\mathcal{Z}|\mathbf{v})] &= E_q[\ln \prod_{\mu=1}^{N} p(z^\mu|\mathbf{v})] \\
&= \sum_{\mu=1}^{N} E_q[\ln p(z^\mu|\mathbf{v})] \\
&= \sum_{\mu=1}^{N} E_q\left[\ln v_{z^\mu} \prod_{t=1}^{z^\mu-1}(1-v_t)\right] \\
&= \sum_{\mu=1}^{N}\left[E_q(\ln v_{z^\mu}) + E_q\left(\sum_{t=1}^{z^\mu-1}\ln(1-v_t)\right)\right] \qquad (4.30)
\end{aligned}$$

where

$$\begin{aligned}
E_q[\ln v_{z^\mu}] &= \sum_{z^\mu=1}^{K} \tau_{z^\mu} \int_{v_{z^\mu}} \ln v_{z^\mu} p(v_{z^\mu}|\rho_{1z^\mu}, \rho_{2z^\mu}) dv_{z^\mu} \\
&= \sum_{k=1}^{K} [\tau_k^\mu(\Psi(\rho_{1k}) - \Psi(\rho_{1k}+\rho_{2k}))] \\
&= \sum_{k=1}^{K-1} [\tau_k^\mu(\Psi(\rho_{1k}) - \Psi(\rho_{1k}+\rho_{2k}))] \qquad (4.31) \\
E_q\left[\left(\sum_{t=1}^{z^\mu-1}\ln(1-v_t)\right)\right] &= \sum_{z^\mu=1}^{K} \tau_{z^\mu} \int_{\mathbf{v}} \sum_{l=1}^{z^\mu-1} \ln(1-v_l) p(v_t|\rho_{1l}, \rho_{2l}) d\mathbf{v} \\
&= \sum_{z^\mu=1}^{K} \tau_{z^\mu} \sum_{l=1}^{z^\mu-1} (\Psi(\rho_{2l}) - \Psi(\rho_{1l}+\rho_{2l})) \\
&= \sum_{k=1}^{K} \tau_k^\mu \sum_{l=1}^{k-1} (\Psi(\rho_{2l}) - \Psi(\rho_{1l}+\rho_{2l}))
\end{aligned}$$

$$= \sum_{k=1}^{K-1} \sum_{t=k+1}^{K} \tau_t^{\mu} (\Psi(\rho_{2k}) - \Psi(\rho_{1k} + \rho_{2k})) \tag{4.32}$$

The above expectation should be a summation of infinite terms. However, the truncation makes them finite. Since we set $v_K = 1$, the corresponding variational parameters ρ_{1K} and ρ_{2K} can also be fixed as $\rho_{1K} = 1$ and $\rho_{2K} = 0$. Thus $\Psi(\rho_{1K}) - \Psi(\rho_{1K} + \rho_{2K}) = 0$ and we need only to perform the summation for the first $K - 1$ terms in Equation (4.32). The rest of the calculation can be seen in Appendix A.3. The $\mathcal{L}$ can be finally expanded as:

$$\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) = C + f(\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\tau}) + g(\boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{\tau}) + h(\boldsymbol{\tau}) \tag{4.33}$$

where

$$C = (K-1) \ln \alpha + K \sum_{i=1}^{D} \ln \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \tag{4.34}$$

$$\begin{aligned} f(\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\tau}) = & \sum_{k-1}^{K-1} \left(\sum_{\mu=1}^{N} \tau_k^{\mu} - \rho_{1k} + 1 \right) [\Psi(\rho_{1k}) - \Psi(\rho_{1k} + \rho_{2k})] \\ & + \sum_{k=1}^{K-1} \left(\alpha + \sum_{\mu=1}^{N} \sum_{t=k+1}^{K} \tau_t^{\mu} - \rho_{2k} \right) [\Psi(\rho_{2k}) - \Psi(\rho_{1k} + \rho_{2k})] \\ & - \sum_{k=1}^{K-1} \ln \frac{\Gamma(\rho_{1k} + \rho_{2k})}{\Gamma(\rho_{1k})\Gamma(\rho_{2k})} \end{aligned} \tag{4.35}$$

$$\begin{aligned} g(\boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{\tau}) = & \sum_{i=1}^{D} \sum_{k=1}^{K} \left(\beta_i + \sum_{\mu=1}^{N} \tau_k^{\mu} x_i^{\mu} - \eta_{ik} \right) [\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\ & + \sum_{i=1}^{D} \sum_{k=1}^{K} \left[\gamma_i + \sum_{\mu=1}^{N} \tau_k^{\mu} (1 - x_i^{\mu}) - \nu_{ik} \right] [\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\ & - \sum_{i=1}^{D} \sum_{k=1}^{K} \ln \frac{\Gamma(\eta_{ik} + \nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} \end{aligned} \tag{4.36}$$

$$h(\boldsymbol{\tau}) = - \sum_{\mu=1}^{N} \sum_{k=1}^{K} \tau_k^{\mu} \ln \tau_k^{\mu} \tag{4.37}$$

We maximize Equation (4.33) with respect to τ_k^{μ}, ρ_{1k}, ρ_{1k}, η_{ik} and ν_{ik}. This

yields:

$$\rho_{1k} = 1 + \sum_{\mu=1}^{N} \tau_k^\mu \tag{4.38}$$

$$\rho_{2k} = \alpha + \sum_{\mu=1}^{N} \sum_{t=k+1}^{T} \tau_t^\mu \tag{4.39}$$

$$\eta_{ik} = \beta + \sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu \tag{4.40}$$

$$\nu_{ik} = \gamma + \sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu) \tag{4.41}$$

$$\begin{aligned} \tau_k^\mu \propto \exp \Bigg\{ & \Psi(\rho_{1k}) + \sum_{k'=1}^{k-1} \Psi(\rho_{2k'}) - \sum_{k'=1}^{k} \Psi(\rho_{1k'} + \rho_{2k'}) \\ & + \sum_{i=1}^{D} x_i^\mu [\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\ & + \sum_{i=1}^{D} (1 - x_i^\mu) [\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \Bigg\} \end{aligned} \tag{4.42}$$

The details of the maximization are given in Appendix A.4. As with the variational EM for the finite Bayesian mixture model, after initialization we first perform the E-step by calculating τ_k^μs by Equation (4.42). Then we perform the M-step by updating the variational parameters ρ_{1k}, ρ_{2k}, η_{ik} and ν_{ik} by Equation (4.38), (4.39), (4.40) and (4.41). The E-step and the M-step are iterated until convergence. This process is shown in Algorithm 4.2. The time cost of updating τ_k^μ is $O(NDK)$ as the term $\sum_{k'=1}^{k-1} \Psi(\rho_{2k'}) - \sum_{k'=1}^{k-1} \Psi(\rho_{1k'} + \rho_{2k'})$ and can be calculated accumulatively. The time cost of updating variational parameters is also $O(NDK)$. Therefore the time cost for an iteration is $O(NDK)$.

The predictive inference of this truncated variational approximation also uses the truncated variational posterior approximation $q(\cdot)$ as a substitute for the true posterior distribution of the model parameters. As the $q(\cdot)$ function is factorized, we can analytically solve the probability integral. The joint probability and the marginal probability given the model are as follows:

$$p(\hat{X}|X_{1:N}) = \int_{\mathbf{v}} \int_{\boldsymbol{\phi}} \sum_{\hat{z}} p(\hat{X}|\hat{z}, \boldsymbol{\phi}_k) p(\hat{z}|\mathbf{v}) q(\mathbf{v}, \boldsymbol{\phi}|\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu}, X_{1:N}) d\mathbf{v} d\boldsymbol{\phi}$$

Algorithm 4.2 Variational EM for the DP Bernoulli mixture model

initialize ρ_{1k}, ρ_{2k}, η_{ik} and ν_{ik}
repeat
  for $\mu = 1$ to N **do**
    for $k = 1$ to K **do**
      Update τ_k^μ according to (4.42)
    end for
    Normalize τ_k^μ to sum to 1
  end for
  $\rho_{1k} = 1 + \sum_{\mu=1}^{N} \tau_k^\mu$
  $\rho_{2k} = \alpha + \sum_{\mu=1}^{N} \sum_{k'=k+1}^{T} \tau_{k'}^\mu$
  $\eta_{ik} = \beta + \sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu$
  $\nu_{ik} = \gamma + \sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu)$
until convergence

$$= \sum_{k=1}^{K} \frac{\rho_{1k}}{\rho_{1k} + \rho_{2k}} \prod_{k'=1}^{k-1} \frac{\rho_{2k'}}{\rho_{1k'} + \rho_{2k'}} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{\hat{x}_i} \left(\frac{\nu_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{1-\hat{x}_i} \tag{4.43}$$

$$p(I|X_{1:N}) = \int_{\mathbf{v}} \int_{\boldsymbol{\phi}} \sum_{\hat{z}} p(I|\hat{z}, \boldsymbol{\phi}) p(\hat{z}|\mathbf{v}) q(\mathbf{v}, \boldsymbol{\phi}|\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\eta}, \boldsymbol{\nu}, X_{1:N}) d\mathbf{v} d\boldsymbol{\phi}$$
$$= \sum_{k=1}^{K} \frac{\rho_{1k}}{\rho_{1k} + \rho_{2k}} \prod_{k'=1}^{k-1} \frac{\rho_{2k'}}{\rho_{1k'} + \rho_{2k'}} \prod_{i=1}^{D} \left(\frac{\eta_{ik}}{\eta_{ik} + \nu_{ik}} \right)^{\delta(i \in I)} \tag{4.44}$$

where $\delta(i \in I)$ is the indicator that

$$\delta(i \in I) = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{if } i \notin I \end{cases} \tag{4.45}$$

In Equation (4.44), we need only to use the values of $\frac{\rho_{1k}}{\rho_{1k}+\rho_{2k}} \prod_{k'=1}^{k-1} \frac{\rho_{2k'}}{\rho_{1k'}+\rho_{2k'}}$ as the proportion of each component. Thus, these values need calculating only once. The time cost of predicting an itemset is still $O(K(N_I + 1))$ where N_I is the length of the itemset. The number of parameters is still $K(D + 1) - 1$.

4.4 Summary

In this chapter we introduced the Dirichlet Process mixture model by applying the Dirichlet Process prior instead of the Dirichlet distribution. By the DP, new

component can be drawn from a base distribution to represent transactions which are not well represented by existing components.

Similar to Chapter 3, in Section 4.2 and Section 4.3 we introduced two algorithms to do the inference of the DP mixture model. Gibbs sampling is based on the posterior distribution given by the Polya Urn model of the DP. The general scheme is similar to the finite one, but the main difference is that there is always a chance that some new components are generated in the sampling process. This difference makes that the model can grow automatically and the user does not need to assign the number of component before learning.

The variational algorithm for the DP mixture model is based on the stick-breaking representation of the DP. By the stick-breaking representation, the DP can be written as a sum of countable infinite atomic distributions. The limit of the partial sum of the series is 1, we can use a truncation to approximate the infinite sum. With this truncation approximation, we proposed the variational EM algorithm for the DP mixture model.

The empirical result of the two algorithms we proposed in this chapter we be presented and discussed in next chapter. Together we will compare them with the algorithms we introduced in Chapter 3 and the non-Bayesian mixture model.

Chapter 5

Experiments and Discussion

In this chapter we evaluate the four proposed algorithms and compare them to the non-Bayesian mixture model from a frequent itemset mining perspective. The four algorithms are variational EM for finite Bayesian mixture model (VFBM), Gibbs sampling for finite Bayesian mixture model (GSFBM), variational EM for the DP mixture model (VDPM), and Gibbs sampling for the DP mixture model (GSDPM). First we report some implementation issues of the experiments including the convergence condition and the framework used for frequent itemset prediction of all the algorithms. Then we introduce the evaluation criteria, the benchmark dataset used in the experiments and other experimental settings. In the third part we give the results. The analysis and discussion of the experiments will be given in section 5.4.

5.1 Implementation Issues

The four algorithms, including Gibbs sampling and the variational EM algorithm for both the finite Bayesian mixture model and the DP mixture model, are all implemented in Matlab. Each Matlab program outputs a model file recording the model parameters including the mixing proportions and the conditional probabilities. We then implemented a Java program to read the model file, search for all the frequent itemsets based on the model and compare the result with the Eclat data mining result (ground truth). We also implemented the EM algorithm for the non-Bayesian mixture model for comparison.

In the implementation of the algorithms, we diagnose convergence by measuring the log-likelihood of the training datasets. After each iteration, we compute

the relative log-likelihood by:

$$\bar{L} = \frac{\ln L(Model|Data)}{ND \ln 2} \tag{5.1}$$

where $ND \ln 2$ is the worst case in which the model provides no information about the data. $\bar{L}$ is always between -1 and 0. A larger $\bar{L}$ implies a better fitting. Although the log-likelihood cannot be a meaningful criterion for model evaluation, it is useful for diagnosing convergence.

In our experiments, we use the following conditions based on $\bar{L}$ to diagnose convergence.

1. If the number of iterations reaches 150, we force the algorithm to stop as it may take too long time.

2. We compute the difference of $\bar{L}$ between two successive iterations:

$$\Delta\bar{L}_i = \bar{L}_{i+1} - \bar{L}_i \tag{5.2}$$

   If the following conditions are continuously satisfied for 7 iterations, we think the algorithm has converged:

   (a) $\Delta\bar{L}_i < 0.001$

   (b) $\Delta\bar{L}_i < 0.01 \times \max(\Delta\bar{L}_1, \ldots, \Delta\bar{L}_{i-1})$

Another issue is the framework for frequent itemset prediction. In our experiments, we evaluate the models by comparing the quality of the set of frequent itemsets predicted. Therefore we need a framework similar to data mining algorithms in order to organize the searching process. We select the Eclat algorithm as the searching scheme of our prediction as it is straightforward for coding and more efficient than Apriori. The only modification is that we do not need to maintain a "*tid-list*" for each itemset. We can directly calculate the probability of a given itemset based on the model.

5.2 Evaluation Criteria and Experiment Settings

We use the following three evaluation criteria for model comparison.

1. We measure the difference between the predicted set of frequent itemsets and the true set of frequent itemsets by calculating the false negative rate (F^-) and the false positive rate (F^+). They are calculated by:

$$F^- = \frac{N_M}{N_M + N_C}, F^+ = \frac{N_F}{N_F + N_C}$$

where N_M is the number of itemsets that the model failed to predict, N_F is the number of itemsets that the model falsely predicted and N_C is the number of itemsets that the model predicted correctly.

2. For any true frequent itemset I, we calculate the relative error by:

$$e(I) = \frac{|p_M(I) - f(I)|}{f(I)},$$

where $p_M(I)$ is the probability predicted by the model. The overall quality of the estimation $\hat{E}$ is:

$$\hat{E} = \frac{1}{N_I} \sum_{j=1}^{N_I} e(I_j), \tag{5.3}$$

where N_I is the total number of true frequent itemsets.

3. To test whether the model is under-estimating or over-estimating, we define the empirical mean of relative difference for a given set S as:

$$\hat{D}_S = \frac{1}{|S|} \sum_{j=1}^{|S|} \frac{p_M(I_j) - f(I_j)}{f(I_j)} \tag{5.4}$$

4. Computation time is also an important aspect of the performance. Similar to the measurement of time cost in other probability models [PMS03], we also call the training time cost as the offline time ($\mathcal{T}_{Off}$) and call the frequent itemsets generation time as the online time ($\mathcal{T}_{On}$). We also record the time cost of Eclat data mining as a comparison.

We evaluate the models on nine benchmark datasets and five synthetic datasets. The synthetic datasets are sampled from the mixture models with 15, 25, 50, 75 and 140 components respectively. The aim of involving synthetic datasets is to validate the inference algorithms. We want to evaluate the difference between the new model and the original model.

The nine benchmark datasets used are Adult, Accidents, Chess, Connect4, LetRecog, Anonymous Microsoft Web data, Mushroom, Nursery and PenDigits. They are all benchmark datasets used for frequent itemset mining research. The datasets Adult, Chess, Connect4, LetRecog, Anonymous Microsoft Web data, Mushroom, Nursery and PenDigits are originally from the UCI Machine Learning repository [FA10]. Among the nine datasets, Accidents and Anonymous Microsoft Web data are typical transaction datasets and the rest are discretised and transformed into the form of transaction datasets. Connect4, Chess and Mushroom are transformed by Roberto Bayardo and the transformed versions are available at `http://fimi.ua.ac.be/data/`. Adult, LetRecog, Nursery and RenDigits have been transformed by Frans Coenen[Coe03] and are available at `http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html`. The dataset Accidents [GWBV03] is a record of traffic accidents and is also available at `http://fimi.ua.ac.be/data/`.

The parameter settings of the experiments are as follows. For the non-Bayesian, finite Bayesian and the truncated DP mixture models, we used 15, 25, 50 and 75 components respectively. For the DP mixture model via Gibbs sampling, we do not need to choose the number of components. For each parameter configuration, we repeat five times to reduce the variance. The hyper-parameters are set as follows: for finite mixture, we set α/K equals 1.5 and for the DP mixture, we set α equals 1.5. The value of β_i equals the frequency of the items in the whole dataset and γ_i equals $1 - \beta_i$.

The last parameter for the experiments is the minimum frequency threshold. In our experiments, the requirement of the threshold is that we need the itemsets to be frequent enough to represent the correlation within the datasets while generating a large enough number of frequent itemsets for model comparison as well. The thresholds for all datasets and the characteristics of the data mining results including the number of frequent itemsets with different lengths will be shown in each dataset results section.

5.3 Experiment Results

5.3.1 A Brief Summary of the Test Results

To make this part more clear and easy to read, we briefly summarize the result and some observations before showing the exact tables and figures.

The tests on synthetic datasets show that the variational approaches and the sampling approaches can find the right parameters of the mixture model with the average relative error less than 2%, which means that if the model is right, the Bayesian methods can recover the model with a small accuracy loss. The Bayesian methods also give a better result than the non-Bayesian method. Comparing the variational approaches and the Gibbs sampling approaches, the sampling methods give more accurate results.

From the test results of the benchmark datasets, we observe that the accuracies of the four Bayesian approaches are all better than non-Bayesian approach, as the empirical relative errors of the Bayesian models are all smaller for all datasets tested. From the false negative rate and false positive rate's view, for most datasets the mixture models have a high false negative rate and a low false positive rate, which means that the models have lost some of the frequent itemsets but the itemsets generated from the model have a relatively high creditability. The Bayesian methods have a better false negative rate while the non-Bayesian model has a better false positive rate.

We also observe that all the mixture models have a estimation bias of underestimation. We demonstrate this by plotting the relative difference rate of the frequent itemsets with different lengths for each model. In most cases the frequencies of longer itemsets are more underestimated by the model. This analysis also explains why our models have a high false negative rate but a low false negative rate. As the models have a bias of underestimation, the false negative rate caused by underestimation is thus higher than the false positive rate caused by overestimation. The Bayesian methods are less biased than the non-Bayesian method. Thus their false negative rates are lower and the non-Bayesian method has a lower false positive rate.

To demonstrate the difference between the $\hat{E}$ measurement and the F^-, F^+ measurement, we also analyzed the distribution of the frequent itemsets by their frequencies. We observe that if a large proportion of the frequent itemsets are distributed near the frequency threshold, the F^- is then very likely to be much

higher than $\hat{E}$. Reversely, if there are only a small proportion of frequent itemsets located at this "sensitive area", the F^- is very likely to be small. Two typical examples are the datasets Connect4 and Mushroom. The Bayesian models give very accurate estimations on the dataset Connect4 with the relative error less then 1%, however the false negative rates are as high as nearly 20%. That is because nearly half of the frequent itemsets' frequencies are very close to the threshold. Therefore a small amount of underestimation will cause a large proportion of true frequent itemsets labeled as infrequent by the model. The situation of Mushroom is the opposite. Only a small amount of frequent itemsets are located near the threshold. This distribution leads low false negative rates while the relative errors of the model are relatively higher.

The overall accuracies of the sampling approaches are better and more stable than the variational methods, though the differences are not significant. The DP mixture model via Gibbs sampling can automatically find the appropriate "K". It gives comparable result to the finite model when it uses less than 75 components and it over-performs other algorithms when it chooses the "K" larger than 75. Here 75 is the maximal number of components in our tests for finite models and the truncated DP mixture model.

We also compare the time cost of the models. The Bayesian models take more time than the non-Bayesian model for training. Among the Bayesian models, the variational methods are faster than the two sampling methods. The DP mixture model via Gibbs sampling is the slowest, however the benefit of this approach is that it does not require multiple runs to find the proper K.

With the model prepared, the process of generating frequent itemsets by the model is much faster than the Eclat data mining. In most cases the model generation process is over 10 times faster than Eclat mining. As the model is irrelevant to the scale of the original dataset and the minimum frequency threshold. The probability models can save more time when we deal with large datasets or we need to mine the dataset multiple times with different thresholds.

A more detailed summary and discussion about the experimental results are given in Section 5.4.

5.3.2 Synthetic Datasets

To validate the variational EM and Gibbs sampling methods, we generate five synthetic datasets from five mixture models with 15, 25, 50, 75 and 140 components respectively and apply the four methods to the synthetic datasets to see how closely the new models compare with the original mixture model. The criteria of model comparison are the same as proposed in Section 5.2. We compare the frequent itemsets generated from the original model with those from the new model to assess the difference.

The synthetic datasets are generated as follows. First we train some models with different number of components from the dataset Accident. Then we implement a data generator that samples data according to the models trained. We sample 10,000 transactions for each dataset and use the generated data to train the new models. We set the minimum frequency threshold at 30%, also as used in the Accidents. As the aim is to validate the inference methods, we assume that our model is already correct. That is, we set the K of the algorithms to be exactly the same as the number of components of the models used to generate data except the DP mixture via Gibbs sampling because this algorithm finds K itself.

The test results of the synthetic datasets are shown in Table 5.1 where F^- is the *False Negative Rate* in percentage, F^+ is the *False Positive Rate* in percentage and the $\hat{E}$ is the *Empirical Relative Error* in percentage. We also calculate the standard variances of these values. In the table, *NBM*, *VFBM*, *GSFBM*, *VDPM* and *GSDPM* are short for non-Bayesian mixture, finite Bayesian mixture via variational EM, finite Bayesian mixture via Gibbs sampler, DP mixture via variational EM and DP mixture model via Gibbs sampler respectively. For the number of components of the DP mixture via Gibbs sampler, we use the mean of the number of components used in five trials.

From Table 5.1 we can observe that the average empirical errors of all four Bayesian methods are below 2%, which means these methods can recover the original model with a relatively small loss of accuracy. Comparing all the methods, GSFBM fits the original model best. Non-Bayesian model gives the worst overall estimation but the best false positive rate. The results of the other approaches are generally comparable. With regards to specific datasets, in the tests on Syn-15 and Syn-25, VDPM is slightly better than GSDPM, and GSDPM is slightly better than VFBM. In the tests of Syn-50, the results of GSDPM and VDPM are

	Criteria		F^-	F^+	$\hat{E}$
Syn15	NBM	K=15	9.55±0.63	**1.40±0.24**	2.12±0.12
	VFBM	K=15	4.11±0.77	2.38±0.49	1.25±0.13
	GSFBM	K=15	**3.19±0.18**	2.93±0.04	**1.17±0.02**
	VDPM	K=15	3.54±0.25	2.69±0.48	1.18±0.05
	GSDPM	K=12.6	3.84±0.41	2.71±0.37	1.24±0.03
Syn25	NBM	K=25	9.50±0.58	**1.24±0.21**	1.94±0.10
	VFBM	K=25	3.73±1.57	2.35±0.37	1.60±0.20
	GSFBM	K=25	**2.63±0.74**	2.84±0.28	**0.95±0.07**
	VDPM	K=25	3.46±0.70	2.48±0.48	1.06±0.13
	GSDPM	K=19	3.63±1.13	2.71±0.64	1.11±0.11
Syn50	NBM	K=50	10.29±0.55	**0.93±0.09**	2.03±0.10
	VFBM	K=50	5.46±0.65	1.26±0.16	1.19±0.12
	GSFBM	K=50	**3.16±0.32**	1.60±0.11	**0.85±0.03**
	VDPM	K=50	5.14±0.57	1.23±0.20	1.13±0.07
	GSDPM	K=31	5.20±1.07	1.21±0.17	1.13±0.16
Syn75	NBM	K=75	9.59±0.22	0.71±0.12	1.79±0.07
	VFBM	K=75	5.92±0.86	0.70±0.09	1.14±0.14
	GSFBM	K=75	**4.34±0.60**	0.81±0.07	**0.89±0.09**
	VDPM	K=75	6.04±0.54	**0.67±0.08**	1.14±0.08
	GSDPM	K=49.6	5.76±0.57	0.91±0.11	1.14±0.08
Syn140	NBM	K=140	11.59±0.32	**0.68±0.06**	2.31±0.06
	VFBM	K=140	8.49±0.54	1.03±0.07	1.76±0.12
	GSFBM	K=140	**5.43±0.30**	1.27±0.16	**1.22±0.01**
	VDPM	K=140	8.66±0.40	1.14±0.21	1.80±0.04
	GSDPM	K=65	6.66±0.26	1.45±0.13	1.47±0.04

Table 5.1: Test result of synthetic datasets (%), average of 5 runs

very close and both are slightly better than VDPM. In the tests on Syn-75, the three methods give similar results. In the tests on Syn-140, GSDPM outperforms the other two approaches whilst the rest are close.

Notice that the components used by GSDPM are becoming relatively fewer as the number of components in the original model increases. This phenomenon might be caused by that GSDPM merged some of the small components of the original model. Generally, the result of Gibbs sampling is more accurate than the variational methods. This result can be easily explained as the quality of the result of the EM algorithm relies on the quality of initialization more than MCMC methods. If the models are correct, the MCMC sampling can normally give a distribution very close to true distribution with enough iteration. In our situation,

as the synthetic datasets are generated from mixture models, our model is always correct. When the situation of initialization is becoming more complicated such as the adding of more components, the probability of reaching a good initialization decreases. Therefore, the average performances of the variational methods become worse than GSDPM which uses far fewer components. If we regard the performance of the GSFBM as a standard, the difference in performance of the variational methods with this standard can be seen as the cost of local minimum. The difference in performance of the GSDPM with this standard can be viewed as the cost of fewer components.

Another issue is that although the empirical relative errors of the four approaches are only about 1%-2%, the F^- is much higher relatively. This can be explained by the distribution of the frequent itemsets over frequency. Figure 5.1 is the distribution of frequent itemsets with different frequencies of the synthetic dataset Syn-15. We use this dataset as an example to demonstrate the sensitivity of estimation error on "edge" itemsets. The rest of the datasets have similar distributions. From this figure, we can see that there are over 35,000 itemsets in the range of 0.30-0.32, which means about one third of the frequent itemsets are on the "edge" of the set of frequent itemsets. Assuming that a model under-estimates each itemset by about 7%, an itemset with a frequency of 32% will be estimated as $32\% \times (1 - 7\%) = 29.76\%$, which is below the minimum frequent threshold and the itemset will be labelled as infrequent. This 7% under-estimation will eventually cause a false negative rate of over 30%. This example is an extreme circumstance. However, it is clear that with a pyramid like distribution of the frequent itemsets, when we use the F^- and F^+ criteria, the actual estimation error will be amplified.

To test whether the models are under-estimating or over-estimating the original model, we calculate the empirical errors of the frequent itemsets in a more detailed way. Firstly we classify the frequent itemsets into different categories by length. Then we calculate the means of relative difference between all categories. We plot the result in Figure 5.2. We also plot the distribution of the frequent itemsets in length for better understanding of Figure 5.2a.

From Figure 5.2a we can observe that the lengths of most frequent itemsets range from 4 to 9. The quality of the estimation of this part is most important for the overall quality of the estimation. It is quite clear that the GSFBM curve is the closest to the x-axis, which means that the trend of under-estimation is at

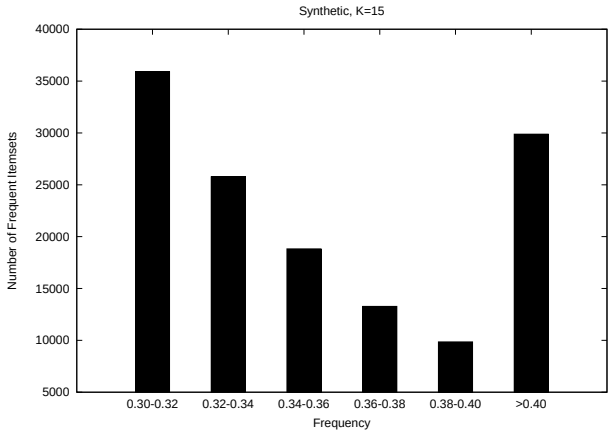
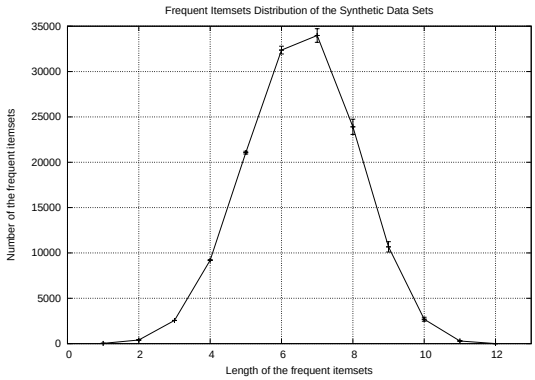
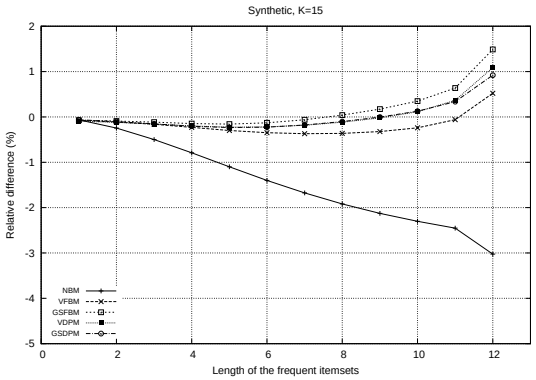


Figure 5.1: Distribution of the frequent itemsets over frequency of dataset Syn-15

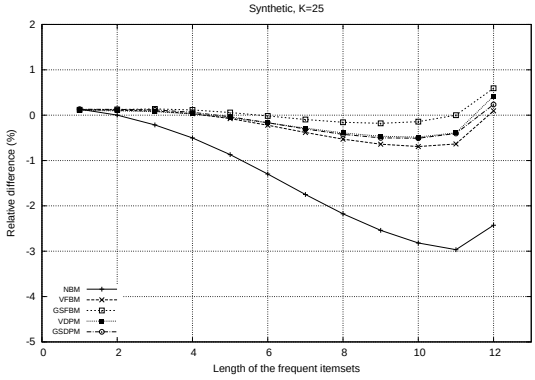
its smallest in GSFBM. In spite of this, we observe that the degrees of underestimation of all five methods increase with growth in the number of components. When the model gets more complicated, finding the right model or a better approximation becomes more difficult.



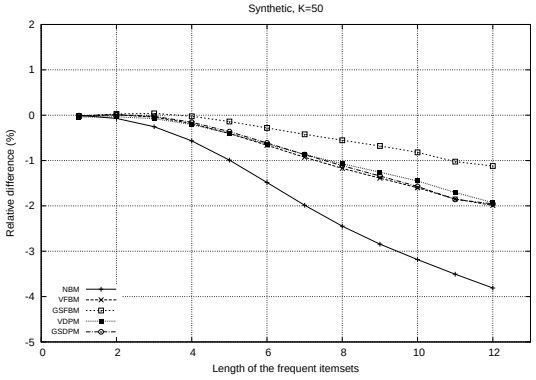
(a) Distribution of frequent itemsets over length



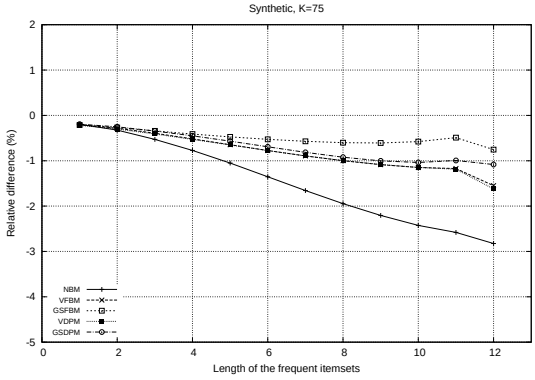
(b) Estimation trend of dataset Syn15



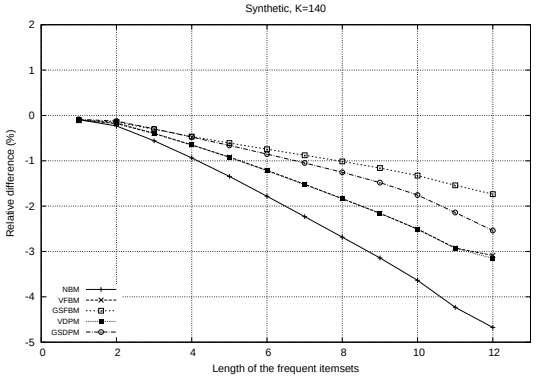
(c) Estimation trend of dataset Syn25



(d) Estimation trend of dataset Syn50



(e) Estimation trend of dataset Syn75



(f) Estimation trend of dataset Syn140

Figure 5.2: Estimation trend analysis of the synthetic data

The aim of introducing synthetic datasets is to validate the optimization ability of the five approaches. We want to check whether the algorithms for mixture models can find the right parameters of the wanted model with correct Ks. The

results show that the losses of the five approaches are acceptable with a not-so-large number of components. When the model gets more complicated, the loss caused by the algorithm tends to increase.

5.3.3 Real Datasets

For each real dataset, we report the basic information of the dataset including its scale, density and background and then list the results table and figures, as with Table 5.1 and Figure 5.2. We also record the time costs of training and predicting with the models.

Accident

This dataset is obtained from the National Institute of Statistics (NIS) for the region of Flanders, Belgium for the period 1991-2000. More specifically, the data are obtained from the Belgian Analysis Form for Traffic Accidents, a copy of which is filled out by a police officer for each traffic accident that occurs with injured or deadly wounded casualties on a public road in Belgium. In total, 340,184 traffic accident records are included in the dataset [GWBV03].

Some basic information is provided in Table 5.2. Here, the term "Density" is the number of "1"s in the dataset divided by the product of the number of rows and the number of items.

Name	Transactions	Items	Density	f_{min}
Accidents	340184	468	7.22%	30%

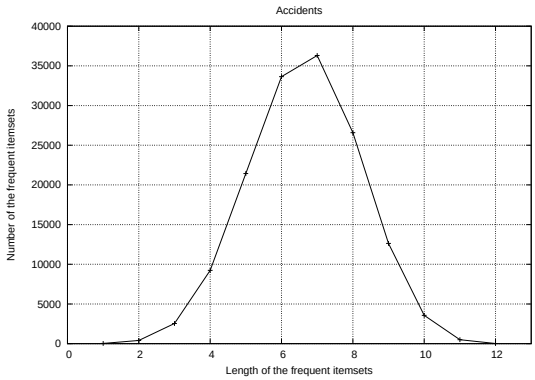
Table 5.2: Basic information of dataset Accidents

The distributions of the frequent itemsets over lengths and frequencies are shown in Figure 5.3a and 5.3b. The aim in plotting these two figures is to understand the relationship of F^-, F^+ and $\hat{E}$, and the degree of under-estimation. To illustrate the sensitivity of $\hat{E}$, we categorize the frequent itemsets into seven categories by their frequencies. These categories are 100%-102.5%, 102.5%-105%, 105%-107.5%, 107.5%-110%, 110%-115%, 115%-120%, >120% of the f_{min}. The percentage on the bar is the proportion of the category in the whole set of frequent itemsets. If the average estimation error is about 2% and there are a great proportion of frequent itemsets in the first category, then the F^- may be much greater than 2%.

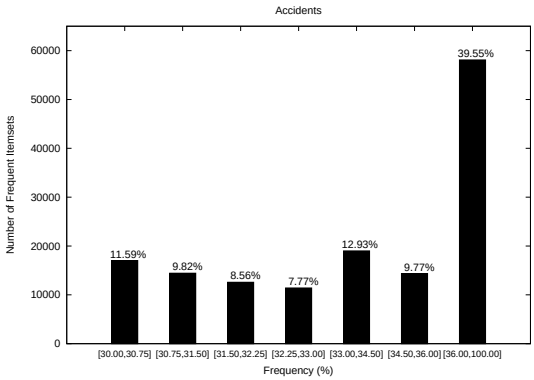
Since the original dataset is too large for training, we sample a training dataset randomly with a sample rate of 1/20. The rest are used for testing. Table 5.3 and Figure 5.3 provide the experiment results and the estimation trend of the models. The estimation trend of GSDPM is merged into Figure 5.3f for easy comparison.

Name		Accidents		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	21.13±1.53	3.07±0.25	5.05±0.40
	K=25	20.60±1.03	2.90±0.34	4.90±0.24
	K=50	17.80±0.52	2.76±0.17	4.12±0.10
	K=75	13.84±0.61	**2.68±0.21**	3.25±0.13
VFBM	K=15	13.61±1.33	5.04±0.22	3.69±0.38
	K=25	11.78±1.22	4.44±0.27	3.13±0.23
	K=50	10.16±0.60	4.07±0.30	2.71±0.13
	K=75	9.84±1.07	3.61±0.26	2.63±0.18
GSFBM	K=15	13.62±0.91	4.19±0.40	3.58±0.27
	K=25	10.85±0.31	4.12±0.31	2.93±0.11
	K=50	8.52±0.91	3.78±0.29	2.41±0.14
	K=75	8.36±0.49	3.83±0.21	2.37±0.10
VDPM	K=15	14.10±0.47	4.82±0.22	3.73±0.17
	K=25	11.00±0.58	4.51±0.27	2.95±0.15
	K=50	10.64±0.46	3.44±0.23	2.73±0.08
	K=75	10.06±0.72	4.00±0.19	2.69±0.13
GSDPM	K=114.6	**6.80±0.25**	3.76±0.17	**2.04±0.05**

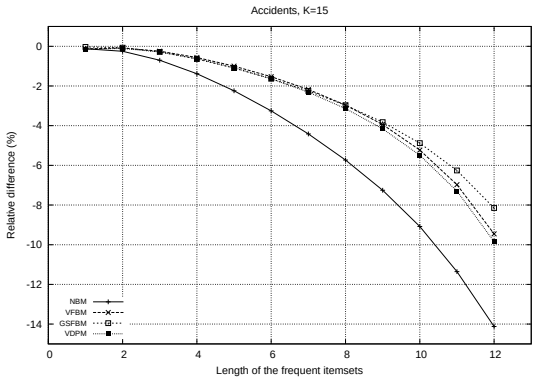
Table 5.3: Test result of dataset Accidents (%), average of 5 runs



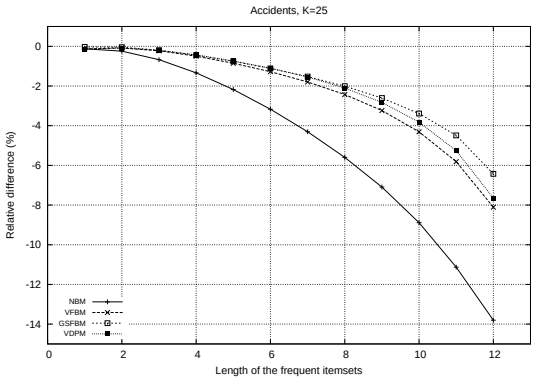
(a) Distribution on length



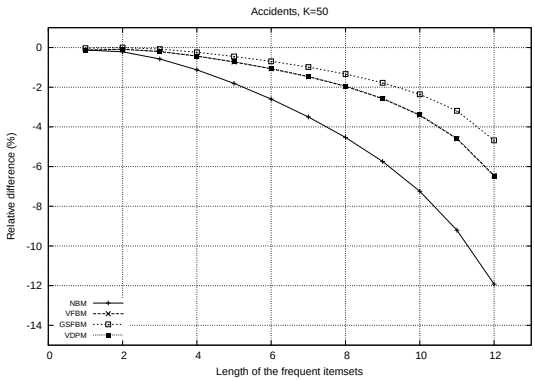
(b) Distribution on frequency



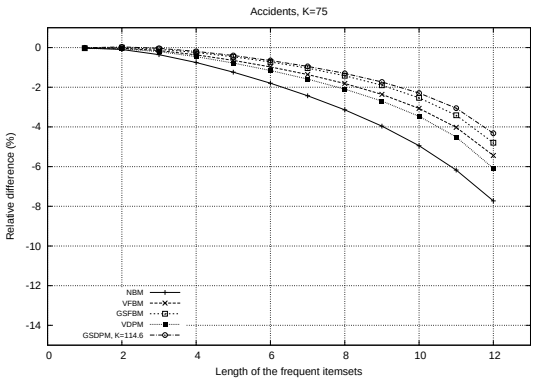
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.3: FI distribution and estimation trend of dataset Accidents

There are 11.59% of the frequent itemsets with frequencies from 100% to 102.5% of the threshold. As all the models under-estimate the frequencies, an estimation error of around 2.5% may lead to a larger false negative rate. If we use A>B to denote that the estimation of algorithm A is more precise than algorithm

B, the comparison result of the five algorithms is GSDPM > GSFBM > VDPM ≈ VFBM > NBM.

From Figure 5.3, we can observe that all the models under-estimate the frequencies. The degree of under-estimation is increased with an increase in length of the itemsets and decreased by an increase in the number of components of the model. The GSDPM uses 114.6 components on average, suggesting 75 components is not enough to describe this dataset. Therefore, both the experiment result and the under-estimation status of GSDPM is the best of the five methods.

The time cost of the training and predicting of the model is shown in Table 5.4 and 5.5. We also calculate the training time cost per iteration as T_{Off}/I.

Name		Accidents		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	16.4	22.0	1.34
	K=25	14.8	33.4	2.26
	K=50	15.0	70.8	4.70
	K=75	14.0	100.6	7.02
VFBM	K=15	18.4	185.0	10.06
	K=25	18.2	187.4	12.86
	K=50	18.0	195.4	10.86
	K=75	16.4	190.4	11.57
GSFBM	K=15	10.0	261.8	26.18
	K=25	10.0	276.2	27.62
	K=50	10.2	344.4	33.75
	K=75	10.2	408.0	40.00
VDPM	K=15	21.0	211.2	10.06
	K=25	16.2	187.2	12.86
	K=50	17.0	184.6	10.86
	K=75	16.4	189.8	11.57
GSDPM	K=114.6	130.2	6812.2	51.64

Table 5.4: Total training time cost and training time per iteration of dataset Accidents (sec), average of 5 runs

The training time costs per iteration of VFBM and VDPM do not increase when the K increases. The reason might be the optimized vector computation in

Matlab, which makes the increasing of K less sensitive. The NBM is implemented in Java. The GSFBM and GSDPM involve sampling a multinomial distribution which cannot be handled as a vector operation in Matlab. Therefore, their training time cost per iteration is still relevant to K. Generally, NBM is the fastest, variational methods are a bit slower and sampling methods are the slowest. Although the training time cost of all methods is $O(NDK)$, NBM does not involve any complex function evaluation. Variational methods need to calculate some functions such as logarithm, exponential and digamma function. The sampling methods need to calculate logarithm and exponential functions and to generate random numbers. A more time consuming aspect to sampling is that it needs to update the parameters after each draw. However, we notice that the iteration used by GSFBM is less than that of variational methods. This is because the model of GSFBM is updated after each draw. It can be viewed as an online updating model. On the other hand, variational methods are both updated in the batch mode which normally takes more iterations to converge. The prediction time cost is simpler in comparison with training cost. If the numbers of components of the different models are the same, the prediction time cost should be the same. So, we calculate the average prediction time of all the models in Table 5.5.

	T_{On}
Eclat	76.31
K=15	0.37
K=25	0.46
K=50	0.66
K=75	0.79
K=114.6	0.94

Table 5.5: FI generation time of dataset Accidents (sec), average of 5 runs

From Table 5.5, we can see that the online mining time costs of mixture models are much less than Eclat and the time costs of the mixture models are approximately proportional to the number of components.

Adult

This data was originally extracted from the census bureau database found at `http://www.census.gov/ftp/pub/DES/www/welcome.html`. It was once used

for the prediction of the annual income of a person. The basic information of this dataset is shown in Table 5.6.

Name	Transactions	Items	Density	f_{min}
Adult	48842	97	15.82%	2%

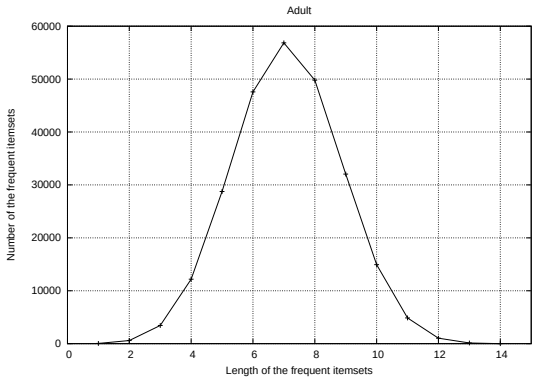
Table 5.6: Basic information of dataset Adult

We set the minimum frequency threshold as 2%. The distribution of the frequent itemsets is shown in Figure 5.4a and 5.4b.

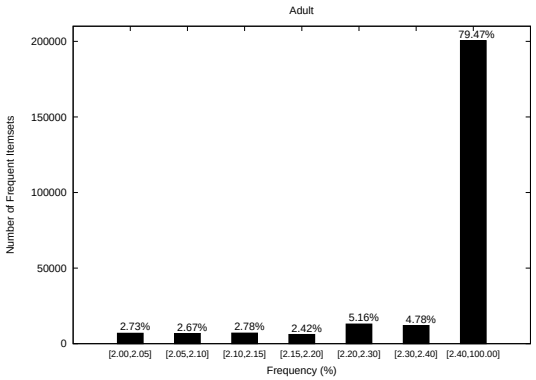
For all the real datasets other than Accidents, we sample half of the dataset as the training set and use the remainder as the testing set. The test result of this dataset is shown in Table 5.7.

Name		Adult		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	14.62±0.51	19.6±0.49	11.79±0.46
	K=25	13.04±0.33	17.74±0.48	10.41±0.33
	K=50	10.93±0.49	15.7±0.53	8.79±0.31
	K=75	10.11±0.47	14.91±0.6	8.23±0.13
VFBM	K=15	8.52±0.3	15.4±1.14	7.61±0.25
	K=25	6.99±0.27	11.75±0.39	6.63±0.27
	K=50	6.08±0.57	9.41±0.8	5.91±0.43
	K=75	5.45±0.37	7.08±0.63	5.33±0.26
GSFBM	K=15	8.45±0.95	16.84±0.88	7.96±0.79
	K=25	7.06±0.36	12.62±0.88	6.83±0.32
	K=50	5.61±0.59	8.67±1.59	5.63±0.37
	K=75	**5.08±0.29**	**6.95±0.9**	**5.11±0.23**
VDPM	K=15	8.01±0.41	14.98±0.71	7.45±0.31
	K=25	7.29±0.25	12.57±0.88	6.73±0.17
	K=50	6.25±0.3	8.75±0.86	5.99±0.35
	K=75	5.63±0.41	7.76±0.98	5.43±0.36
GSDPM	K=48.8	6.76±0.48	11.47±0.66	6.09±0.33

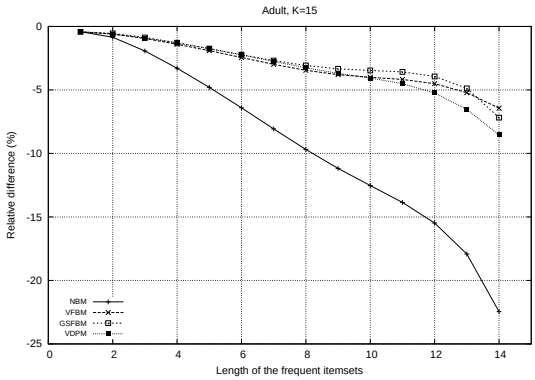
Table 5.7: Test result of dataset Adult (%), average of 5 runs



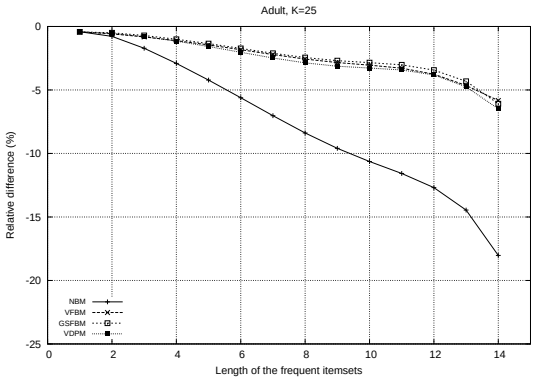
(a) Distribution on length



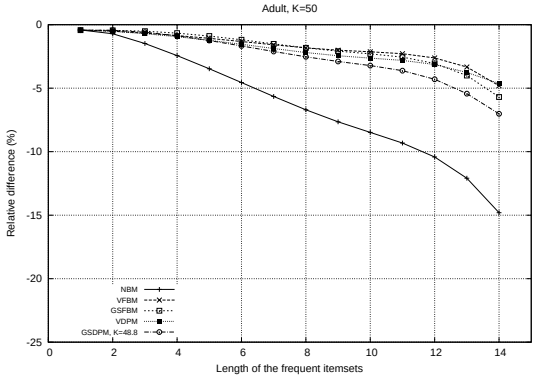
(b) Distribution on frequency



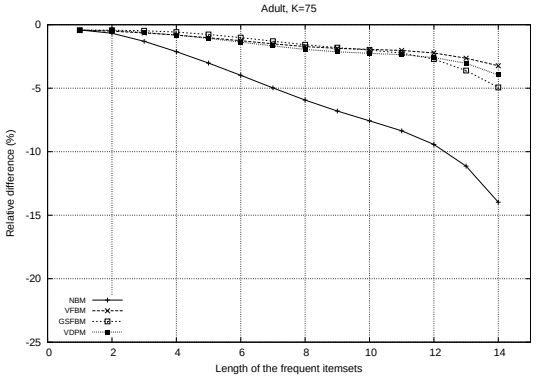
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.4: FI distribution and estimation trend of dataset Adult

The average estimation errors of GSFBM, VDPM and VFBM are about 5%-6%. The performance order is GSFBM > VDPM ≈ VFBM > GSDPM > NBM. Notice that GSDPM only uses 48.8 components on average. The result of GSDPM is comparable with the results of result algorithms with 50 components.

Their results are close. Adding 25 components to 50 gives an improvement of approximately 0.5%. As the $\hat{E}$ is about 5%-6%, the itemsets in the first two categories in Figure 5.4b are all very likely to be labelled as infrequent.

The under-estimation on this dataset is not very serious. The average under-estimation of most frequent itemsets is about 2%-3%. The NBM has an under-estimation of nearly 10% with 75 components. The time costs are shown in Table 5.8 and 5.9. The situation is quite similar to the previous dataset. The online time costs of mixture models are much less than Eclat because the prediction time costs of the models are irrelevant to the scale of the original dataset.

Name		Adult		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	22.40	29.03	1.29
	K=25	22.20	47.24	2.13
	K=50	22.40	93.35	4.18
	K=75	21.20	128.39	6.06
VFBM	K=15	27.00	977.20	36.20
	K=25	27.40	995.80	36.37
	K=50	26.60	975.00	36.66
	K=75	24.80	917.60	37.00
GSFBM	K=15	12.80	988.00	77.18
	K=25	13.80	1111.40	80.57
	K=50	14.20	1222.80	86.11
	K=75	14.60	1362.40	93.47
VDPM	K=15	27.60	997.60	36.14
	K=25	25.40	927.60	36.53
	K=50	26.20	960.60	36.66
	K=75	24.20	896.60	37.05
GSDPM	K=48.8	80.40	7151.40	88.59

Table 5.8: Total training time cost and training time per iteration of dataset Adult (sec), average of 5 runs

	T_{On}
Eclat	563.70
K=15	0.58
K=25	0.62
K=50	0.85
K=75	1.04
K=48.8	0.87

Table 5.9: FI generation time of dataset Adult (sec), average of 5 runs

Chess

This dataset is a collection of chess end-game board descriptions. Table 5.10 shows the basic information for this dataset.

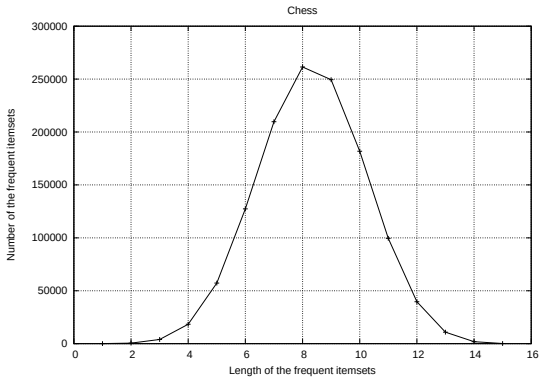
Name	Transactions	Items	Density	f_{min}
Chess	3197	75	49.32%	50%

Table 5.10: Basic information of dataset Chess

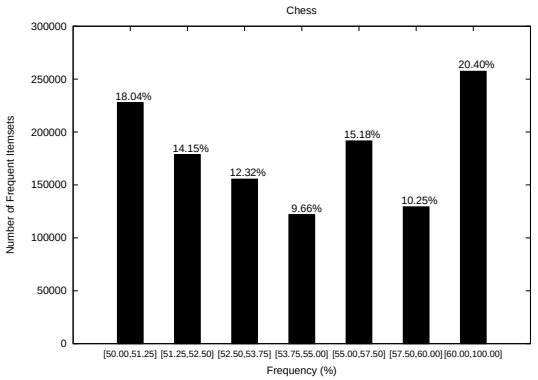
As this dataset is quite dense, the threshold is set at 50%. The dataset contains only 3197 transactions, but the number of frequent itemsets is over 1 million. The test result of Chess is in Table 5.11. The false positive and false negative rates look high, but the average estimation error is only around 3%. This is because a large proportion of frequent itemsets' frequencies are just a little higher than the threshold.

Name		Chess		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	22.47±1.24	6.68±0.76	3.89±0.17
	K=25	19.71±1.30	7.03±0.28	3.47±0.19
	K=50	17.22±0.98	7.66±0.77	3.12±0.08
	K=75	15.01±1.05	8.62±0.58	2.89±0.12
VFBM	K=15	15.10±0.75	9.60±1.17	3.03±0.10
	K=25	13.49±0.62	9.55±0.27	**2.77±0.08**
	K=50	14.20±0.62	9.20±0.91	2.83±0.02
	K=75	13.95±0.91	9.4±0.46	2.81±0.09
GSFBM	K=15	18.85±1.94	7.03±0.76	3.39±0.28
	K=25	15.63±1.82	8.25±1.05	2.97±0.17
	K=50	14.86±0.55	8.03±0.42	2.83±0.06
	K=75	17.4±0.6	**6.6±0.39**	3.07±0.06
VDPM	K=15	15.92±1.67	9.56±1.87	3.17±0.13
	K=25	14.31±0.85	8.81±0.84	2.83±0.05
	K=50	**13.49±0.73**	9.66±0.34	2.78±0.10
	K=75	14.43±1.08	8.25±0.77	2.77±0.11
GSDPM	K=29.6	14.46±0.72	8.40±0.89	2.83±0.05

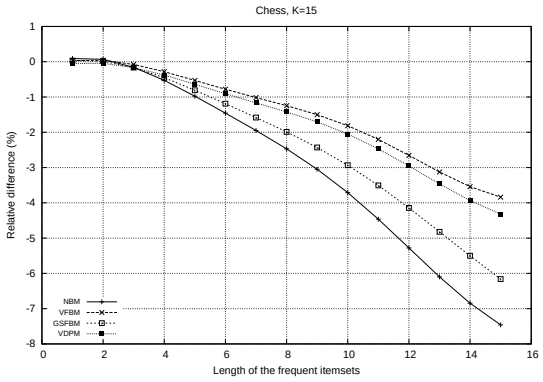
Table 5.11: Test result of dataset Chess (%), average of 5 runs



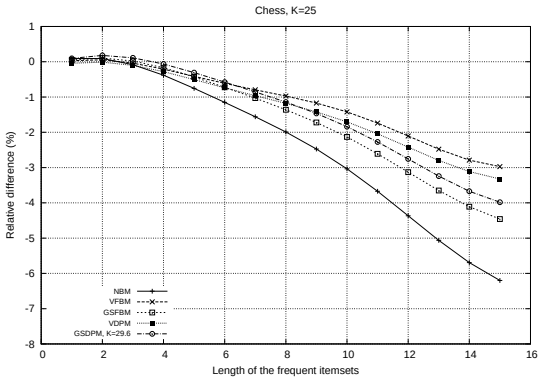
(a) Distribution on length



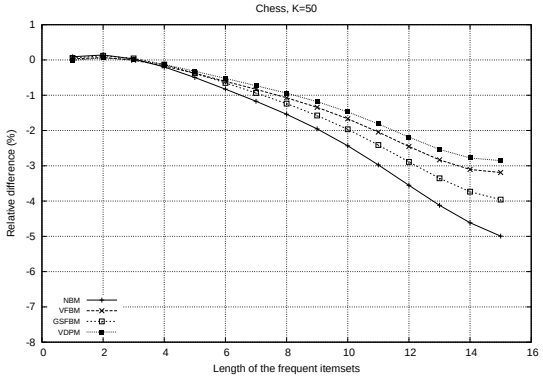
(b) Distribution on frequency



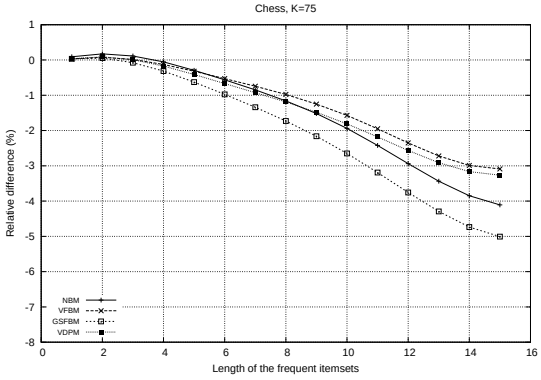
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.5: FI distribution and estimation trend of dataset Chess

The GSDPM suggests around 30 components. We see that the differences between all five algorithms are not too considerable. Variational methods are slightly better than sampling methods. The under-estimation degrees are about 1%-2% on average, as shown in Figure 5.5.

As the dataset is small but dense, the training process does not take too much time compared with the time cost of Eclat mining.

Name		Chess		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	17.20	1.38	0.08
	K=25	17.60	2.26	0.13
	K=50	15.80	3.88	0.24
	K=75	15.80	6.40	0.41
VFBM	K=15	19.00	8.00	0.42
	K=25	19.80	8.46	0.43
	K=50	16.80	7.58	0.45
	K=75	17.00	8.08	0.48
GSFBM	K=15	14.40	15.80	1.09
	K=25	14.40	17.80	1.23
	K=50	14.20	23.60	1.66
	K=75	14.00	27.20	1.94
VDPM	K=15	18.40	7.74	0.42
	K=25	18.40	7.88	0.43
	K=50	16.60	7.50	0.45
	K=75	16.40	7.80	0.48
GSDPM	K=29.6	36.20	47.20	1.30

Table 5.12: Total training time cost and training time per iteration of dataset Chess (sec), average of 5 runs

	T_{On}
Eclat	47.36
K=15	2.77
K=25	3.62
K=50	4.81
K=75	6.20
K=48.8	4.10

Table 5.13: FI generation time of dataset Chess (sec), average of 5 runs

Connect4

This dataset is a collection of board descriptions of the game of Connect-4. Its density looks normal, but it generates a very large number of high frequency itemsets. We raise the threshold to 85% to make sure the memory of our test machine is enough for the experiment.

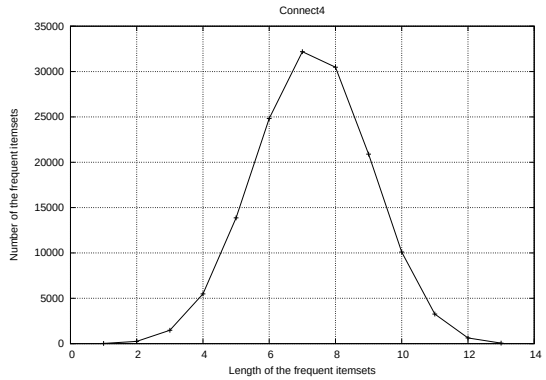
Name	Transactions	Items	Density	f_{min}
Connect4	67557	129	33.83%	85%

Table 5.14: Basic information of dataset Connect4

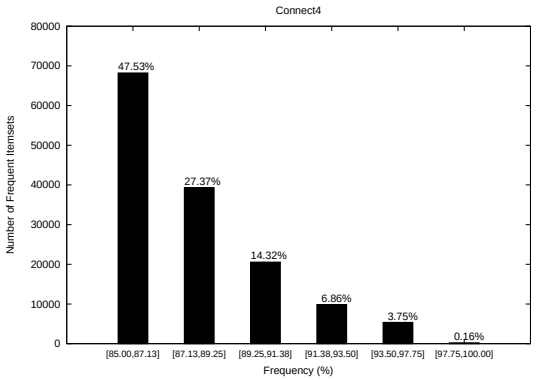
As nearly half of the frequent itemsets are distributed at the frequency interval of [85%, 87.13%], we can expect a high false negative rate even with a very accurate estimation.

Name		Connect4		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	26.91±0.69	0.62±0.12	1.16±0.03
	K=25	25.35±0.54	0.67±0.03	1.09±0.03
	K=50	22.67±0.75	0.75±0.06	0.97±0.03
	K=75	21.41±1.13	0.75±0.08	0.91±0.05
VFBM	K=15	23.34±1.33	**0.63±0.20**	0.98±0.06
	K=25	18.05±2.44	0.89±0.19	0.75±0.11
	K=50	11.43±1.28	1.24±0.17	0.46±0.05
	K=75	**10.57±0.97**	1.25±0.11	**0.43±0.04**
GSFBM	K=15	20.39±1.28	1.02±0.12	0.88±0.07
	K=25	16.79±0.94	1.06±0.22	0.7±0.05
	K=50	13.98±1.84	0.97±0.15	0.57±0.08
	K=75	11.28±1.31	1.19±0.12	0.44±0.06
VDPM	K=15	22.70±0.83	0.92±0.15	0.97±0.03
	K=25	16.93±1.47	0.83±0.05	0.70±0.07
	K=50	12.48±1.00	1.06±0.17	0.50±0.04
	K=75	11.59±1.34	1.33±0.44	0.47±0.05
GSDPM	K=13.6	23.90±1.76	1.08±0.01	1.02±0.08

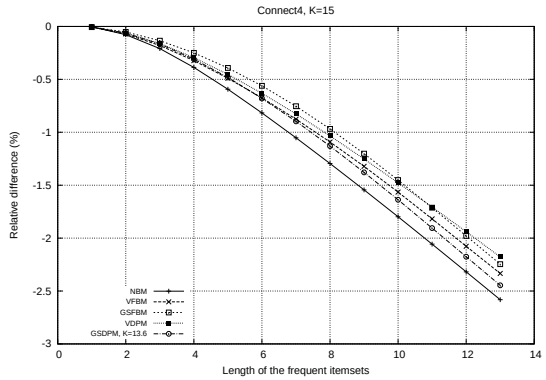
Table 5.15: Test result of dataset Connect4 (%), average of 5 runs



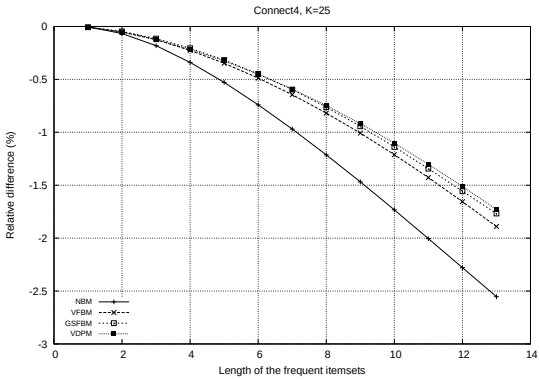
(a) Distribution on length



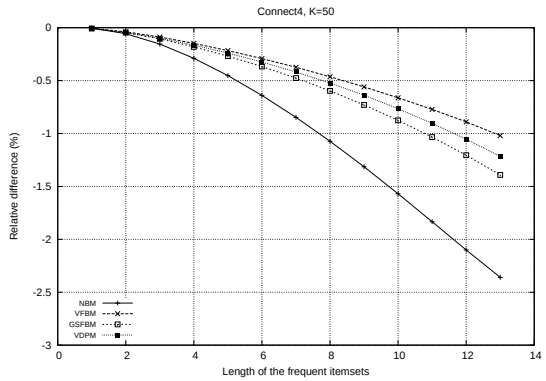
(b) Distribution on frequency



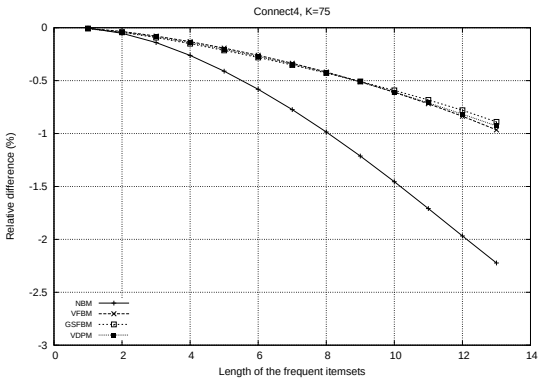
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.6: FI distribution and estimation trend of dataset Connect4

Although the false negative rates are over 10%, the estimations of the models are in fact quite accurate. GSDPM suggests 13.6 components on average, implying that this dataset is quite simple in structure. The under-estimation degrees of Bayesian models with 75 components are no more than 1%.

Table 5.16 and 5.17 show the time cost of the models. The training of the Bayesian models takes hours to converge. As a comparison, the non Bayesian mixture model is much faster.

Name		Connect4		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	16.80	35.88	2.14
	K=25	16.80	62.20	3.69
	K=50	17.40	154.30	8.87
	K=75	18.00	214.20	11.86
VFBM	K=15	18.00	3484.40	193.58
	K=25	17.80	3447.80	193.69
	K=50	17.40	3422.00	196.69
	K=75	18.20	3567.40	196.01
GSFBM	K=15	14.20	5646.60	397.65
	K=25	15.20	6175.00	406.20
	K=50	16.80	7024.00	418.03
	K=75	17.80	7650.80	429.86
VDPM	K=15	18.20	3523.20	193.60
	K=25	17.60	3404.80	193.44
	K=50	18.00	3517.40	195.41
	K=75	18.20	3567.80	196.03
GSDPM	K=13.6	21.60	8819.60	408.25

Table 5.16: Total training time cost and training time per iteration of dataset Connect4 (sec), average of 5 runs

	T_{On}
Eclat	52.95
K=15	0.19
K=25	0.24
K=50	0.37
K=75	0.47
K=13.6	0.17

Table 5.17: FI generation time of dataset Connect4 (sec), average of 5 runs

LetRecog

This dataset records the information of letter images for recognition.

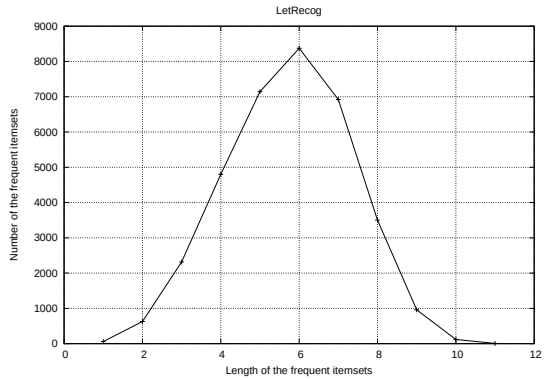
Name	Transactions	Items	Density	f_{min}
LetRecog	20000	106	16.53%	5%

Table 5.18: Basic information of dataset LetRecog

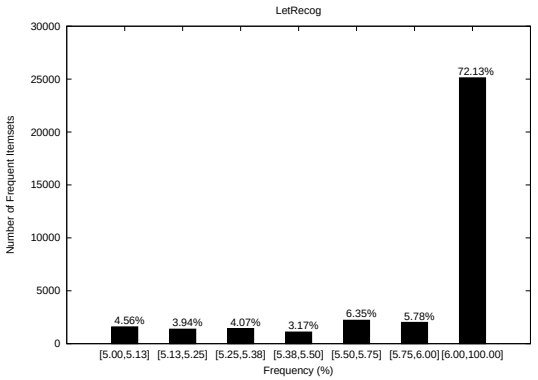
There are about 8.5% of the frequent itemsets in the frequency interval of [5%, 5.25%]. If the models under-estimates the frequency by 5%, the itemsets in this interval are likely to be labelled as infrequent.

Name		LetRecog		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	14.99±0.75	8.06±1.68	10.66±0.78
	K=25	13.09±1.12	6.33±0.67	9.02±0.67
	K=50	12.36±0.83	4.91±0.74	8.07±0.67
	K=75	10.76±1.15	5.13±0.39	7.29±0.58
VFBM	K=15	10.77±1.48	8.02±1.37	8.35±0.49
	K=25	9.06±0.56	5.93±0.69	6.74±0.39
	K=50	7.57±1.01	4.85±0.49	5.48±0.34
	K=75	**7.19±0.66**	4.05±0.40	**5.01±0.22**
GSFBM	K=15	11.58±1.10	6.35±0.37	8.20±0.50
	K=25	10.44±0.87	5.15±0.51	7.06±0.35
	K=50	8.62±0.76	4.49±0.37	5.88±0.35
	K=75	9.27±0.90	3.54±0.18	5.90±0.48
VDPM	K=15	10.72±0.99	7.65±0.80	8.14±0.40
	K=25	8.85±1.42	6.11±0.88	6.70±0.45
	K=50	7.33±0.84	4.75±0.92	5.43±0.27
	K=75	7.41±0.33	3.92±0.10	5.03±0.16
GSDPM	K=98.4	9.61±0.21	**3.29±0.10**	5.99±0.09

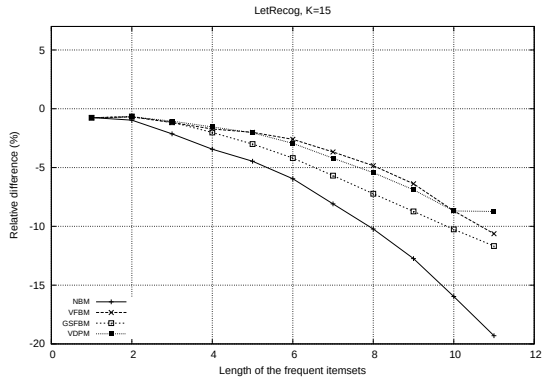
Table 5.19: Test result of dataset LetRecog (%), average of 5 runs



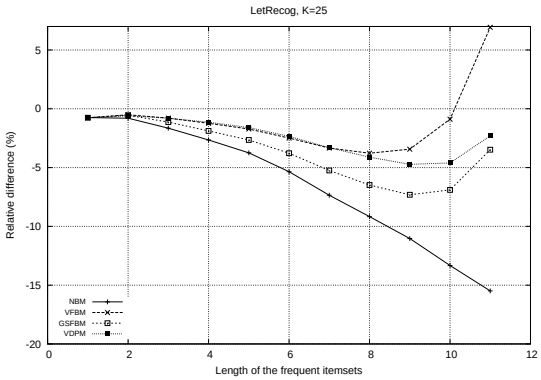
(a) Distribution on length



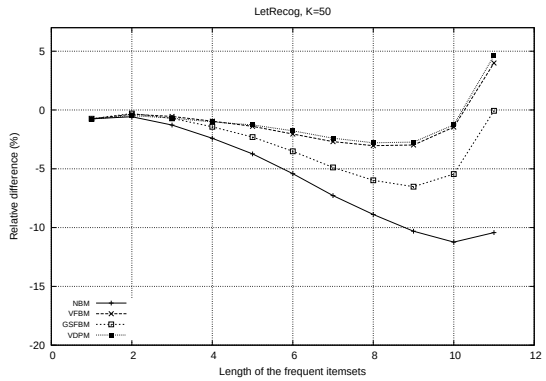
(b) Distribution on frequency



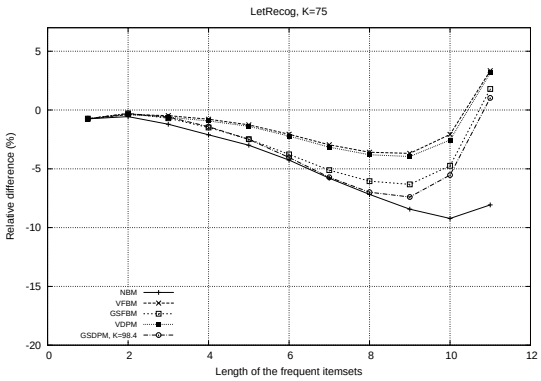
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.7: FI distribution and estimation trend of dataset LetRecog

For this dataset, variational methods work slightly better than sampling methods. GSDPM uses nearly 100 components, implying that 75 components is not enough for this dataset. The average under-estimation of the models for this dataset is around 3%-5%.

Name		LetRecog		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	19.60	10.38	0.53
	K=25	19.00	16.85	0.88
	K=50	21.20	39.13	1.84
	K=75	20.00	57.36	2.87
VFBM	K=15	23.00	165.80	7.21
	K=25	22.20	161.60	7.28
	K=50	23.80	177.00	7.44
	K=75	21.60	164.40	7.61
GSFBM	K=15	13.00	214.80	16.52
	K=25	12.80	226.20	17.67
	K=50	16.20	330.60	20.40
	K=75	16.80	392.20	23.35
VDPM	K=15	23.00	165.80	7.21
	K=25	22.20	161.60	7.28
	K=50	23.80	177.00	7.44
	K=75	21.60	164.40	7.61
GSDPM	K=98.4	125.60	3128.60	24.87

Table 5.20: Total training time cost and training time per iteration of dataset LetRecog (sec), average of 5 runs

	T_{On}
Eclat	3.58
K=15	0.14
K=25	0.14
K=50	0.18
K=75	0.21
K=98.4	0.25

Table 5.21: FI generation time of dataset LetRecog (sec), average of 5 runs

MS Web

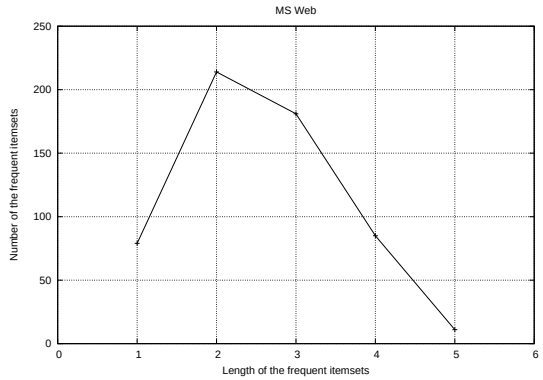
The data records the use of `www.microsoft.com` by 38,000 anonymous, randomly-selected users. For each user, the data lists all the areas of the website that the user visited in a one week timeframe. It is a record of a web log. This dataset is very sparse as its density is only about 1%. We have to decrease the threshold to 0.5% to obtain enough frequent itemsets. On the other hand, we do not want to set the threshold too low as a very low frequency is not statistically significant for correlation analysis.

Name	Transactions	Items	Density	f_{min}
MS Web	37711	294	1.03%	0.5%

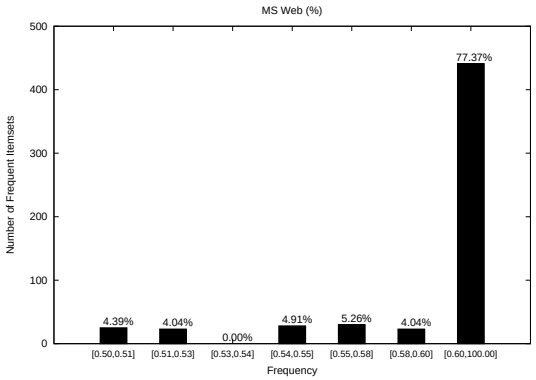
Table 5.22: Basic information of dataset MS Web

Name		MS Web		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	47.89±1.52	7.02±0.57	41.77±1.51
	K=25	44.91±0.67	6.44±0.50	38.49±0.64
	K=50	43.96±0.67	5.45±0.43	36.47±0.50
	K=75	39.72±1.07	4.77±0.32	32.17±1.01
VFBM	K=15	37.40±2.41	4.52±0.75	30.08±2.16
	K=25	34.11±0.67	3.74±0.37	26.46±0.52
	K=50	32.95±0.44	3.39±0.11	25.17±0.53
	K=75	32.63±2.54	**3.27±0.71**	24.41±2.51
GSFBM	K=15	36.53±0.99	4.83±0.72	29.32±0.51
	K=25	33.96±2.38	4.56±0.59	26.67±1.49
	K=50	30.11±0.72	4.14±0.19	23.23±0.23
	K=75	28.49±0.58	4.18±0.26	22.21±0.59
VDPM	K=15	38.18±2.13	5.04±0.68	30.22±1.30
	K=25	35.23±1.37	4.06±0.14	26.75±0.88
	K=50	33.37±0.37	3.50±0.23	25.66±0.37
	K=75	33.96±0.54	3.49±0.39	25.51±0.56
GSDPM	K=140.6	**28.00±0.75**	4.33±0.25	**20.95±0.43**

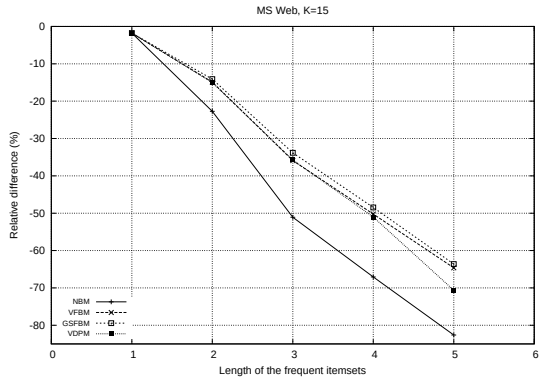
Table 5.23: Test result of dataset MS Web (%), average of 5 runs



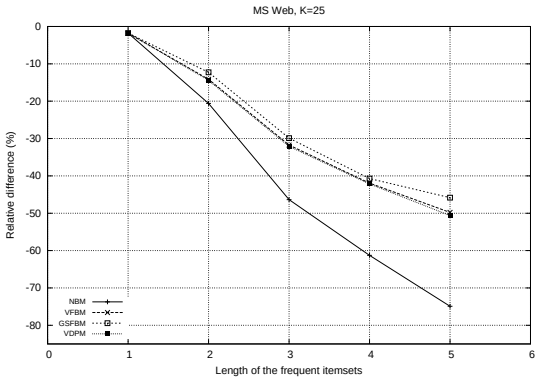
(a) Distribution on length



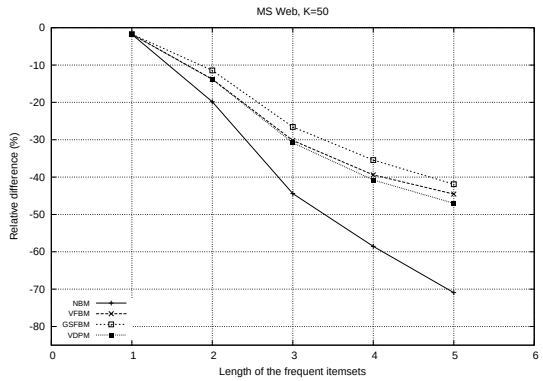
(b) Distribution on frequency



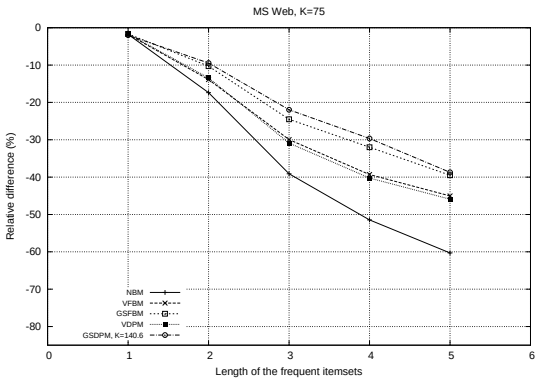
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.8: FI distribution and estimation trend of dataset MS Web

According to the result in Figure 5.8 and Table 5.23, the under-estimation is serious for all five models. GSDPM uses 140.6 components on average but the estimation is still not good enough.

Name		MS Web		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	25.00	40.00	1.60
	K=25	23.87	69.21	2.90
	K=50	22.00	132.88	6.04
	K=75	21.20	182.40	8.60
VFBM	K=15	33.20	155.00	4.72
	K=25	31.40	152.40	4.88
	K=50	28.60	160.00	5.60
	K=75	27.80	174.20	6.27
GSFBM	K=15	9.00	154.20	17.13
	K=25	9.60	208.60	21.72
	K=50	9.00	235.80	26.20
	K=75	9.00	305.80	33.98
VDPM	K=15	32.80	155.00	4.73
	K=25	30.60	152.40	4.98
	K=50	28.40	160.00	5.63
	K=75	26.80	174.20	6.51
GSDPM	K=140.6	150.00	7184.50	47.90

Table 5.24: Total training time cost and training time per iteration of dataset MS Web (sec), average of 5 runs

	T_{On}
Eclat	0.29
K=15	0.04
K=25	0.03
K=50	0.05
K=75	0.09
K=140.6	0.19

Table 5.25: FI generation time of dataset MS Web (sec), average of 5 runs

Mushroom

This dataset consists of records drawn from The Audubon Society Field Guide to North American Mushrooms.

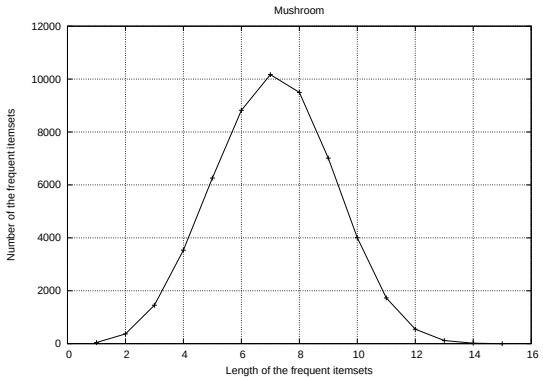
Name	Transactions	Items	Density	f_{min}
Mushroom	8125	119	19.33%	20.0%

Table 5.26: Basic information of dataset Mushroom

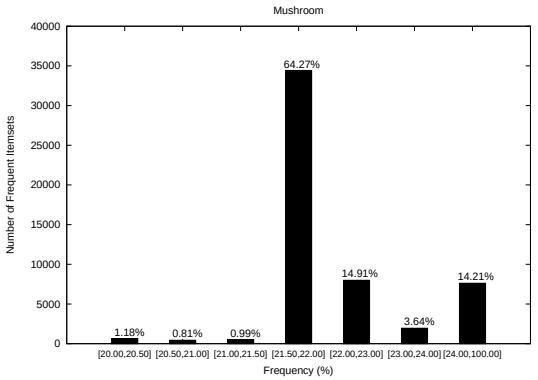
Checking the distribution of frequent itemsets on frequency, we notice that most itemsets are distributed in a small range of [21.5%,22%]. This fact may cause the false negative rate to change very rapidly with a certain estimation error.

Name		Mushroom		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	71.36±1.75	1.63±0.12	11.59±0.46
	K=25	68.53±1.12	1.40±0.11	10.17±0.21
	K=50	65.72±1.05	1.33±0.09	9.10±0.10
	K=75	25.16±2.60	0.63±0.01	6.81±0.05
VFBM	K=15	7.58±13.17	0.75±0.16	5.04±1.18
	K=25	6.27±7.33	**0.40±0.05**	5.70±0.38
	K=50	2.07±1.80	0.59±0.04	5.27±0.09
	K=75	0.97±0.05	0.62±0.03	5.48±0.01
GSFBM	K=15	**0.77±0.01**	0.74±0.06	4.09±0.07
	K=25	0.90±0.02	0.72±0.06	4.12±0.03
	K=50	0.90±0.03	0.78±0.05	4.32±0.05
	K=75	0.86±0.01	0.79±0.01	4.29±0.03
VDPM	K=15	7.86±9.08	0.83±0.21	4.90±0.56
	K=25	6.09±10.26	0.75±0.19	4.91±0.66
	K=50	1.23±0.39	0.57±0.10	5.44±0.51
	K=75	2.92±1.36	0.63±0.05	5.83±0.82
GSDPM	K=18.4	0.85±0.01	0.70±0.04	**3.99±0.03**

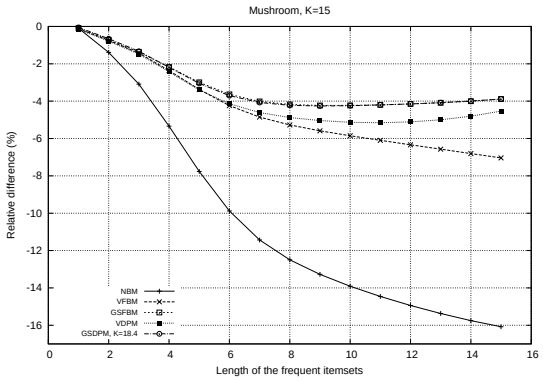
Table 5.27: Test result of dataset Mushroom (%), average of 5 runs



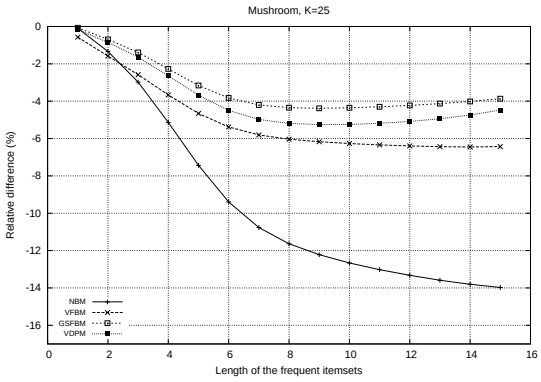
(a) Distribution on length



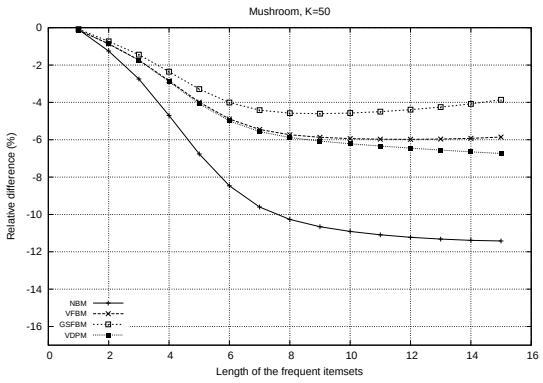
(b) Distribution on frequency



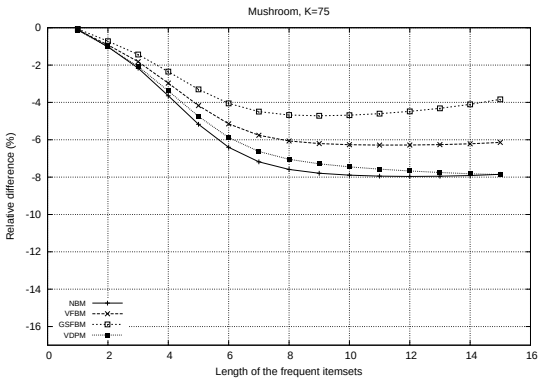
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.9: FI distribution and estimation trend of dataset Mushroom

From Table 5.27 we observe that when the estimation error is below 5%, the false negative rate improves dramatically. The difference in relative error between non-Bayesian and Bayesian models is about 6 percent. However the difference in the false negative rate is extraordinary. When training the VDP model, it

is very likely that the algorithm becomes stuck in some local minimum, causing significant differences among the five trials. In some trials, the F^- are as low as about 1% while in other trials, the F^- rise to about 25%. This is the reason that the standard deviations of VDP at truncation levels of 15 and 25 are larger than the averages. On the other hand, the Gibbs sampler suggests that about 19 components are enough to approximate the distribution of 'mushroom'. It gives better results than VDP at a truncation level of 50.

Name		Mushroom		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	12.20	1.72	0.14
	K=25	13.40	3.11	0.23
	K=50	13.80	6.63	0.48
	K=75	13.00	9.19	0.71
VFBM	K=15	13.00	20.00	1.55
	K=25	13.60	21.80	1.61
	K=50	17.20	20.00	1.18
	K=75	15.00	21.60	1.46
GSFBM	K=15	11.00	45.20	4.11
	K=25	12.60	59.20	4.69
	K=50	11.80	68.00	5.76
	K=75	11.20	79.00	7.05
VDPM	K=15	12.80	20.00	1.56
	K=25	14.00	21.80	1.56
	K=50	12.40	20.00	1.61
	K=75	12.80	21.60	1.69
GSDPM	K=18.4	20.80	90.60	4.37

Table 5.28: Total training time cost and training time per iteration of dataset Mushroom (sec), average of 5 runs

	T_{On}
Eclat	3.77
K=15	0.12
K=25	0.13
K=50	0.19
K=75	0.21
K=18.4	0.12

Table 5.29: FI generation time of dataset Mushroom (sec), average of 5 runs

Nursery

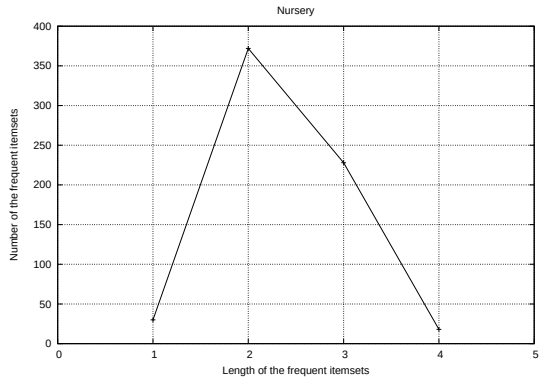
Nursery Database was derived from a hierarchical decision model originally developed to rank applications for nursery schools.

Name	Transactions	Items	Density	f_{min}
Nursery	12960	32	28.62%	5%

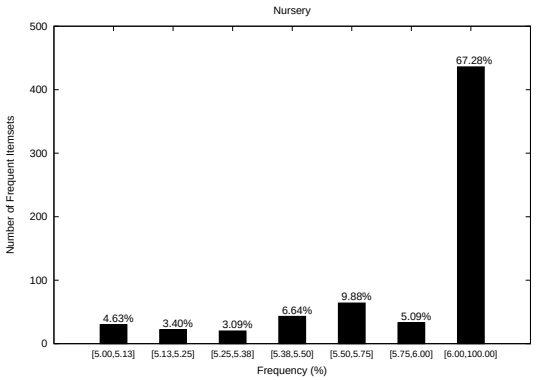
Table 5.30: Basic information of dataset Nursery, average of 5 runs

Name		Nursery		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	14.99±0.75	8.06±1.68	10.66±0.78
	K=25	13.09±1.12	6.33±0.67	9.02±0.67
	K=50	12.36±0.83	4.91±0.74	8.07±0.67
	K=75	10.76±1.15	5.13±0.39	7.29±0.58
VFBM	K=15	10.77±1.48	8.02±1.37	8.35±0.49
	K=25	9.06±0.56	5.93±0.69	6.74±0.39
	K=50	7.57±1.01	4.85±0.49	5.48±0.34
	K=75	**7.19±0.66**	4.05±0.40	**5.01±0.22**
GSFBM	K=15	11.58±1.10	6.35±0.37	8.2±0.50
	K=25	10.44±0.87	5.15±0.51	7.06±0.35
	K=50	8.62±0.76	4.49±0.37	5.88±0.35
	K=75	9.27±0.90	3.54±0.18	5.90±0.48
VDPM	K=15	10.72±0.99	7.65±0.80	8.14±0.40
	K=25	8.85±1.42	6.11±0.88	6.70±0.45
	K=50	7.33±0.84	4.75±0.92	5.43±0.27
	K=75	7.41±0.33	3.92±0.10	5.03±0.16
GSDPM	K=17.8	9.61±0.21	**3.29±0.10**	5.99±0.09

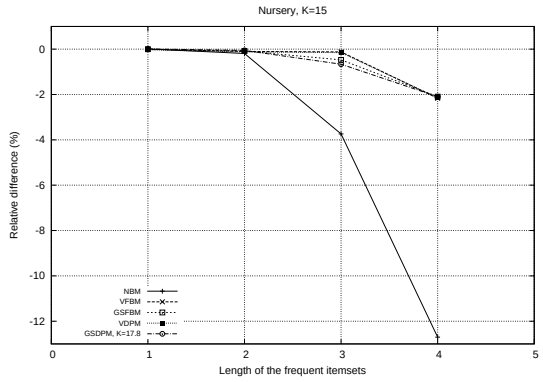
Table 5.31: Test result of dataset Nursery (%), average of 5 runs



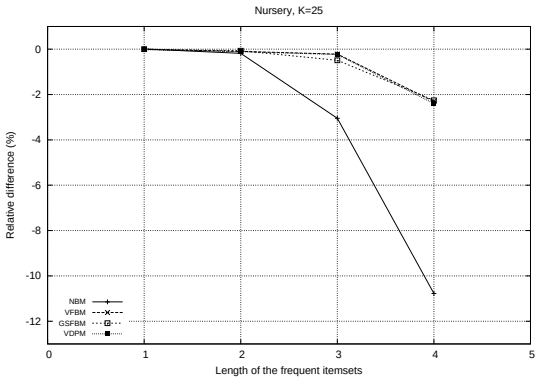
(a) Distribution on length



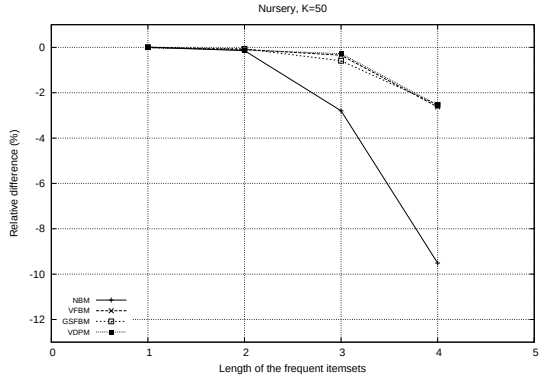
(b) Distribution on frequency



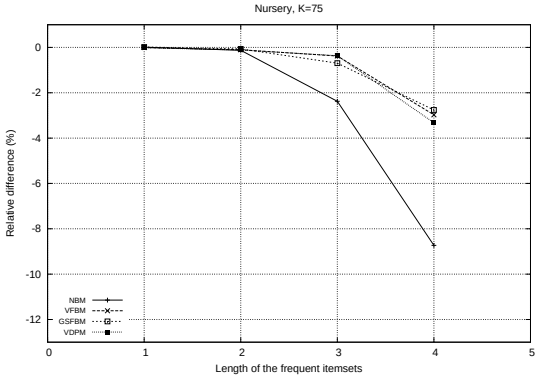
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.10: FI distribution and estimation trend of dataset Nursery

Name		Nursery		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	27.13	5.43	0.20
	K=25	25.53	8.89	0.35
	K=50	26.93	17.88	0.66
	K=75	26.60	26.28	0.99
VFBM	K=15	32.80	52.60	1.60
	K=25	31.20	50.60	1.62
	K=50	30.80	51.00	1.66
	K=75	29.80	50.80	1.70
GSFBM	K=15	16.00	63.00	3.94
	K=25	16.60	69.20	4.17
	K=50	19.40	92.00	4.74
	K=75	17.60	93.80	5.33
VDPM	K=15	32.60	52.60	1.61
	K=25	28.80	46.40	1.61
	K=50	30.60	50.80	1.66
	K=75	32.40	55.60	1.72
GSDPM	K=17.8	66.00	304.40	4.64

Table 5.32: Total training time cost and training time per iteration of dataset Nursery (sec), average of 5 runs

	T_{On}
Eclat	3.58
K=15	0.03
K=25	0.04
K=50	0.05
K=75	0.05
K=17.8	0.03

Table 5.33: FI generation time of dataset Nursery (sec), average of 5 runs

PenDigits

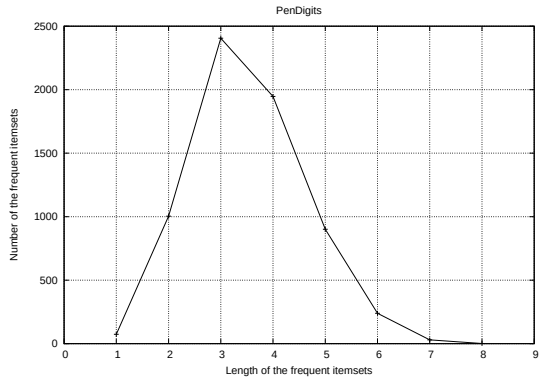
This dataset is a collection of records of hand writing digits.

Name	Transactions	Items	Density	f_{min}
Nursery	10992	89	19.60%	5%

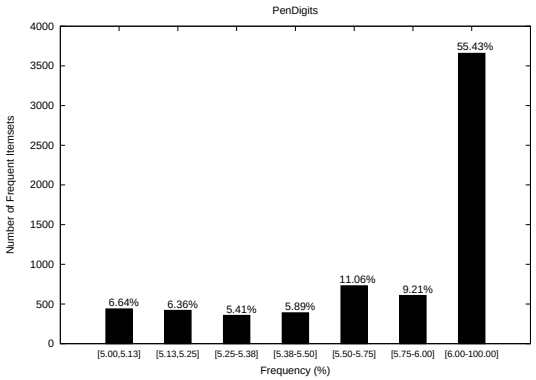
Table 5.34: Basic information of dataset PenDigits

Name		Nursery		
Criteria		F^-	F^+	$\hat{E}$
NBM	K=15	44.01±1.06	9.49±0.57	20.03±1.06
	K=25	39.76±0.80	8.42±0.77	17.15±0.73
	K=50	34.59±1.50	6.74±0.50	14.10±0.75
	K=75	30.06±0.92	**6.66±0.31**	11.83±0.29
VFBM	K=15	31.55±2.80	10.91±0.92	14.5±1.75
	K=25	25.49±1.09	8.66±0.33	10.88±0.72
	K=50	21.55±0.74	8.19±0.29	9.18±0.29
	K=75	20.82±0.96	7.86±0.34	8.68±0.34
GSFBM	K=15	29.34±1.86	10.56±0.54	13.26±1.06
	K=25	23.71±0.85	8.97±0.42	10.22±0.26
	K=50	19.67±0.86	8.36±0.68	8.57±0.36
	K=75	**18.12±0.15**	8.23±0.12	**7.96±0.08**
VDPM	K=15	30.67±1.69	9.98±0.43	13.72±1.12
	K=25	24.66±1.22	8.99±0.74	10.69±0.57
	K=50	22.18±0.85	8.42±0.58	9.34±0.36
	K=75	21.21±0.87	8.01±0.22	8.87±0.23
GSDPM	K=60.2	18.65±0.15	8.02±0.19	8.01±0.07

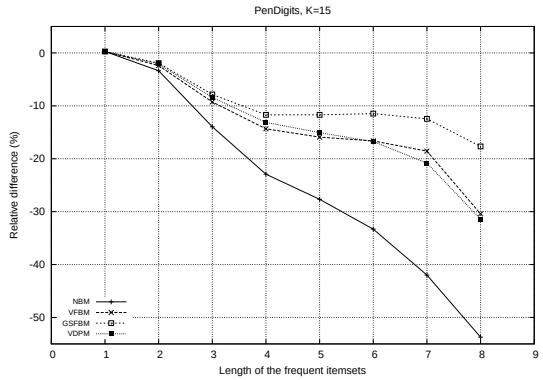
Table 5.35: Test result of dataset PenDigits (%), average of 5 runs



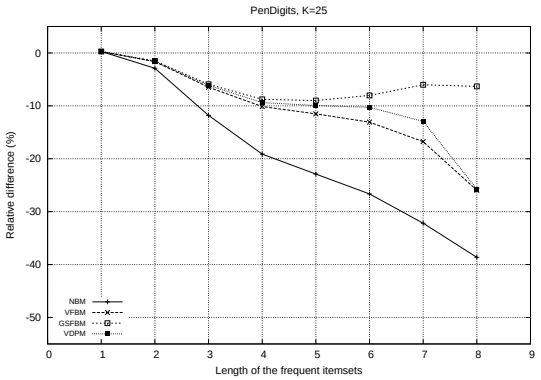
(a) Distribution on length



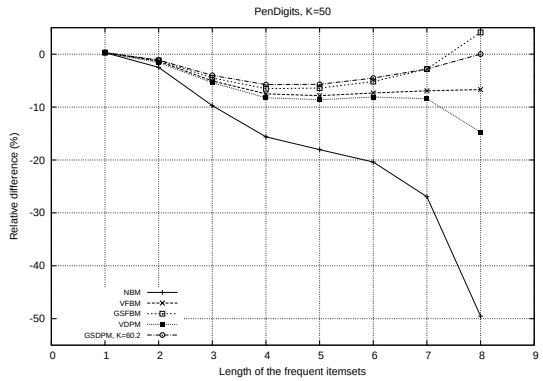
(b) Distribution on frequency



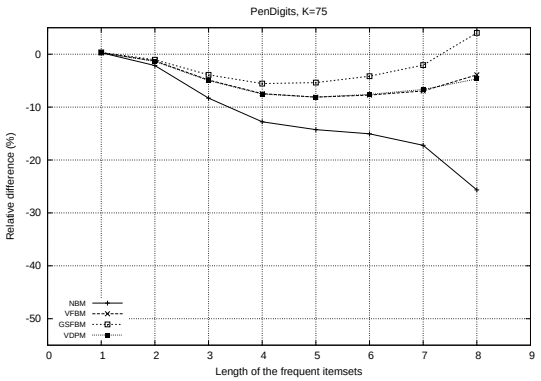
(c) Estimation trend, K=15



(d) Estimation trend, K=25



(e) Estimation trend, K=50



(f) Estimation trend, K=75

Figure 5.11: FI distribution and estimation trend of dataset PenDigits

The best estimation in all the models is about 8%. The false negative rate and false positive rate are also quite high.

Name		Nursery		
Criteria		Iterations	T_{Off}	T_{Off}/I
NBM	K=15	18.40	5.13	0.28
	K=25	17.80	8.19	0.46
	K=50	17.00	15.64	0.92
	K=75	15.80	22.18	1.40
VFBM	K=15	20.60	42.20	2.05
	K=25	20.20	42.20	2.09
	K=50	17.20	37.20	2.16
	K=75	16.00	35.80	2.24
GSFBM	K=15	12.80	66.20	5.17
	K=25	14.00	80.00	5.71
	K=50	15.80	111.40	7.06
	K=75	14.20	118.40	8.33
VDPM	K=15	21.60	44.20	2.05
	K=25	19.20	40.00	2.08
	K=50	18.00	39.00	2.17
	K=75	17.20	38.60	2.24
GSDPM	K=60.2	113.60	943.20	8.30

Table 5.36: Total training time cost and training time per iteration of dataset PenDigits (sec), average of 5 runs

	T_{On}
Eclat	1.97
K=15	0.07
K=25	0.08
K=50	0.09
K=75	0.11
K=60.2	0.11

Table 5.37: FI generation time of dataset PenDigits (sec), average of 5 runs

5.4 Discussion

In this section we summarize the experiment results and compare the performance of the four algorithms for all nine real datasets. Table 5.38 shows the resulting comparisons.

Name	Best Estimation ($\hat{E}$)	Best Method
Accidents	2.04±0.05	GSDPM
Adult	5.11±0.23	GSFBM
Chess	2.77±0.08	VFBM
Connect	0.43±0.04	VFBM
LetRecog	5.01±0.02	VFBM
MS Web	20.95±0.43	GSDPM
Mushroom	3.99±0.03	GSDPM
Nursery	5.11±0.22	VFBM
PenDigits	7.96±0.08	GSFBM

Table 5.38: Test result comparison (%)

When GSDPM works best, it normally means that 75 components are not enough to describe the data. Aside from Mushroom, only GSDPM uses less than 20 components. Generally, although VDPM never appears in this table, it has very similar performance to VFBM. All four algorithms give comparable results. They all work better than the non-Bayesian mixture model.

Generally the MCMC method should achieve a better result than the variational methods with enough iterations, especially when a large number of components are used. As shown in the tests with synthetic datasets, when we increase K, the difference between GSFBM and variational methods becomes larger. However, in our experiments, for some cases variational methods indeed achieve a better result. We think the reason may be the convergence condition. For the convenience of the experiment, we unify all the methods to the same convergence condition. However, to diagnose the convergence of Gibbs sampling is more complicated.

Comparing the GSDPM with the other three algorithms, it emerges that GSDPM's key advantage is that it is nonparametric, which means the problem of choosing the number of components can be left to the algorithm itself. When faced with an unknown dataset, choosing an appropriate K is difficult. One has

to try several times to determine the K. Tests on the nine test cases have shown that GSDPM can find the proper number. The idea of choosing K automatically is simply as follows: create a new cluster if no existing cluster fits the current data point significantly better than the average across the whole population. DP mixture via Gibbs sampler implements this idea in a stochastic way.

From the experiment results come several interesting observations about mixture models for frequent itemset discovery. Firstly, as the false negative rates are always much higher than the false positive rates, we observe that the mixture models tend to under-estimate the probabilities of the frequent itemsets.

Additionally, there appears significant difference between the models' performance on MS Web and on the other datasets. All the results of the datasets are acceptable except MS Web, in which the errors of the models are over 20%. As a probability model, this estimation is not usable for further prediction. The under-estimation in MS Web is also much more serious than the rest. Checking Table 5.22, we notice that MS Web is much sparser than the other datasets. Further background investigation into the datasets shows that this difference may be caused by the fact that the correlations between items within the other datasets are much stronger than they are in MS Web, which is sparse. Therefore the distribution of these data records can be better approximated by a mixture model structure. More improvements for the mixture model may be required to achieve a satisfying performance for sparse datasets.

Furthermore, we think the reason for under-estimation is that the mixture model is a mixture of independent models. In independent Bernoulli model, the probabilities of patterns are simply the multiplications of the parameters, which always under-estimate the correlated item combinations. In mixture models, compensations are made by the assumption of conditional independence. Correlations of the datasets are contained by different components and described by various groups of conditional probabilities. For strong correlated datasets such as classification data, feature attributes of each class would show high dependencies. These dependencies are strong and simple because the correlated attributes are clustered by the latent classes. Under these circumstances, most correlations are represented by the model rendering any under-estimation tolerable. However, for non-classification datasets where the correlations are not so strong and relatively loose and chaotic, the mixture model cannot hold all the complexity with a feasible number of components. We think this explains why for, MS Web data, there

is severe under-estimation for all five algorithms.

Another reason for the mixture models' poor performance may be the contradiction of the symmetric likelihood function and the asymmetric way of generating frequent itemsets. As described in Chapters 3 and 4, we predict the probability of an itemset by simply ignoring the "0"s. For a sparse dataset, as the likelihood function is symmetric to "1"s and "0"s and the number of "0"s is far more than the number of "1"s, the algorithm will pay more attention to fitting the "0"s. However, in the process of generating frequent itemsets, the situation of "0"s is ignored. To overcome this contradiction, some modification on the likelihood function may be necessary. This can be a direction for future work.

The training time costs of the Bayesian mixture models are longer than when mining the dataset directly. However if the model is ready, the online prediction time is much shorter. We list comparison of the data mining time cost with the online prediction time of the models with 75 components of the 9 real datasets in Table 5.39. For most datasets, the model prediction is over 10 times faster than data mining. For some large and dense datasets, the prediction process is even over hundreds times faster. The reason is that the frequency of an itemset by the model is calculated simply by some multiplications and summations while in data mining it is basically obtained by counting. Another two good features of probability model prediction are that the model is irrelevant to the scale of the original dataset and the minimum frequency threshold. Being irrelevant to the scale of the dataset means that for large datasets the prediction time cost can gain more advantage comparing with data mining, as we showed by the datasets Accidents, Adult and Connect4. Being irrelevant to the threshold means that we need only to train the model once and use them multiple times for different thresholds. As one of the difficulty of the frequent itemset mining is finding proper threshold for the dataset, the probability model allows more trials within the same amount of time.

	Mining	Prediction	M/P
Accidents	76.31	0.79	96.59
Adult	563.70	1.04	542.02
Chess	47.36	6.20	7.64
Connect4	52.95	0.17	311.47
LetRecog	3.58	0.25	14.32
MS Web	0.29	0.09	3.22
Mushroom	3.77	0.21	17.95
Nursery	3.58	0.05	71.60
PenDigits	1.97	0.11	17.91

Table 5.39: FI generation time comparison

For training time cost, variational methods are faster than sampling methods. With a good implementation, variational methods can gain advantage on time consumption when the number of components increases. The main restriction of sampling methods is that they need to update all the parameters of the model after each draw. This feature on the other hand causes GSFBM to use less iterations in order to achieve convergence. Bayesian inferences are widely used in many machine learning tasks and they can achieve a good result. However, the time issue limits the Bayesian inference to larger dataset applications. Variational methods have some advantages when compared with sampling, but still involve computation of complex functions and are still slower than non-Bayesian algorithms. If we compare the variational methods and the sampling methods from both the accuracy aspect and the time aspect, we can discover the tradeoff of speed versus accuracy. The choice of using which methods should depend on the requirement of the specific problems.

Chapter 6

Conclusion

In this thesis we have proposed Bayesian mixture models with a view to improving upon the non-Bayesian mixture model for frequent itemset mining, thereby enhancing the efficiency of data mining techniques. We have described two Bayesian mixture models, introducing both a finite Dirichlet distribution and an infinite Dirichlet process prior to the mixture model. We have also proposed two inference algorithms, namely variational approximation and collapsed Gibbs sampling, for use with the two models respectively. We have evaluated and compared the four algorithms with the non-Bayesian mixture model on five synthetic and nine real datasets.

By the analysis of the experimental results, we think that for most cases the Bayesian mixture models can achieve a quite accurate result. Comparing with the frequent itemset mining, the model can provide a structured explanation of the dataset. It explains how the dataset is generated by different probability-based components. Therefore all related probabilities can be calculated directly with a small accuracy loss. By the model we can easily generate all frequent itemsets with much less time. Furthermore, for non-frequent itemsets or some other kind of queries such as joint probabilities, mixture model can provide results that the set of frequent itemsets cannot. This property of the model means that we can recover the dataset approximately simply by sampling based on the model. Although we did not show the scale of the models in Chapter 5, it is clearly that for most cases the sizes of the models are also much smaller than the original data since the models only contain $K(I+1)$ floating numbers where K is the number of components and I is the number of items. However, the dataset contain NL integers where N is the number of transactions and L is the average length of the

transactions and the product is often much larger than $K(I+1)$. With a smaller scale, the model can be more convenient to use than the original dataset when lack of large memory or slow to scan the whole database.

6.1 Contributions

This thesis consists of several significant contributions and conclusions:

1. We have improved on non-Bayesian mixture models for frequent itemset mining with Bayesian mixture models. Comparing with the non-Bayesian model, the Bayesian models improve the quality of estimation by reducing the bias of the non-Bayesian mixture model. The Bayesian models also solve the problem of parameter smoothing in the non-Bayesian model inference by involving proper priors. Furthermore, for some situations if there is background knowledge, it is easy for the Bayesian models to incorporate it. Certainly, the experimental results show that Bayesian mixture models improve accuracy without adding extra model complexity. The cost of accuracy is that the Bayesian mixture models are computational more expensive.

2. We have applied the nonparametric Dirichlet process prior to the mixture model to remove limitations on the number of components. The algorithm can detect the proper number of components automatically, which means that under-fitting problems in mixture models caused by having too few components can be effectively addressed. Combined with the benefits of introducing Bayesian priors to the mixture model, the Dirichlet process mixture model can improve the original probability model from both the over-fitting and the under-fitting sides.

3. We have developed both variational approximation and MCMC sampling methods for the models. MCMC sampling can achieve better accuracy with enough iterations whilst variational methods can provide a comparable result much faster.

4. The frequent itemset mining process has been greatly accelerated based on the probability models comparing with traditional data mining techniques.

As the mixture model is independent from the minimum frequency threshold and the itemset generation from the model is irrelevant to the scale of the dataset, it is very convenient to do fast mining on large and dense datasets with various thresholds using the models.

5. In the evaluation of the algorithms, we have found that the current Bayesian mixture models perform poorly on sparse datasets. This may be due to weak correlation between items and the contradiction between likelihood function and the way we generate frequent itemsets. Problems caused by such sparsity represent a possible direction for ongoing improvement.

6.2 Future Work

With the rapid development of Bayesian inference in recent years, the Bayesian mixture model for frequent itemset mining stands to be improved in many ways, as briefly discussed below.

More delicate structure When we try to understand the reason of model underestimation, we review the current mixture model. The current mixture model is a two-layer model. The first layer is the mixing proportion and the second layer is the conditional probabilities. We can imagine the mixing proportion as a point on a $K-1$ simplex. A non-Bayesian mixture model tries to find the optimal point which maximizes the likelihood of the model. Bayesian methods extend the optimal point to an optimal distribution of the mixing proportion by introducing a Dirichlet distribution as a prior. However, for more complex data, a simple Dirichlet distribution may still not be enough to describe the distribution over the simplex. We can also think of this problem in terms of a supermarket scenario. In our original mixture model, the whole list of purchasing transactions is a mixture. It is a mixture of types such as food, clothes, books etc. and each transaction is drawn from one of the types. However it is clear that when we go to supermarket we do not restrict ourselves to buying things only from one type. A more natural assumption is that each purchasing transaction is a mixture of types and the types are shared by all transactions. Therefore the mixture model becomes a three-layer model. In fact, the three-layer mixture model has already been used in natural language processing. For a finite number of base components, the model is called the *Latent Dirichlet Allocation* (LDA) [BNJ03]. For an infinite number of base components, the model is called the

Hierarchical Dirichlet Process mixture model (HDP mixture model) [TJBB04]. According to these models, each transaction of the dataset can be viewed as a mixture of different latent components. Whilst this may fit the data better, it could come with a cost of extra complexity.

Sparsity During the experiment with MS Web, we have noticed that all the mixture models gravely under-estimate the itemsets. As analyzed at the conclusion of Chapter 5, the contradiction between the symmetric likelihood function and the asymmetric way of generating frequent itemsets may be the cause of this phenomenon. For sparse datasets, this contradiction is more serious than for dense datasets. Some compensation, such as applying a sparsity factor on the likelihood function might be a possible solution to this problem.

Improvement on variational approximation Although MCMC sampling for Bayesian inference can normally achieve better accuracy, its usage is greatly limited by its speed. Variational methods therefore become an important alternative. During work on this thesis, improvements have indeed been made on variational inference such as [KWT07, WTK08, WD10]. These works focus on raising both the speed and quality of variational approximation and therefore represent possibilities for improvements to current variational algorithms of Bayesian models.

Appendix A

Related Computations

A.1 The Computation of $\mathcal{L}$ in Section 3.3

In Chapter 3 we need to expand and compute $\mathcal{L}$ as the following

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\eta}, \boldsymbol{\nu} : \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}) \\
&= E_q[\ln p(\mathcal{T}, \mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi} | \alpha, \boldsymbol{\beta}, \boldsymbol{\gamma})] - E_q[\ln q(\mathcal{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})] \\
&= \underbrace{E_q[\ln p(\boldsymbol{\pi}|\alpha)]}_{①} \underbrace{+E_q[\ln p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma})]}_{②} \underbrace{+E_q[\ln p(\mathcal{Z}|\boldsymbol{\pi})]}_{③} \underbrace{+E_q[\ln p(\mathcal{T}|\mathcal{Z}, \boldsymbol{\phi})]}_{④} \\
&\quad \underbrace{-E_q[\ln q(\boldsymbol{\pi}|\boldsymbol{\rho})]}_{⑤} \underbrace{-E_q[\ln q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu})]}_{⑥} \underbrace{-E_q[\ln q(\mathcal{Z}|\boldsymbol{\tau})]}_{⑦}
\end{aligned} \tag{A.1}$$

We treat the terms one by one. In the computation we use the fact that $E[\ln \pi_k] = \Psi(\alpha_k) - \Psi(\sum_{k'=1}^{K} \alpha_{k'})$ if $\boldsymbol{\pi} \sim \mathrm{Dir}(\alpha)$ where $\Psi(\cdot)$ is the digamma function.

$$\begin{aligned}
① &= E_q[\ln p(\boldsymbol{\pi}|\alpha)] \\
&= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} q(\mathcal{Z}, \boldsymbol{\phi}, \boldsymbol{\pi}) \ln p(\boldsymbol{\pi}|\alpha) d\boldsymbol{\phi} d\boldsymbol{\pi} \\
&= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} q(\mathcal{Z}) q(\boldsymbol{\phi}) q(\boldsymbol{\pi}|\boldsymbol{\rho}) \ln p(\boldsymbol{\pi}|\alpha) d\boldsymbol{\phi} d\boldsymbol{\pi} \\
&= \int_{\boldsymbol{\pi}} q(\boldsymbol{\pi}|\boldsymbol{\rho}) \ln p(\boldsymbol{\pi}|\alpha) d\boldsymbol{\pi} \\
&= \int_{\boldsymbol{\pi}} q(\boldsymbol{\pi}|\boldsymbol{\rho}) \ln \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} \prod_{k=1}^{K} \pi_k^{\frac{\alpha}{K}-1} d\boldsymbol{\pi}
\end{aligned}$$

$$= \ln \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} + E_q \left[\sum_{k=1}^{K} (\frac{\alpha}{K} - 1) \ln \pi_k \right]$$

$$= \ln \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} + (\frac{\alpha}{K} - 1) \sum_{k=1}^{K} [\Psi(\rho_k) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})] \tag{A.2}$$

$$\textcircled{2} = E_q[\ln p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma})]$$

$$= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} q(\mathcal{Z}, \boldsymbol{\phi}, \boldsymbol{\pi}) \ln p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma}) d\boldsymbol{\phi} d\boldsymbol{\pi}$$

$$= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} q(\mathcal{Z}) q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) q(\boldsymbol{\pi}) \ln p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma}) d\boldsymbol{\phi} d\boldsymbol{\pi}$$

$$= \int_{\boldsymbol{\phi}} q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \ln p(\boldsymbol{\phi}|\boldsymbol{\beta}, \boldsymbol{\gamma}) d\boldsymbol{\phi}$$

$$= \int_{\boldsymbol{\phi}} q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \ln \prod_{i=1}^{D} \prod_{k=1}^{K} p(\phi_{ik}|\beta_i, \gamma_i) d\boldsymbol{\phi}$$

$$= \sum_{i=1}^{D} \sum_{k=1}^{K} \int_{\boldsymbol{\phi}} q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \ln p(\phi_{ik}|\beta_i, \gamma_i) d\boldsymbol{\phi}$$

$$= \sum_{i=1}^{D} \sum_{k=1}^{K} \int_{\boldsymbol{\phi}} q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \ln \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \phi_{ik}^{\beta_i - 1} (1 - \phi_{ik})^{\gamma_i - 1} d\boldsymbol{\phi}$$

$$= \sum_{i=1}^{D} \sum_{k=1}^{K} \ln \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} + \sum_{i=1}^{D} \sum_{k=1}^{K} E_q[(\beta_i - 1) \ln \phi_{ik} + (\gamma_i - 1) \ln(1 - \phi_{ik})]$$

$$= \sum_{i=1}^{D} \sum_{k=1}^{K} \ln \frac{\Gamma(\beta_i + \gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} + \sum_{i=1}^{D} \sum_{k=1}^{K} (\beta_i - 1)[\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})]$$

$$+ \sum_{i=1}^{D} \sum_{k=1}^{K} (\gamma_i - 1)[\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \tag{A.3}$$

$$\textcircled{3} = E_q[\ln p(\mathcal{Z}|\boldsymbol{\pi}]$$

$$= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} q(\mathcal{Z}|\boldsymbol{\tau}) q(\boldsymbol{\phi}) q(\boldsymbol{\pi}|\boldsymbol{\rho}) \ln p(\mathcal{Z}|\boldsymbol{\pi}) d\boldsymbol{\phi} d\boldsymbol{\pi}$$

$$= \int_{\boldsymbol{\pi}} \sum_{\mathcal{Z}} q(\mathcal{Z}|\boldsymbol{\tau}) q(\boldsymbol{\pi}|\boldsymbol{\rho}) \ln p(\mathcal{Z}|\boldsymbol{\pi}) d\boldsymbol{\phi} d\boldsymbol{\pi}$$

$$= \int_{\boldsymbol{\pi}} \sum_{\mathcal{Z}} \left[\prod_{\mu'=1}^{N} q(z^{\mu'}|\boldsymbol{\tau}^{\mu'}) \right] q(\boldsymbol{\pi}|\boldsymbol{\rho}) \sum_{\mu=1}^{N} [\ln p(z^{\mu}|\boldsymbol{\pi})] \, d\boldsymbol{\pi}$$

$$= \sum_{\mathcal{Z}} \sum_{\mu=1}^{N} \left[\prod_{\mu'=1}^{N} q(z^{\mu'}|\boldsymbol{\tau}^{\mu'}) \right] \int_{\boldsymbol{\pi}} q(\boldsymbol{\pi}|\boldsymbol{\rho}) \ln \pi_{z^{\mu}} d\boldsymbol{\pi}$$

$$
\begin{aligned}
&= \sum_{\mu=1}^{N} \sum_{\mathcal{Z}} \left[\prod_{\mu'=1}^{N} q(z^{\mu'}|\boldsymbol{\tau}^{\mu'}) \right] [\Psi(\rho_{z^\mu}) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})] \\
&= \sum_{\mu=1}^{N} \sum_{\mathcal{Z}\setminus\{z^\mu\}} \left[\prod_{\mu'\neq\mu}^{N} q(z^{\mu'}|\boldsymbol{\tau}^{\mu'}) \right] \sum_{z^\mu=1}^{K} q(z^\mu|\boldsymbol{\tau}^\mu) [\Psi(\rho_{z^\mu}) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})] \\
&= \sum_{\mu=1}^{N} \sum_{k=1}^{K} \tau_k^\mu [\Psi(\rho_k) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})] \qquad \text{(A.4)}
\end{aligned}
$$

$$
\begin{aligned}
④ &= E_q[\ln p(\mathcal{T}|\mathcal{Z}, \boldsymbol{\phi})] \\
&= \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} q(\mathcal{Z}|\boldsymbol{\tau}) q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) q(\boldsymbol{\pi}) \ln p(\mathcal{T}|\mathcal{Z}, \boldsymbol{\phi}) d\boldsymbol{\phi} d\boldsymbol{\pi} \\
&= \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} \left[\prod_{\mu'=1}^{N} q(z^{\mu'}|\boldsymbol{\tau}^{\mu'}) \right] q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \sum_{\mu=1}^{N} \sum_{i=1}^{D} \ln \phi_{iz^\mu}^{x_i^\mu} (1 - \phi_{iz^\mu})^{(1-x_i^\mu)} d\boldsymbol{\phi} \\
&= \sum_{i=1}^{D} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}} \left[\prod_{\mu'=1}^{N} \tau_{z^{\mu'}}^{\mu'} \right] q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \sum_{\mu=1}^{N} [x_i^\mu \ln \phi_{iz^\mu} + (1 - x_i^\mu) \ln(1 - \phi_{iz^\mu})] d\boldsymbol{\phi} \\
&= \sum_{i=1}^{D} \int_{\boldsymbol{\phi}} \sum_{\mathcal{Z}\setminus\{z^\mu\}} \left[\prod_{\mu'\neq\mu}^{N} \tau_{z^{\mu'}}^{\mu'} \right] \sum_{z^\mu=1}^{K} \tau_{z^\mu}^\mu q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \sum_{\mu=1}^{N} [\sim] d\boldsymbol{\phi} \\
&= \sum_{i=1}^{D} \int_{\boldsymbol{\phi}} \sum_{k=1}^{K} \tau_k^\mu q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \sum_{\mu=1}^{N} [x_i^\mu \ln \phi_{ik} + (1 - x_i^\mu) \ln(1 - \phi_{ik})] d\boldsymbol{\phi} \\
&= \sum_{\mu=1}^{N} \sum_{i=1}^{D} \sum_{k=1}^{K} \tau^\mu \int_{\boldsymbol{\phi}} q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) [x_i^\mu \ln \phi_{ik} + (1 - x_i^\mu) \ln(1 - \phi_{ik})] d\boldsymbol{\phi} \\
&= \sum_{\mu=1}^{N} \sum_{i=1}^{D} \sum_{k=1}^{K} \tau^\mu \{x_i^\mu [\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] + (1 - x_i^\mu) [\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})]\}
\end{aligned}
\qquad \text{(A.5)}
$$

$$
\begin{aligned}
⑤ &= -E_q[\ln q(\boldsymbol{\pi}|\boldsymbol{\rho})] \\
&= -\int_{\boldsymbol{\pi}} q(\boldsymbol{\pi}|\boldsymbol{\rho}) \ln q(\boldsymbol{\pi}|\boldsymbol{\rho}) d\boldsymbol{\pi} \\
&= -\ln \frac{\Gamma(\sum_{k=1}^{K} \rho_k)}{\prod_{k=1}^{K} \Gamma(\rho_k)} - \sum_{k=1}^{K} (\rho_k - 1) [\Psi(\rho_k) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})] \qquad \text{(A.6)}
\end{aligned}
$$

$$
\begin{aligned}
⑥ &= -E_q[\ln q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu})] \\
&= -\int_{\boldsymbol{\phi}} q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) \ln q(\boldsymbol{\phi}|\boldsymbol{\eta}, \boldsymbol{\nu}) d\boldsymbol{\phi}
\end{aligned}
$$

$$= -\sum_{i=1}^{D}\sum_{k=1}^{K}\ln\frac{\Gamma(\eta_{ik}+\nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} - \sum_{i=1}^{D}\sum_{k=1}^{K}(\eta_{ik}-1)[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]$$
$$- \sum_{i=1}^{D}\sum_{k=1}^{K}(\nu_{ik}-1)[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})] \tag{A.7}$$

$$⑦ = -E_q[\ln q(\mathcal{Z}|\boldsymbol{\tau})]$$
$$= -\int_{\boldsymbol{\pi}}\int_{\boldsymbol{\phi}}\sum_{\mathcal{Z}} q(\mathcal{Z}|\boldsymbol{\tau})q(\boldsymbol{\phi})q(\boldsymbol{\pi}|\boldsymbol{\rho})\ln p(\mathcal{Z}|\boldsymbol{\tau})d\boldsymbol{\phi}d\boldsymbol{\pi}$$
$$= -\sum_{\mathcal{Z}} q(\mathcal{Z}|\boldsymbol{\tau})\ln p(\mathcal{Z}|\boldsymbol{\tau})$$
$$= -\sum_{\mathcal{Z}}\left[\prod_{\mu'=1}^{N}\tau^{\mu'}\right]\sum_{\mu=1}^{N}[\ln p(z^{\mu}|\tau^{\mu})]$$
$$= -\sum_{\mathcal{Z}}\sum_{\mu=1}^{N}\left[\prod_{\mu'=1}^{N}\tau^{\mu'}\right]\ln\tau^{\mu}$$
$$= -\sum_{\mu=1}^{N}\sum_{\mathcal{Z}\setminus\{z^{\mu}\}}\left[\prod_{\mu'\neq\mu}^{N}\tau^{\mu'}\right]\sum_{z^{\mu}=1}^{K}\tau^{\mu'}\ln\tau^{\mu}$$
$$= -\sum_{\mu=1}^{N}\sum_{k=1}^{K}\tau_k^{\mu}\ln\tau_k^{\mu} \tag{A.8}$$

We regroup $\mathcal{L}$ by the variational parameters for further optimization.

$$\mathcal{L} = C + f(\boldsymbol{\rho},\boldsymbol{\tau}) + g(\boldsymbol{\eta},\boldsymbol{\nu},\boldsymbol{\tau}) + h(\boldsymbol{\tau}) \tag{A.9}$$

where

$$C = \ln\frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} + K\sum_{i=1}^{D}\ln\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \tag{A.10}$$

$$f(\boldsymbol{\rho},\boldsymbol{\tau}) = \sum_{k=1}^{K}(\frac{\alpha}{K}+\sum_{\mu=1}^{N}\tau_k^{\mu}-\rho_k)[\Psi(\rho_k)-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})]$$
$$- \ln\frac{\Gamma(\sum_{k=1}^{K}\rho_k)}{\prod_{k=1}^{K}\Gamma(\rho_k)} \tag{A.11}$$

$$g(\boldsymbol{\eta},\boldsymbol{\nu},\boldsymbol{\tau}) = \sum_{i=1}^{D}\sum_{k=1}^{K}(\beta_i+\sum_{\mu=1}^{N}\tau^{\mu}x_i^{\mu}-\eta_{ik})[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]$$

$$+\sum_{i=1}^{D}\sum_{k=1}^{K}(\gamma_i+\sum_{\mu=1}^{N}\tau^{\mu}(1-x_i^{\mu})-\nu_{ik})[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})]$$
$$-\sum_{i=1}^{D}\sum_{k=1}^{K}\ln\frac{\Gamma(\eta_{ik}+\nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} \tag{A.12}$$

$$h(\boldsymbol{\tau})=-\sum_{\mu=1}^{N}\sum_{k=1}^{K}\tau_k^{\mu}\ln\tau_k^{\mu} \tag{A.13}$$

A.2 Maximizing $\mathcal{L}$ in Section 3.3

We need to maximize $\mathcal{L}$ with respect to τ_k^{μ}, ρ_k, η_{ik} and ν_{ik}. The constraints are $\sum_{k=1}^{K}\tau_k^{\mu}=1,\forall\mu$.

We pick out all terms of $\mathcal{L}$ containing τ_k^{μ} and write it as $\mathcal{L}_{[\tau_k^{\mu}]}$:

$$\begin{aligned}\mathcal{L}_{[\tau_k^{\mu}]}=&\tau_k^{\mu}[\Psi(\rho_k)-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})]\\&+\tau_k^{\mu}\sum_{i=1}^{D}\{x_i^{\mu}[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]+(1-x_i^{\mu})[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})]\}\\&-\tau_k^{\mu}\ln\tau_k^{\mu}\end{aligned} \tag{A.14}$$

We form the Lagrangian by adding the appropriate Lagrange multipliers $\lambda(\sum_{k=1}^{K}\tau_k^{\mu}-1)$ and take derivatives with respect to τ_k^{μ}, we obtain:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\tau_k^{\mu}}=&\Psi(\rho_k)-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})\\&+\sum_{i=1}^{D}\{x_i^{\mu}[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]+(1-x_i^{\mu})[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})]\}\\&-\ln\tau_k^{\mu}-1+\lambda\end{aligned} \tag{A.15}$$

Setting this derivative to 0, we can solve for τ_k^{μ}:

$$\begin{aligned}\tau_k^{\mu}=&\exp(\lambda-1)\exp\{\Psi(\rho_k)-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})\\&+\sum_{i=1}^{D}\{x_i^{\mu}[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]+(1-x_i^{\mu})[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})]\}\end{aligned} \tag{A.16}$$

which means

$$\tau_k^{\mu}\propto\exp\{\Psi(\rho_k)-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})$$

$$+\sum_{i=1}^{D}\{x_i^\mu[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]+(1-x_i^\mu)[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})]\} \quad \text{(A.17)}$$

Setting an appropriate λ could make the sum of τ_k^μ to be 1. In practice, we calculate τ_k^μs by Equation (A.17) and directly normalize them.

Next, we maximize $\mathcal{L}$ with respect to ρ_k, the kth component of the approximate posterior Dirichlet parameter. The terms containing ρ_k are:

$$\begin{aligned}\mathcal{L}_{[\rho_k]} =&(\frac{\alpha}{K}+\sum_{\mu=1}^{N}\tau_k^\mu-\rho_k)[\Psi(\rho_k)-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})]\\ &-\ln\frac{\Gamma(\sum_{k'=1}^{K}\rho_{k'})}{\Gamma(\rho_k)} \qquad \text{(A.18)}\\ & \qquad \text{(A.19)}\end{aligned}$$

We take derivative with respect to ρ_k:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\rho_k} =&(\frac{\alpha}{K}+\sum_{\mu=1}^{N}\tau_k^\mu-\rho_k)[\Psi'(\rho_k)-\Psi'(\textstyle\sum_{k'=1}^{K}\rho_{k'})]\\ &-[\Psi(\rho_k)-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})]-\Psi(\textstyle\sum_{k'=1}^{K}\rho_{k'})+\Psi(\rho_k)\\ =&(\frac{\alpha}{K}+\sum_{\mu=1}^{N}\tau_k^\mu-\rho_k)[\Psi'(\rho_k)-\Psi'(\textstyle\sum_{k'=1}^{K}\rho_{k'})] \qquad \text{(A.20)}\end{aligned}$$

Setting Equation (A.20) to 0 for all k and solve for ρ_k, we obtain the maximum at:

$$\rho_k=\frac{\alpha}{K}+\sum_{\mu=1}^{N}\tau_k^\mu \qquad \text{(A.21)}$$

Now we maximize $\mathcal{L}$ with respect to η_{ik} and ν_{ik}, the kth component of the approximate posterior Beta parameter for the ith dimension. The terms containing η_{ik} and $\nu_{ik}$are:

$$\begin{aligned}&\mathcal{L}_{[\eta_{ik}]}=\mathcal{L}_{[\nu_{ik}]}\\ =&(\beta_i+\sum_{\mu=1}^{N}\tau^\mu x_i^\mu-\eta_{ik})[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]\\ &+(\gamma_i+\sum_{\mu=1}^{N}\tau^\mu(1-x_i^\mu)-\nu_{ik})[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})]\end{aligned}$$

$$- \ln \frac{\Gamma(\eta_{ik} + \nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} \tag{A.22}$$

Taking derivatives:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \eta_{ik}} =& (\beta_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu - \eta_{ik})[\Psi'(\eta_{ik}) - \Psi'(\eta_{ik} + \nu_{ik})] \\
& - [\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\
& - [\gamma_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu) - \nu_{ik}]\Psi'(\eta_{ik} + \nu_{ik}) \\
& - \Psi(\eta_{ik} + \nu_{ik}) + \Psi(\eta_{ik}) \\
=& (\beta_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu - \eta_{ik})\Psi'(\eta_{ik}) \\
& - (\beta_i + \gamma_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu - \eta_{ik} - \nu_{ik})\Psi'(\eta_{ik} + \nu_{ik})
\end{aligned} \tag{A.23}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \nu_{ik}} =& - (\beta_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu - \eta_{ik})\Psi'(\eta_{ik} + \nu_{ik}) \\
& + [\gamma_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu) - \nu_{ik}][\Psi'(\nu_{ik}) - \Psi'(\eta_{ik} + \nu_{ik})] \\
& - [\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\
& - \Psi(\eta_{ik} + \nu_{ik}) + \Psi(\nu_{ik}) \\
=& [\gamma_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu) - \nu_{ik}]\Psi'(\nu_{ik}) \\
& - (\beta_i + \gamma_i + \textstyle\sum_{\mu=1}^{N} \tau_k^\mu - \eta_{ik} - \nu_{ik})\Psi'(\eta_{ik} + \nu_{ik})
\end{aligned} \tag{A.24}$$

Setting (A.23) and (A.24) to 0 and solve η_{ik} and ν_{ik}:

$$\eta_{ik} = \beta_i + \sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu \tag{A.25}$$

$$\nu_{ik} = \gamma_i + \sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu) \tag{A.26}$$

As a summary, $\mathcal{L}$ is maximized by

$$\tau_k^\mu \propto \exp\Big\{ \Psi(\rho_k) - \Psi(\textstyle\sum_{k'=1}^{K} \rho_{k'})$$

$$+\sum_{i=1}^{D}\Big\{x_i^{\mu}[\Psi(\eta_{ik})-\Psi(\eta_{ik}+\nu_{ik})]+(1-x_i^{\mu})[\Psi(\nu_{ik})-\Psi(\eta_{ik}+\nu_{ik})]\Big\}\Big\} \tag{A.27}$$

$$\rho_k = \frac{\alpha}{K}+\sum_{\mu=1}^{N}\tau_k^{\mu} \tag{A.28}$$

$$\eta_{ik} = \beta_i+\sum_{\mu=1}^{N}\tau_k^{\mu}x_i^{\mu} \tag{A.29}$$

$$\nu_{ik} = \gamma_i+\sum_{\mu=1}^{N}\tau_k^{\mu}(1-x_i^{\mu}) \tag{A.30}$$

A.3 The Computation of $\mathcal{L}$ in Section 4.3

We need to do the computation of the following terms.

$$\begin{aligned}\mathcal{L}(q,\boldsymbol{\Theta}) =& E_q[\ln p(\mathcal{T},\mathcal{Z},\mathbf{v},\boldsymbol{\phi}|\alpha,\boldsymbol{\beta},\boldsymbol{\gamma})]-E_q[\ln q(\mathcal{Z},\mathbf{v},\boldsymbol{\phi})]\\ =& E_q[\ln p(\mathbf{v}|\alpha)]+E_q[\ln p(\boldsymbol{\phi}|\boldsymbol{\beta},\boldsymbol{\gamma})]+E_q[\ln p(\mathcal{Z}|\mathbf{v})]+E_q[\ln p(\mathcal{T}|\mathcal{Z},\boldsymbol{\phi})]\\ &-E_q[\ln q(\mathbf{v}|\boldsymbol{\rho}_1,\boldsymbol{\rho}_2)]-E_q[\ln q(\boldsymbol{\phi}|\boldsymbol{\eta},\boldsymbol{\nu})]-E_q[\ln q(\mathcal{Z}|\boldsymbol{\tau})]\end{aligned} \tag{A.31}$$

The terms $E_q[\ln p(\boldsymbol{\phi}|\boldsymbol{\beta},\boldsymbol{\gamma})]$, $E_q[\ln p(\mathcal{T}|\mathcal{Z},\boldsymbol{\phi})]$, $E_q[\ln q(\boldsymbol{\phi}|\boldsymbol{\eta},\boldsymbol{\nu})]$ and $E_q[\ln q(\mathcal{Z}|\boldsymbol{\tau})]$ are the same as we did in Appendix A.1. The computation of $E_q[\ln p(\mathcal{Z}|\mathbf{v})]$ are in Chapter 4. We only compute $E_q[\ln p(\mathbf{v}|\alpha)]$ and $-E_q[\ln q(\mathbf{v}|\boldsymbol{\rho}_1,\boldsymbol{\rho}_2)]$ and list the rest without calculation.

$$\begin{aligned}E_q[\ln p(\mathbf{v}|\alpha)] =& E_q[\ln\prod_{k=1}^{K-1}p(v_k|\alpha)]\\ =&\sum_{k=1}^{K-1}\int_{v_k}\ln p(v_k|\alpha)q(v_k|\rho_{1k},\rho_{2k})dv_k\\ =&\sum_{k=1}^{K-1}\int_{v_k}\ln\alpha(1-v_t)^{\alpha-1}q(v_k|\rho_{1k},\rho_{2k})dv_k\\ =&\sum_{k=1}^{K-1}\left[\ln\alpha+(\alpha-1)\int_{v_k}(1-v_t)q(v_k|\rho_{1k},\rho_{2k})dv_k\right]\end{aligned}$$

$$= (K-1)\ln\alpha + (\alpha-1)\sum_{k=1}^{K-1}[\Psi(\rho_{2k}) - \Psi(\rho_{1k}+\rho_{2k})] \quad \text{(A.32)}$$

$$E_q[\ln p(\boldsymbol{\phi}|\boldsymbol{\beta},\boldsymbol{\gamma})] = K\sum_{i=1}^{D}\ln\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} + \sum_{i=1}^{D}\sum_{k=1}^{K}(\beta_i-1)[\Psi(\eta_{ik}) - \Psi(\eta_{ik}+\nu_{ik})]$$
$$+ \sum_{i=1}^{D}\sum_{k=1}^{K}(\gamma_i-1)[\Psi(\nu_{ik}) - \Psi(\eta_{ik}+\nu_{ik})] \quad \text{(A.33)}$$

$$E_q[\log p(\mathcal{Z}|\mathbf{v})] = \sum_{\mu=1}^{N}\sum_{k-1}^{K-1}\tau_k^{\mu}(\Psi(\rho_{1k}) - \Psi(\rho_{1k}+\rho_{2k}))$$
$$+ \sum_{\mu=1}^{N}\sum_{k-1}^{K-1}\sum_{t=k+1}^{K}\tau_t^{\mu}(\Psi(\rho_{2k}) - \Psi(\rho_{1k}+\rho_{2k})) \quad \text{(A.34)}$$

$$E_q[\ln p(\mathcal{T}|\mathcal{Z},\boldsymbol{\phi})] = \sum_{\mu=1}^{N}\sum_{i=1}^{D}\sum_{k=1}^{K}\tau^{\mu}x_i^{\mu}[\Psi(\eta_{ik}) - \Psi(\eta_{ik}+\nu_{ik})]$$
$$+ \sum_{\mu=1}^{N}\sum_{i=1}^{D}\sum_{k=1}^{K}\tau^{\mu}(1-x_i^{\mu})[\Psi(\nu_{ik}) - \Psi(\eta_{ik}+\nu_{ik})] \quad \text{(A.35)}$$

$$-E_q[\ln q(\mathbf{v}|\boldsymbol{\rho}_1,\boldsymbol{\rho}_2)] = -\sum_{k=1}^{K-1}E_q[\ln q(v_k|\rho_{1k},\rho_{2k})]$$
$$= -\sum_{k=1}^{K-1}E_q\left[\ln\frac{\Gamma(\rho_{1k}+\rho_{2k})}{\Gamma(\rho_{1k})\Gamma(\rho_{2k})} + (\rho_{1k}-1)\ln v_k + (\rho_{2k}-1)\ln(1-v_k)\right]$$
$$= -\sum_{k=1}^{K-1}\ln\frac{\Gamma(\rho_{1k}+\rho_{2k})}{\Gamma(\rho_{1k})\Gamma(\rho_{2k})} - \sum_{k=1}^{K-1}(\rho_{1k}-1)[\Psi(\rho_{1k}) - \Psi(\rho_{1k}+\rho_{2k})]$$
$$- \sum_{k=1}^{K-1}(\rho_{2k}-1)[\Psi(\rho_{2k}) - \Psi(\rho_{1k}+\rho_{2k})] \quad \text{(A.36)}$$

$$-E_q[\ln q(\boldsymbol{\phi}|\boldsymbol{\eta},\boldsymbol{\nu})] = -\sum_{i=1}^{D}\sum_{k=1}^{K}\ln\frac{\Gamma(\eta_{ik}+\nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} - \sum_{i=1}^{D}\sum_{k=1}^{K}(\eta_{ik}-1)[\Psi(\eta_{ik}) - \Psi(\eta_{ik}+\nu_{ik})]$$
$$- \sum_{i=1}^{D}\sum_{k=1}^{K}(\nu_{ik}-1)[\Psi(\nu_{ik}) - \Psi(\eta_{ik}+\nu_{ik})] \quad \text{(A.37)}$$

$$-E_q[\ln q(\mathcal{Z}|\boldsymbol{\tau})] = -\sum_{\mu=1}^{N}\sum_{k=1}^{K}\tau_k^{\mu}\ln\tau_k^{\mu} \quad \text{(A.38)}$$

This yields:

$$\mathcal{L} = C + f(\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\tau}) + g(\boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{\tau}) + h(\boldsymbol{\tau}) \tag{A.39}$$

where

$$C = (K-1)\ln\alpha + K\sum_{i=1}^{D}\ln\frac{\Gamma(\beta_i+\gamma_i)}{\Gamma(\beta_i)\Gamma(\gamma_i)} \tag{A.40}$$

$$\begin{aligned} f(\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \boldsymbol{\tau}) = &\sum_{k-1}^{K-1}(\sum_{\mu=1}^{N}\tau_k^\mu - \rho_{1k} + 1)[\Psi(\rho_{1k}) - \Psi(\rho_{1k}+\rho_{2k})] \\ &+ \sum_{k=1}^{K-1}(\alpha + \sum_{\mu=1}^{N}\sum_{t=k+1}^{K}\tau_t^\mu - \rho_{2k})[\Psi(\rho_{2k}) - \Psi(\rho_{1k}+\rho_{2k})] \\ &- \sum_{k=1}^{K-1}\ln\frac{\Gamma(\rho_{1k}+\rho_{2k})}{\Gamma(\rho_{1k})\Gamma(\rho_{2k})} \end{aligned} \tag{A.41}$$

$$\begin{aligned} g(\boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{\tau}) = &\sum_{i=1}^{D}\sum_{k=1}^{K}(\beta_i + \sum_{\mu=1}^{N}\tau_k^\mu x_i^\mu - \eta_{ik})[\Psi(\eta_{ik}) - \Psi(\eta_{ik}+\nu_{ik})] \\ &+ \sum_{i=1}^{D}\sum_{k=1}^{K}(\gamma_i + \sum_{\mu=1}^{N}\tau_k^\mu(1-x_i^\mu) - \nu_{ik})[\Psi(\nu_{ik}) - \Psi(\eta_{ik}+\nu_{ik})] \\ &- \sum_{i=1}^{D}\sum_{k=1}^{K}\ln\frac{\Gamma(\eta_{ik}+\nu_{ik})}{\Gamma(\eta_{ik})\Gamma(\nu_{ik})} \end{aligned} \tag{A.42}$$

$$h(\boldsymbol{\tau}) = -\sum_{\mu=1}^{N}\sum_{k=1}^{K}\tau_k^\mu\ln\tau_k^\mu \tag{A.43}$$

A.4 Maximizing $\mathcal{L}$ in Section 4.3

In this section we maximize $\mathcal{L}$ with respect to τ_k^μ, ρ_k, η_{ik} and ν_{ik}. The constraints are $\sum_{k=1}^{K}\tau_k^\mu = 1, \forall\mu$.

Picking out all the terms containing τ_k^μ, we have:

$$\begin{aligned} \mathcal{L}_{[\tau_k^\mu]} = &\tau_k^\mu[\Psi(\rho_{1k}) - \Psi(\rho_{1k}+\rho_{2k})] + \tau_k^\mu\sum_{k'=1}^{k-1}[\Psi(\rho_{2k'}) - \Psi(\rho_{1k'}+\rho_{2k'})] \\ &+ \tau_k^\mu\sum_{i=1}^{D}x_i^\mu[\Psi(\eta_{ik}) - \Psi(\eta_{ik}+\nu_{ik})] \end{aligned}$$

$$+ \tau_k^\mu \sum_{i=1}^{D} (1 - x_i^\mu)[\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] - \tau_k^\mu \ln \tau_k^\mu \tag{A.44}$$

We form the Lagrangian by taking the constraint $\sum_{k=1}^{K} \tau_k^\mu = 1$ into $\mathcal{L}_{[\tau_k^\mu]}$ and take the derivative of $\mathcal{L}_{[\tau_k^\mu]}$ with respect to τ_k^μ:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \tau_k^\mu} =& [\Psi(\rho_{1k}) - \Psi(\rho_{1k} + \rho_{2k})] + \sum_{k'=1}^{k-1} [\Psi(\rho_{2k'}) - \Psi(\rho_{1k'} + \rho_{2k'})] \\ &+ \sum_{i=1}^{D} \{x_i^\mu [\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] + (1 - x_i^\mu)[\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})]\} \\ &- \ln \tau_k^\mu - 1 + \lambda \end{aligned} \tag{A.45}$$

Letting Equation (A.45) to be 0, we can solve for τ_k^μ. With proper λ, we have

$$\begin{aligned}\tau_k^\mu \propto \exp \Bigg\{ & \Psi(\rho_{1k}) + \sum_{k'=1}^{k-1} \Psi(\rho_{2k'}) - \sum_{k'=1}^{k} \Psi(\rho_{1k'} + \rho_{2k'}) \\ & + \sum_{i=1}^{D} x_i^\mu [\Psi(\eta_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \\ & + \sum_{i=1}^{D} (1 - x_i^\mu)[\Psi(\nu_{ik}) - \Psi(\eta_{ik} + \nu_{ik})] \Bigg\} \end{aligned} \tag{A.46}$$

Then we maximize $\mathcal{L}$ with respect to ρ_{1k} and ρ_{2k}.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \rho_{1k}} =& (\sum_{\mu=1}^{N} \tau_k^\mu - \rho_{1k} + 1)[\Psi'(\rho_{1k}) - \Psi'(\rho_{1k} + \rho_{2k})] - [\Psi(\rho_{1k}) - \Psi(\rho_{1k} + \rho_{2k})] \\ &+ (\alpha + \sum_{\mu=1}^{N} \sum_{t=k+1}^{K} \tau_t^\mu - \rho_{2k})[-\Psi'(\rho_{1k} + \rho_{2k})] - [\Psi(\rho_{1k} + \rho_{2k}) - \Psi(\rho_{1k})] \\ =& (\sum_{\mu=1}^{N} \tau_k^\mu - \rho_{1k} + 1)[\Psi'(\rho_{1k}) - \Psi'(\rho_{1k} + \rho_{2k})] \\ &+ (\alpha + \sum_{\mu=1}^{N} \sum_{t=k+1}^{K} \tau_t^\mu - \rho_{2k})[-\Psi'(\rho_{1k} + \rho_{2k})] \end{aligned} \tag{A.47}$$

$$\frac{\partial \mathcal{L}}{\partial \rho_{2k}} = (\sum_{\mu=1}^{N} \tau_k^\mu - \rho_{1k} + 1)[-\Psi'(\rho_{1k} + \rho_{2k})]$$

$$
\begin{aligned}
&+ (\alpha + \sum_{\mu=1}^{N} \sum_{t=k+1}^{K} \tau_t^\mu - \rho_{2k})[\Psi'(\rho_{2k} - \Psi'(\rho_{1k} + \rho_{2k})] \\
&- [\Psi(\rho_{2k}) - \Psi(\rho_{1k} + \rho_{2k})] - [\Psi(\rho_{1k} + \rho_{2k}) - \Psi(\rho_{2k})] \\
=&(\sum_{\mu=1}^{N} \tau_k^\mu - \rho_{1k} + 1)[-\Psi'(\rho_{1k} + \rho_{2k})] \\
&+ (\alpha + \sum_{\mu=1}^{N} \sum_{t=k+1}^{K} \tau_t^\mu - \rho_{2k})[\Psi'(\rho_{2k}) - \Psi'(\rho_{1k} + \rho_{2k})] \qquad \text{(A.48)}
\end{aligned}
$$

We want to solve for ρ_{1k} and ρ_{2k} when Equation (A.47) and (A.48) are 0. An obvious solution is

$$\rho_{1k} = \sum_{\mu=1}^{N} \tau_k^\mu + 1 \qquad \text{(A.49)}$$

$$\rho_{2k} = \sum_{\mu=1}^{N} \sum_{t=k+1}^{K} \tau_t^\mu + \alpha \qquad \text{(A.50)}$$

Notice that this solution is only for $k = 1 \ldots K-1$. For $k = K$, ρ_{1K} and ρ_{2K} are fixed to 1 and 0 because of the truncation assumption. The maximization of η_{ik} and ν_{ik} are the same as we did in section A.2. We list the result here:

$$\eta_{ik} = \beta + \sum_{\mu=1}^{N} \tau_k^\mu x_i^\mu \qquad \text{(A.51)}$$

$$\nu_{ik} = \gamma + \sum_{\mu=1}^{N} \tau_k^\mu (1 - x_i^\mu) \qquad \text{(A.52)}$$

Bibliography

[AGM04] Foto Afrati, Aristides Gionis, and Heikki Mannila. Approximating a collection of frequent sets. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04, pages 12–19, New York, NY, USA, 2004. ACM.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. SIGMOD Rec., 22:207–216, June 1993.

[Ant74] Charles E. Antoniak. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. The Annals of Statistics, 2(6):1152–1174, 1974.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In Data Engineering, 1995. Proceedings of the Eleventh International Conference on, pages 3 –14, mar 1995.

[AY98] Charu C. Aggarwal and Philip S. Yu. A new framework for itemset generation. In Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, PODS '98, pages 18–24, New York, NY, USA, 1998. ACM.

[Bay98] Roberto J. Bayardo, Jr. Efficiently mining long patterns from databases. SIGMOD Rec., 27:85–93, June 1998.

[BCG01] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In In ICDE, pages 443–452, 2001.

[Bea03] Matthew J. Beal. Variational Algorithms for Approximate Bayesian Inference. PhD thesis, University of London, May 2003.

[Bis07] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[BJ05] David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. Bayesian Analysis, 1:121–144, 2005.

[Bla73] D. Blackwell. Discreteness of Ferguson selections. The Annals of Statistics, 1:356–358, 1973.

[BM73] David Blackwell and James B. MacQueen. Ferguson Distributions Via Polya Urn Schemes. The Annals of Statistics, 1(2):353–355, 1973.

[BMS97] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: generalizing association rules to correlations. SIGMOD Rec., 26:265–276, June 1997.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In Proceedings of the 1997 ACM SIGMOD international conference on Management of data, SIGMOD '97, pages 255–264, New York, NY, USA, 1997. ACM.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, March 2003.

[BW71] Jr. Boas, R. P. and Jr. Wrench, J. W. Partial sums of the harmonic series. The American Mathematical Monthly, 78(8):pp. 864–870, 1971.

[CG02] Toon Calders and Bart Goethals. Mining all non-derivable frequent itemsets. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, Principles of Data Mining and Knowledge Discovery, volume

2431 of Lecture Notes in Computer Science, pages 1–42. Springer Berlin / Heidelberg, 2002. 10.1007/3-540-45681-3-7.

[CG05] Toon Calders and Bart Goethals. Depth-first non-derivable itemset mining. In In Proc. SIAM Int. Conf. on Data Mining SDM'05, page 2007, 2005.

[CL68] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. Information Theory, IEEE Transactions on, 14(3):462 – 467, may 1968.

[Coe03] F. Coenen. The LUCS-KDD Discretised/normalised ARM and CARM Data Library. `http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/`, 2003. Department of Computer Science, The University of Liverpool, UK.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 39(1):pp. 1–38, 1977.

[EH81] Brian. Everitt and D. J. Hand. Finite mixture distributions / B.S. Everitt and D.J. Hand. Chapman and Hall, London ; New York :, 1981.

[FA10] A. Frank and A. Asuncion. UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`, 2010. University of California, Irvine, School of Information and Computer Sciences.

[Fer73] Thomas S. Ferguson. A bayesian analysis of some nonparametric problems. The Annals of Statistics, 1(2):pp. 209–230, 1973.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(6):721–741, November 1984.

[GLSS06] Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava, editors. Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA. SIAM, 2006.

[Goe03] Bart Goethals. Survey on frequent pattern mining. Technical report, Helsinki Institute for Information Technolog, 2003.

[GWBV03] Carolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling high frequency accident locations using association rules. In proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA), January 12-16, page 18pp, 2003.

[GZ03] Gösta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets, 2003.

[Has70] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. Biometrika, 57(1):97–109, April 1970.

[HCXY07] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery, 15:55–86, 2007. 10.1007/s10618-006-0059-1.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. SIGMOD Rec., 29:1–12, May 2000.

[Jar04] Szymon Jaroszewicz. Interestingness of frequent itemsets using bayesian networks as background knowledge. In In Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining, pages 178–186. ACM Press, 2004.

[JK77] N. L. Johnson and S. Kotz. Urn Models and Their Applications: An Approach to Modern Discrete Probability Theory. Wiley, New York, 1977.

[JS05] Szymon Jaroszewicz and Tobias Scheffer. Fast discovery of unexpected patterns in data, relative to a bayesian network. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, KDD '05, pages 118–127, New York, NY, USA, 2005. ACM.

[Kru56] Jr. Kruskal, Joseph B. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society, 7(1):pp. 48–50, 1956.

[KWT07] K. Kurihara, M. Welling, and Y. W. Teh. Collapsed variational Dirichlet process mixture models. In Proceedings of the International Joint Conference on Artificial Intelligence, volume 20, 2007.

[LLLY03] Guimei Liu, Hongjun Lu, Wenwu Lou, and Jeffrey Xu Yu. On computing, storing and querying frequent patterns. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, pages 607–612, New York, NY, USA, 2003. ACM.

[LLWH06] Guimei Liu, Jinyan Li, Limsoon Wong, and Wynne Hsu. Positive borders or negative borders: How to make lossless generator based representations concise. In Ghosh et al. [GLSS06].

[Mac94] Steven N. Maceachern. Estimating normal means with a conjugate style dirichlet process prior. Communications In Statistics - Simulation and Computation, 23(3):727–741, 1994.

[Mac96] International Business Machine. IBM intelligent miner users guide, version 1, release 1, 1996.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics, 21(6):1087–1092, 1953.

[MU49] Nicholas Metropolis and S. Ulam. The monte carlo method. Journal of the American Statistical Association, 44(247):pp. 335–341, 1949.

[Omi03] Edward R. Omiecinski. Alternative interest measures for mining associations in databases. IEEE Transactions on Knowledge and Data Engineering, 15:57–69, 2003.

[PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In Catriel Beeri and Peter Buneman, editors, Database Theory ICDT99, volume 1540 of Lecture Notes in Computer Science, pages 398–416. Springer Berlin / Heidelberg, 1999.

[PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In Proceedings of the 1995 ACM SIGMOD international conference on Management of data, SIGMOD '95, pages 175–186, New York, NY, USA, 1995. ACM.

[PDZH02] Jian Pei, Guozhu Dong, Wei Zou, and Jiawei Han. On computing condensed frequent pattern bases. In Data Mining, 2002. ICDM 2002. Proceedings. 2002 IEEE International Conference on, pages 378 – 385, 2002.

[Pea88] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., 1988.

[PHM00] Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In SIGMOD Intl Workshop on Data Mining and Knowledge Discover, pages 21–30, 2000.

[PHMA+01] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. Data Engineering, International Conference on, 0:0215, 2001.

[PMS03] Dmitry Pavlov, Heikki Mannila, and Padhraic Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. IEEE Transactions on Knowledge and Data Engineering, 15:1409–1421, 2003.

[PS91] G. Piatetsky-Shapiro. Discovery, Analysis, and Presentation of Strong Rules, pages 229–248. AAAI/MIT Press, Cambridge, MA, 1991.

[RJBAG99] Jr. Roberto J. Bayardo, Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. Data Engineering, International Conference on, 0:188, 1999.

[RMZ03] Ganesh Ramesh, William A. Maniatty, and Mohammed J. Zaki. Feasible itemset distributions in data mining: theory and application. In Proceedings of the twenty-second ACM

SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '03, pages 284–295, New York, NY, USA, 2003. ACM.

[Set94] J. Sethuraman. A constructive definition of Dirichlet priors. Statistica Sinica, 4:639–650, 1994.

[SON95] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In Proceedings of the 21th International Conference on Very Large Data Bases, VLDB '95, pages 432–444, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[Tat08] Nikolaj Tatti. Maximum entropy based significance of itemsets. Knowl. Inf. Syst., 17:57–77, October 2008.

[TJBB04] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical dirichlet processes. Journal of the American Statistical Association, 101, 2004.

[TM10] Nikolaj Tatti and Michael Mampaey. Using background knowledge to rank itemsets. Data Min. Knowl. Discov., 21:293–309, September 2010.

[Toi96] Hannu Toivonen. Sampling large databases for association rules. In Proceedings of the 22th International Conference on Very Large Data Bases, VLDB '96, pages 134–145, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[WD10] Lianming Wang and David B. Dunson. Fast bayesian inference in dirichlet process mixture models. Journal of Computational and Graphical Statistics, 2010.

[Web07] Geoffrey I. Webb. Discovering significant patterns. Mach. Learn., 68:1–33, July 2007.

[WHM+05] Jianyong Wang, Jiawei Han, Senior Member, Ying Lu, and Petre Tzvetkov. Tfp: An efficient algorithm for mining top-k frequent closed itemsets. IEEE Trans. on Knowledge and Data Engineering, 17:2005, 2005.

[WM03] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. SIGKDD Explor. Newsl., 5:59–68, July 2003.

[WP06] Chao Wang and Srinivasan Parthasarathy. Summarizing itemset patterns using probabilistic models. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '06, pages 730–735, New York, NY, USA, 2006. ACM.

[WTK08] M. Welling, Y. W. Teh, and H. J. Kappen. Hybrid Variational/Gibbs collapsed inference in topic models. In Proceedings of the International Conference on Uncertainty in Artificial Intelligence, volume 24, 2008.

[XHYC05] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining compressed frequent-pattern sets. In Proceedings of the 31st international conference on Very large data bases, VLDB '05, pages 709–720. VLDB Endowment, 2005.

[Yan04] Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04, pages 344–353, New York, NY, USA, 2004. ACM.

[YCHX05] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: a profile-based approach. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, KDD '05, pages 314–323, New York, NY, USA, 2005. ACM.

[Zak00] M.J. Zaki. Scalable algorithms for association mining. Knowledge and Data Engineering, IEEE Transactions on, 12(3):372 –390, may/jun 2000.

[ZG03] Mohammed J. Zaki and Karam Gouda. Fast vertical mining using diffsets. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, pages 326–335, New York, NY, USA, 2003. ACM.

[ZjH02] Mohammed J. Zaki and Ching jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In SDM'02, pages 457–473, 2002.