January 2016

GENERIC FRAMEWORKS FOR INTERACTIVE PERSONALIZED INTERESTING PATTERN DISCOVERY

Md Mansurul Alam Bhuiyan
Purdue University

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By BHUIYAN, MD MANSURUL, ALAM

Entitled

GENERIC FRAMEWORKS FOR INTERACTIVE PERSONALIZED INTERESTING PATTERN DISCOVERY

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

MOHAMMAD AL HASAN Chair	ELISA BERTINO
SNEHASIS MUKHOPADHYAY	JEAN HONORIO
CHRIS CLIFTON	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): MOHAMMAD AL HASAN

Approved by: SUNIL PRABHAKAR/WILLIAM J. GORMAN 9/30/2016

Head of the Departmental Graduate Program Date

GENERIC FRAMEWORKS FOR INTERACTIVE PERSONALIZED INTERESTING PATTERN DISCOVERY

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Md Mansurul Bhuiyan

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2016

Purdue University

West Lafayette, Indiana

To my parents and wife; without whom none of these would be possible.

ACKNOWLEDGMENTS

I can not acknowledge enough to my PhD advisor Dr. Mohammad Hasan. Six years back when I was fresh from the boat, Dr. Hasan took my responsibility. He understood my shortcomings and recognized my eagerness to learn. He taught me so many things that became part of my everyday work. I thank him from the bottom of my heart. I would also like to thank Dr. Chris Clifton, Dr. Snehasis Mukhopadhyay, Dr. Elisa Burtino, Dr. Jean Honorio and Dr. David Gleich for their guidance, encouragement and suggestions. I appreciate the friendship and research contributions from all the members of our lab and would like to take this opportunity to thank Mahmud, Tanay, Baichuan and Vachik. Lastly, I am thankful to my all family members for their support and encouragement.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Bhuiyan, Md Mansurul Ph.D., Purdue University, December 2016. Generic Frameworks for Interactive Personalized Interesting Pattern Discovery. Major Professor: Mohammad Al Hasan.

The traditional frequent pattern mining algorithms generate an exponentially large number of patterns of which a substantial portion are not much significant for many data analysis endeavors. Due to this, the discovery of a small number of interesting patterns from the exponentially large number of frequent patterns according to a particular user's interest is an important task. Existing works on pattern summarization compute a summary that globally represents the entire frequent pattern-set. Such a representative set solves the interesting pattern discovery problem from a global perspective which, more often, far from the personalization that is required to fulfill the pattern discovery criteria of a particular user. In this dissertation, we propose two interactive pattern discovery frameworks to identify a set of interesting patterns for a particular user without requiring any prior input on the measure of interest of patterns from the user. The proposed frameworks are generic to support discovery of the interesting set, sequence and graph type patterns.

In the first framework, we solve the problem by proposing a novel solution which is based on Markov Chain Monte Carlo (MCMC) sampling of patterns. Our solution allows interactive sampling so that the sampled patterns can fulfill the user's requirement effectively. Instead of returning all the patterns for feedback, the proposed paradigm sends back a small set of randomly selected patterns so that an adversary will not be able to reconstruct the entire dataset with higher accuracy using the released set of patterns, hence protecting the confidentiality of the data-set as a whole.

This feature enables the proposed framework to mine interesting patterns from hidden datasets.

In the second framework, we develop an interactive pattern discovery framework called PRIIME that is based on iterative learning of a user's interestingness function. The learning process is supervised where the supervision is provided by a user through feedback on a small set of pattern. Depending on the nature of feedback i.e. non-negative real valued or discrete, in PRIIME we develop gradient boosted tree-based regression and softmax classification algorithms that use a limited number of interactive feedbacks from the user to learn the interestingness profile of the user, and use this profile for pattern recommendation.

Both the proposed frameworks are generic in nature. First proposed framework discover patterns by commencing a random walk over a pattern space so the framework is inherently generic. However, in PRIIME, the interactive pattern discovery is performed by modeling a traditional regression/classification function, for which we need a vector representation of pattern instances. For vector representation of set patterns we use bag-of-words based model. However, for graph and sequential pattern we propose a neural net (NN) based unsupervised feature construction approach.

In an interactive pattern discovery system, effective nomination of patterns for feedback to train a learning model is an important task. In both the proposed frameworks, we develop efficient strategy that combine exploration and exploitation to select patterns for feedback. We show experimental results on several real-life datasets to validate the performance of the proposed frameworks. We also compare with the existing methods of interactive pattern discovery to show that our methods are substantially superior in performance.

Finally, to portray the applicability of the interactive pattern discovery, we build a new home discovery tool for home buyers called RAVEN. It uses interactive feedback over a collection of home feature-sets to learn a buyer's interestingness profile. Then it recommends a small list of homes that match with the buyer's interest, eventually decreasing the time interval between home search initiation and purchase.

1. INTRODUCTION

Frequent pattern (itemsets, sequences, or graphs) mining [1] has been a core research task in the data mining domain for over two decades, yet its deployment in real life data analysis has been moderate due to the following two challenges: (1) the pattern space is combinatorial, so the number of patterns that a mining task produces is generally too large to process by an end user, (2) for various data analysis tasks, frequency threshold is not a sufficient filtering criterion for selecting patterns. To overcome (1), many works have been proposed for pattern summarization and compression [2–6], and to overcome (2), several alternative *interestingness* metrics, such as Jaccard index, odds ratio, and lift [1, 7, 8] have been proposed. Nevertheless, the discovery of interesting patterns remains an unsolved problem due to the subjectivity in the definition of *interestingness*. In many cases, off-the-shelf interestingness metrics does not represent true interestingness for a specific user over the patterns. Pattern summarization does not help either, as such an approach solves interesting pattern discovery from a global perspective which is far from personalization that is needed to meet the pattern discovery demand of a specific user.

There exist few works which target personalized pattern discovery by using users feedback [9, 10]. The overall methodologies of these works are to build an interactive pattern discovery system, that works as the following—a user provides ratings on a small collection of patterns, then the system uses these ratings to train a personalization model, which is later used to isolate user's preferred patterns from the remaining ones. The major design choices of an interactive pattern discovery system are: (1) learning model that the system uses; (2) pattern selection process for feedback—whether it is active [11] or non-active; (3) feature representation of patterns for facilitating learning; and (4) genericness, i.e. the kinds of patterns that the system supports. Existing solutions [9, 10] for interactive pattern discovery are not

effective for a number of the above design choices. In this dissertation, we address the challenges associated with each of the above design choices to build an efficient, personalized pattern discovery system.

1.1 Problem Definition of Interactive Personalized Interesting Pattern Discovery

Consider a transactional dataset $\mathcal{D}$, where each transaction in $\mathcal{D}$ is a combinatorial object, such as an itemset, a sequence, or a graph. Depending on the objects that it contains, the dataset $\mathcal{D}$ can be an itemset, a sequence or a graph dataset. A user u is interested in mining interesting patterns from $\mathcal{D}$, where the patterns (denoted by set $\mathcal{O}$) are sub-objects, say subsets, subsequences, or subgraphs, over the objects in $\mathcal{D}$. Traditional approaches [12, 13] consider frequency as the interestingness metric and design frequent pattern mining algorithms which return patterns that exceed minimum support thresholds over the transactions in $\mathcal{D}$. However, due to the fact that the frequent pattern space is combinatorial [1], existing frequent pattern mining algorithms generally return a large number of patterns, causing an information overload. Interactive pattern discovery is a framework for negotiating information overload so that a succinct set of interesting patterns (a small subset of frequent patterns, $\mathcal{O}$) can be delivered to u, which are personalized by utilizing u's feedback on a small number of patterns. We call this framework IPIPD, which stands for Interactive Personalized Interesting Pattern Discovery.

We assume u possesses an interestingness function (say, f) over the patterns ($\mathcal{O}$). f maps each pattern in $\mathcal{O}$ to a score indicating u's interest level for the pattern, i.e., $f : \mathcal{O} \rightarrow Y$. Here, Y can be modeled as discrete or non-negative real number. IPIPD framework learns f through an iterative user's feedback session. During an iteration (say i), the system returns a set of l patterns $\{p_t\}_{1 \leq t \leq l}$ to u, which is selected from $\mathcal{O}$; u sends feedback $\{y_t = f(p_t)\}_{1 \leq t \leq l}$ using an interactive console; y_t's are discrete or non-negative real number reflecting the user's empathy towards the selected patterns;

using the feedback the system updates its current model of the user's interestingness function, f.

In the following sections, we will explain our contributions addressing the four design choices: i) interactive learning model, ii) pattern selection process for feedback, iii) effective feature representation of pattern, and iv) genericness associated with an efficient personalized interactive interesting pattern discovery system. Contribution of this dissertation is also demonstrate through a real-life application of interactive pattern discovery framework.

1.2 Interactive Learning Model

Learning the user's interestingness criteria (function f) is at the core of personalized pattern discovery problem. The task of identifying a learning model and training the model for effective pattern recommendation is the most prioritized job of an IPIPD system. In this dissertation, we model the learning of a user's interestingness function, f in two ways. In one, we design it as a sampling mechanism that interactively modifies the sampling distribution (proportional to the interestingness function) according to the user's feedback (y_t). In second, we design the problem as supervised learning where we adopt both regression and classification for learning the model iteratively while leveraging the user's feedback. In the following, we first discuss the user interaction design that we use in this work and then we explain both of the learning models in details.

1.2.1 User Interaction Design

The applicability of the IPIPD framework in a real life setting depends strongly on the way a user interacts with the system. Ideally, level of interaction between the system and the user should be simple and the system should put manageable burden to the user while collecting feedback. Existing systems fail to provide above feature in their interactive mechanism. In [9], the user has to provide a complete ordering

of a set of patterns for feedback. Since the user has to cross check among patterns in the set to formulate the feedback, the chance of making mistakes is high. This misinformation directly affects the learning model. In [10], apart from providing the ordering, the user has to decide whether the system chooses between exploitation or exploration. Such interaction protocol puts more burden on a user in the sense that the quality of the learned model is directly influenced by the user's ability to provide precise feedback. From the very beginning, we were motivated to design an interaction mechanism that is simple as well as effective. With that in mind, in our proposed IPIPD frameworks, a user only considers one pattern at a time when she formulates a feedback. The user considers a notion of an interestingness score, either discrete or positive real-valued, to construct a feedback on the patterns quality.

1.2.2 Interactive Pattern Discovery though Sampling

In [14], we propose a novel sampling-based interactive pattern discovery method where the user interacts with a Metropolis-Hastings (MH) based Markov Chain Monte Carlo (MCMC) sampling entity. It samples patterns from the data; the target distribution of the sampler is iteratively refined based on binary feedback (discrete y_t) from the user. We model the user's interestingness function (f) over the patterns using the target sampling distribution. As an additional feature, this sampling based approach protects the integrity of a hidden dataset by releasing patterns in a restrictive manner such that the probability of entire dataset reconstruction using the released set of patterns with high accuracy is very low [15]. In Chapter 3, we discuss in detail of the proposed algorithm and show experimental results from several real-life data sets to validate the capability and usefulness of our solution.

1.2.3 Interactive Pattern Discovery through Supervised Learning

Even though the overall idea is novel, the sampling-based solution has a crucial drawback. The learning and the sampling are unified, which causes the sampler to

converge to a sharp distribution prematurely. As a consequence, this sampler may sample patterns only from a local region of the frequent pattern space while keeping a significant amount of space unexplored. We solve the drawback by modeling the interestingness function as a linear predictor function. Depending on the nature of feedback, we either use a regression (positive real-valued feedback) or a classification (discrete values) algorithm.

Classification

In the classification approach, interestingness function f can be defined as $f : \mathcal{O} \rightarrow \mathbb{C}$; i.e. f maps each pattern in $\mathcal{O}$ to a discrete number (class label). We build a softmax classification based iterative learning algorithm that leverages a limited number of interactive feedback from the user to learn the interestingness profile of the user and uses this profile for pattern recommendation. In Chapter 5 we discuss the model in detail.

Regression

In the regression approach, interestingness function f can be defined as $f : \mathcal{O} \rightarrow \mathbb{R}_+$; i.e. f maps each pattern in $\mathcal{O}$ to a non-negative real number. We develop a gradient boosted regression tree (an ensemble of high bias regression trees) based iterative learning algorithm that uses a limited number of interactive feedback from the user to learn her interestingness profile of the patterns and uses this profile for pattern recommendation. In Chapter 5 we elaborately discuss the model.

1.3 Selecting Patterns for Feedback

For any interactive learning platform, designing a method for selecting a small set of patterns for which a user's feedback is sought is critical. Given that the number of frequent patterns is typically enormous, the performance of interactive learning

depends critically on the module that selects the patterns for collecting user's feedback. In the sampling-based approach, we combine model learning with the feedback set selection. First, we [14] adopt a step function to nominate patterns for feedback. Later in [15], we upgrade the pattern selection process by providing the system a capability to measure whether the information obtained through feedback is significant for learning or not. One major flaw with the heavy coupling between the learning and selection is that the initial sets of feedbacks play a vital role to the model's configuration. Hence, the model may risk suffering from the *positive reinforcement* phenomenon. It sometimes makes the predictive model representing some areas of the pattern space, making it sub-optimal over the entire space. To resolve this issue in the supervised learning setup, we reduce the coupling between the existing state of the learning model and the feedback pattern selection which yields a much improved learning model. We effectively combine exploration and exploitation which reduces the bias that the existing learning model imposes on pattern selection.

1.4 Feature Representation of Patterns and Genericness

The majority of the existing platforms for interactive pattern discovery only support itemset patterns because they do not have an effective metric embedding for more complex patterns, say, sequences, or graphs. Metric embedding of patterns is needed for facilitating the training of a model which discriminates between interesting and not-so-interesting patterns. For itemset patterns, existing works leverage bag of items for instrumenting a metric representation. However, no such natural instrumentation is available for complex patterns, such as a sequence, or a graph. Sometimes, n-grams are used for metric embedding of sequences [16], and topological measures, such as centralities, eccentricity, egonet degree, egonet size, and diameter are used for feature representation of graphs [17]. While the above feature representation may work well for the task of traditional sequence or graph classification, they do not work well for frequent patterns which are numerous and much smaller in size.

In [14], we used a bag of items/edges based technique to find feature representation of set and graph type patterns. In [15] for graph patterns, we improve the representation by considering topological information of the graph in the bag in terms of 3 and 4 size induced subgraphs in the pattern. Such crude representation for a complex pattern like the graph is unable to produce a satisfactory performance. As an alternative, we seek a solution to build a generic IPIPD framework using the unsupervised feature learning in traditional and neural network domain. These methods help an analyst discover features automatically, thus obviating feature engineering using domain knowledge. We develop a technique called "Pattern2Vec" that computes the metric representation of graph and sequence patterns using traditional single layer neural network. "Pattern2Vec" map each pattern into a sentence and the pattern elements (events in a sequence or edges in a graph) into words. Afterward, it leverages a language model similar to [18] for finding d-dimensional feature vector representation of the patterns. We experimentally show that such a technique performs better than the existing ad-hoc metric embedding.

1.5 Real-life Application of Interactive Personalized Interesting Pattern Discovery

To demonstrate the applicability of interactive personalized interesting pattern discovery in a real world setting, we perform a case study in the real-estate domain. Generally, the success of a home searching process depends on a user's ability to construct a well-defined search query. It also depends on her patience while going through the hundreds of houses that are returned by the search engine. This assessment is particularly challenging for new home buyers, who are sometimes not even familiar with all different home features. The overall process of selecting the right home takes time; for 40% of first-time home buyers, the lag time between research and action (buying a home) is around 120 days [19]. Given that home buying is a significant investment, most home buyers take help from an experienced real estate

agent who can read the buyer's mind as soon as possible and show her the home that is just the right one for her.

Our developed IPIPD system named RAVEN in this setup works just like a virtual real estate agent for the home buyers, where each house can be thought of as a set type transaction and house features as the items in the transaction. In each iteration, RAVEN presents a set of patterns which are a summary set of features of the houses in the data. By utilizing the feedback over the quality of patterns, RAVEN gradually learns the user's interestingness criteria on the house features. Finally, using the learned model, RAVEN identifies the house summaries (patterns) as well as the houses that the user will prefer.

1.6 Organization of the Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we discuss background on traditional frequent pattern mining and related works from pattern summarization and interactive pattern discovery domain. In Chapter 3, we discuss the sampling based approach of personalized interesting pattern discovery, work published in the Information and Knowledge Management (CIKM) conference, 2012 [14]. In Statistical Analysis and Data Mining journal we further published an extended version of this work [15]. In Chapter 4, we discuss "Pattern2Vec" approach for computing effective feature vector representation of patterns that we leverage to build generic interactive pattern discovery framework. In Chapter 5, we discuss the softmax classification and gradient boosted regression tree based IPIPD system called PRIIME. We submitted PRIIME in the IEEE Big Data conference, 2016. In Chapter 6, we present a web-based smart house discovery tool called RAVEN, that is build upon the concept of personalization interesting pattern discovery. We submitted RAVEN in Web Search and Data Mining (WSDM) conference, 2017. We conclude this dissertation with a conclusion and future works (Chapter 7).

2. BACKGROUND AND RELATED WORKS

In this chapter, we will discuss background of the frequent pattern mining and related work under four different categories: frequent pattern summarizations, interestingness measure of patterns, interactive knowledge discovery in general, and interactive pattern mining.

2.1 Background

This section contains background knowledge on frequent pattern mining.

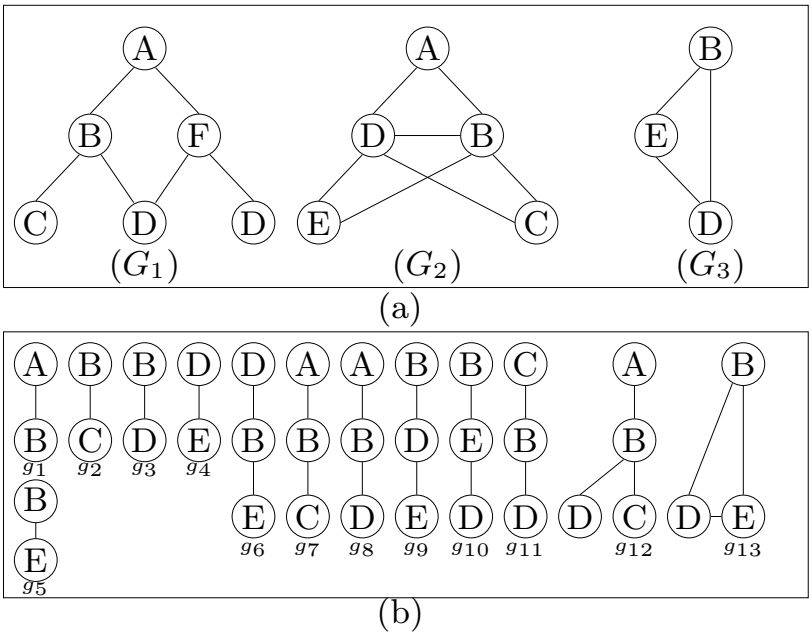


Fig. 2.1. (a) Graph database with 3 transaction graphs (b) Frequent subgraph of (a) with $minsup = 2$

T_1	ABDE
T_2	BCE
T_3	ABCDE
T_4	BCD
T_5	BCDE

(a)

Support	Frequent Itemsets	Maximal Itemsets	Closed Itemset
3	CD,CE,DE,BCD, BCE,BDE	BCE,BDE, BCD	BCE,BDE, BCD
4	C,D,E,BC,BD,BE	NA	BE,BD,BC
5	B	NA	B

(b)

Fig. 2.2. (a) Itemset database with 5 transaction sets (b) Frequent, maximal and closed itemsets of (a) with $minsup = 3$

2.1.1 Frequent Pattern Mining

Let, $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ be a database, where each $T_i \in \mathcal{T}, \forall i = \{1 \ldots n\}$ represents a transaction. $\mathbf{t}(p) = \{T_i : p \subseteq T_i \in \mathcal{T}\}, \forall i = \{1 \ldots n\}$, is the *support-set* of pattern p. This set contains all the transactions in $\mathcal{T}$ of which pattern p is part of. The cardinality of the *support-set* is called the *support* of p. Pattern p is called frequent if $support(p) \geq \pi^{\mathbf{min}}$, where $\pi^{\mathbf{min}}$ is predefined/user-specified *minimum support (minsup)* threshold. A closed frequent pattern is defined as a pattern that does not have a super-pattern (super-set) that is frequent with the same support count. The pattern is maximal if it is not a sub-pattern of any existing frequent pattern. if $\mathcal{F}$, $\mathcal{C}$ and $\mathcal{M}$ are the set of all frequent, closed and maximal itemset patterns, respectively then $\mathcal{M} \subseteq \mathcal{C} \subseteq \mathcal{F}$ is true.

Note that, if we divide the *support* of a pattern by the total number of transaction, we got normalized support. There can be different types of pattern i.e. set, graph, sequence and tree. In case of set, each transaction T_i is a set of items whereas for graph, its a labeled, undirected and connected graph.

Example: Figure 2.4(a) shows a database with 3 graphs (G_1, G_2 and G_3). With $\pi^{\mathbf{min}} = 2$, there are thirteen frequent subgraphs as shown in Figure 2.4(b). Figure 2.2(a) shows a toy itemset transaction data-set with 5 transactions and Figure 2.2(b) shows all frequent, closed and maximal itemset patterns for $\pi^{\mathbf{min}} = 3$.

2.1.2 Frequent Pattern Space

Candidate generation and test is the most common form of frequent pattern mining algorithms. In each iteration of mining, the algorithm first generates candidate pattern(s) and then it compares each of these patterns against the transactions in the dataset to compute frequency and test if the frequency falls below of a minimum frequency threshold. Depending on the type of patterns i.e. set, sequence or graph, work procedures inside these two steps differs. For itemset pattern, if I is the set of all items in the dataset, each iteration of candidate generation enumerates all possible itemset patterns $p \subseteq I$ as candidates for being frequent patterns. The size of the candidate search space is exponential, since there are $2^{|I|}$ potential frequent itemset patterns. Large candidate space if true for sequence and graph type patterns. For sequence, the set I contains events and a candidate pattern is an ordered list of repetitive events i.e sequence. For graph, the set I is a collection of edges and a candidate pattern is a graph induced by the edges from set I. It is informative to mention that the structure of the candidate patterns space forms a lattice, where any two patterns p_1 and p_2 is connected by a link if and only if $p_1 \subset p_2$ and $|p_2| = |p_1| + 1$.

Example: In Figure 2.3, we show the lattice structure of the candidate space for itemset patterns for the toy data mentioned in Figure 2.2. As we can see, candidate generation step performs a bottom up exploration from the null set using BFS or DFS over the lattice. A candidate can have multiple generation steps, but prefix-based enumeration ensures to use only one path for candidate generation. For example, pattern AB can be generated from A (shown in solid line) or B (shown in dotted line). Prefix-based enumeration generates AB only from A. We also mention the support of each candidate and for minimum support 1 all the candidates qualify as frequent patterns, hence produces exponentially large output set for further analysis. For minimum support 2, significant amount of candidate patterns qualify as frequent as well. We use red ink to mark the frequent patterns for support 3 in the Figure.

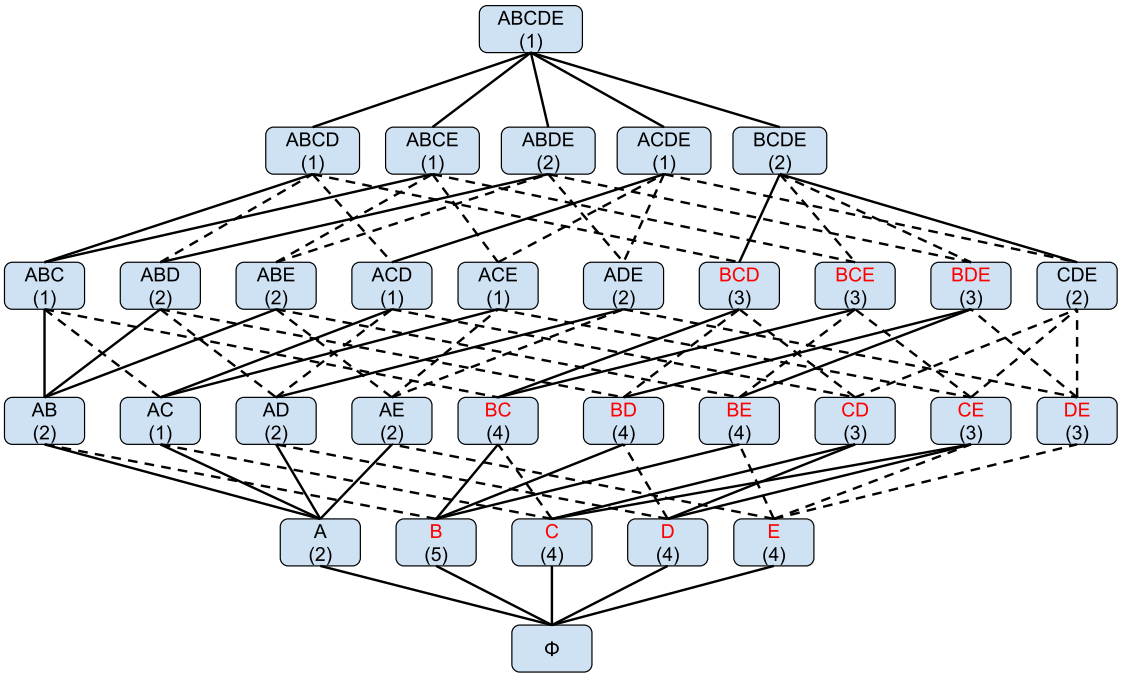


Fig. 2.3. Candidate space lattice of the toy itemset dataset

In Figure 2.4, we show the frequent (minimum support 3) pattern space for the toy graph data mentioned above.

2.2 Related Works

In this Section, we discuss related works of frequent pattern mining.

2.2.1 Frequent Pattern Summarization

To confront the challenge of managing exponentially large mined frequent patterns i.e. solve "information overload" problem researchers develop several pattern summarization algorithms. All of this works have a common goal of producing a smaller number of patterns that globally represent the entire dataset. In the following, we discuss several of these works.

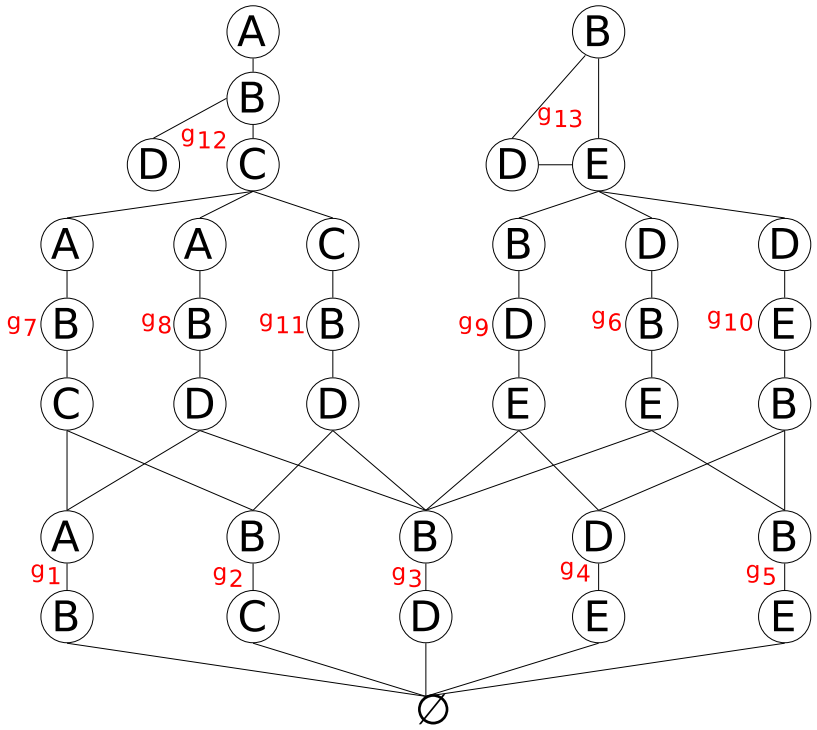


Fig. 2.4. Frequent subgraph patterns lattice for minimum support 2

Mining frequent maximal patterns [20] is one of the early attempts for pattern summarization. A frequent pattern is called maximal, if it does not have any super-pattern. Maximal patterns have the ability to regenerate all frequent patterns but due to support information loss, sometimes this approach become unacceptable. Next attempts for pattern summarization came along with the closed patterns [21]. A pattern is called closed, if it has no super-pattern with the same support. Closed patterns store support information successfully but the number of closed patterns can be very high in number.

Mining top-k patterns is another popular strategy for pattern summarization. This techniques only applicable when the user has a predetermined number of patterns to work with. Han et.al propose one of the earliest work [22] in this direction where they mine k-most frequent closed patterns with a user-specified specific minimum length. In [23], authors design a top-k pattern mining algorithm with high

compressibility and maximal coverage as summarization criteria. Authors in [24], develop an algorithm to mine top-k patterns that collectively offers the best significance with minimal redundancy. They compute significance by the degree of usefulness of a pattern and redundancy between a pair of patterns by considering their similarity.

Following the foot-steps of the research work mentioned above, scientist have invested their time to develop several pattern summarization algorithms from different perspective, notable among them are the profile-based techniques [4, 5], support distance-based methods [6] and maximum entropy-based models [2,3]. Yan et.al in [4] summarizes a set of frequent itemset set pattern using a small number of representative patterns those can be easily usable by a user. The proposed algorithm finds such representative set that has maximum coverage over the entire patterns set as well as capable to approximate supports of any frequent patterns. The authors design a generative model to identify and create profile of these representatives to recompute supports of the patterns without the help of original dataset. Cp-Summary by [5] extends the pattern-profile designed by [4] and propose a new profile called "Cp-summary" which is essentially a list of pattern-profile. Cp-summary allows encoding of several frequent patterns into a single profile, hence enable easy analyze of the summary in the profile independently. In [6], authors claim to produce minimum pattern summarization set possible in practice along with a theoretical error guarantee. Authors investigate the computational deficiency for finding a small representative pattern set and propose an algorithm with reduced execution time and less usage of memory. Specifically, the proposed method uses a tree structure called CFP-tree [25] that saves frequent pattern in a compact efficient manner.

Mampaey et.al developed a probabilistic maximum entropy-based model [2] that can iteratively identifies patterns that carry most novel information. This model operates by looking for patterns whose frequency in the data will surprise the user most and in turn updates the model accordingly. The proposed algorithm is called "MTV". "MTV" is capable of extracting top-k most informative itemsets. At the same time "MTV" can be used to identify a representative set of itemset patterns

that summarizes the original data using the Bayesian Information Criterion (BIC) or the Minimum Description Length (MDL). "MTV" is a one-phase algorithm that can mine patterns and their supports on-the-fly. Vreeken et.al propose an algorithm called krimp [3], that uses minimum description length (MDL) to identify the best set of patterns that compress the database best. None of the above pattern summarization techniques can be helpful for personalized pattern discovery because such approaches solve interesting pattern discovery from a global perspective which is far from personalization what is needed to meet the pattern discovery demand of a specific user.

2.2.2 Interactive Knowledge Discovery

There are some recent works in interactive knowledge discovery, which are not necessarily frequent pattern mining. Examples include mining geospatial redescriptions [26, 27], and subgroup discovery [28–31]. Broadly speaking these works fall in the category of interactive pattern discovery, but the problem definition and the scope of these works are very specific.

In the work on mining geospatial redescriptions [26, 27], the authors propose a system called SIREN, in which a user builds queries based on her interests and the system answers the queries through visualization of redescriptions (different ways of characterizing the same things). The interaction in SIREN divided into two stages: i) interact with the algorithm and ii) interact with the visualized result. The user of SIREN moves back-and-forth between executing commands to discover new results and analyzing the discovered ones.

In [28], authors proposed an interactive solution for subgroup discovery. The developed method begins from a set of top subgroups as per an arbitrary objective measure and attempts to learn a ranking function that can used to rank subgroups. The learned ranking function can later be used to discover novel interesting subgroups. In [30] the authors proposed an Interactive Diverse Subgroup Discovery (IDSD) framework leveraging the diverse beam search. In the proposed framework,

the selection of beam is interactive: on each level of the search, a user interacts with the system to influence the beam by providing feedback in-terms of liking and disliking the subgroups.

In [31], authors develop an interactive framework for subgroup discovery. At each step of the proposed method, first a user visualizes group members then optionally perform an operation on the group either by adding or deleting members, then, the user choose an operation either exploitation or exploration to discover more groups i.e more users. Each discovery action results in k-most relevant and diverse groups. Authors formalize the group exploitation and exploration as and optimization problem and devise a greedy algorithms for efficient group discovery.

2.2.3 Interactive Pattern Mining

Surprisingly, existing works on interactive pattern mining (IPM) is sparse [32–34]. One of the main techniques that the existing works follow is constraint-based mining, where a user can include additional constraints as an interactive input. The mining system considers these constraints to filter the output set to tailor it according to the user's requirements. In [32], authors present an algorithm called "DualMiner" that efficiently prunes its search space using both monotone and antimonotone constraints. This is the first work that claims to handle both types of constraints i.e. monotone and antimonotone simultaneously. [33] introduces "ExAnte", a pre-processing data reduction based approach which reduces the search space dramatically. ExAnte can leverage succinct monotone constraints as well as convertible monotone constraints that reduces the mining computation. Authors also show that ExAnte can be coupled with any constrained pattern mining algorithm, and claim it is always favorable to start any constrained patterns mining with an ExAnte preprocess. In [34], authors present an interactive visual pattern mining framework called "MIME". "MIME" lets a user to browse through the data and patterns easily and intuitively. It provides a toolbox consisting of several interestingness measures, mining and post-processing

algorithms that can be used to identify interesting patterns. Interactive mining allows a user to add their background knowledge with their subjective interestingness criteria to efficiently discover most interesting patterns.

All of the above constrained based pattern mining has following drawbacks. Setting constraints is effective for some mining problems, but for many others, applying hard constraints may yield sub-optimal results. Furthermore, designing constraints to guide mining is not easy; an analyst first needs to explore some example patterns before she can think about the constraints that would work optimally for her specific applications. This can best be described as *"I don't know what I am looking for, but I would definitely know if I see it."* Additionally, in the scenario of an inter-disciplinary research, a domain expert may find it difficult to set constraints for pattern mining—in particular; a constraint must be well-formed according to its formatting specification, which may require significant time from the domain expert.

Few methods of interactive mining exist that are not constraint-guided—probably the most similar to our work are [9,10]. However, our method differs from [9] in various ways. First, [9] requires to have a complete set of frequent patterns up-front, whereas our method unifies the exploration and selection of the frequent patterns using MCMC based random walk—thus for many data-sets, our method may not explore all the frequent patterns. Second, our method uses binary feedback instead of rank order feedback that the above method adopts—the latter puts more burden on the user, and in the case of a hidden data-set, the user may have little information to rank the patterns properly. Boley et al.'s [10] work also considers learning a ranking function of a user over the frequent patterns' space. They do overcome the constraint of mining all frequent patterns up-front, but this work also requires the user to provide a complete ordering among released patterns. There also exist a few notable works [35, 36], that mine a small set of interesting patterns by defining novel interestingness metrics for frequent patterns. For instance, Mampaey et al. [35] summarizes the frequent pattern set and Blei et. al. [36] defines subjective interestingness, both using maximum entropy model.

2.2.4 Interestingness Measures of Patterns

As discussed earlier, for years frequency has been heavily used as a interestingness measure of patterns. But for various data mining task, frequency alone is not a sufficient criterion. To address the shortcoming of frequency, over the years researchers have proposed several interestingness measures [37] mostly to assess the quality of association rules. In [38], authors propose "lift" that compares the conditional probability of an association rule's precedent to its unconditional probability. A lift score higher than one denotes positive correlation between the antecedent and precedent of an association rule and a value smaller than one shows negative correlation. Brin et.al in [39], introduces "Multi-way" χ^2 test to measure interestingness of an itemset directly. In [8], Heikinheimo et al. propose to mine low-entropy itemsets using the entropy measure. Webb et al. introduces the measure of leverage [7] that computes the difference between observed and expected joint probability of an association rule. In many cases, for a specific user, none of these interestingness metrics represent true interestingness over the patterns, hence the motivation of incorporating the user in the discovery pipeline to capture true interestingness of a particular user.

3. INTERACTIVE PATTERN MINING ON HIDDEN DATA: A SAMPLING-BASED SOLUTION

3.1 Introduction

Frequent pattern mining plays a key role in exploratory data analysis. Over the last two decades, researchers invented various efficient algorithms for mining patterns of varying degrees of complexity–such as, sets [12, 40], sequences [41], trees [42] and graphs [43, 44]. All these algorithms assume that the entire data-set is available to the analyst before the mining task is commenced. This assumption holds if the data owner and the data analyst are the same entity. However, for the case of hidden data, data is owned by an enterprise and an analyst who mines the data in the quest for interesting patterns is an outsider who does not have full access. The data owner, though interested in maintaining the ownership of the data by prohibiting full duplication, wants to help the analyst by providing a curated set of frequent patterns that would benefit her the most. We call this task frequent pattern mining from hidden data. A real-life example of such a knowledge discovery setup is given below.

User session data, which contains a collection of related queries that a user executes in a browsing session, is a valuable data source for an eCommerce marketplace. It helps the marketplace to build features, such as query suggestion [45], query segmentation [46], and product recommendation [47]. To maintain the competitive advantage, the data manager of the marketplace keeps the user session data inaccessible (hidden to public). However, the marketplace also has incentives to make a summary of the user session data, say, a subset of interesting itemsets of user queries available to its sellers so that the sellers are aware of the demand trends in the marketplace. Availability of interesting itemsets of user session queries also helps the sellers in choosing an informative title for their product to facilitate effective match-

ing of seller's product with the buyer's query—eventually, boosting the marketplace's revenue. Since different sellers may be interested in distinctive sets of queries, the key challenge for the marketplace in this task is to find a mechanism to assist a seller by providing interesting itemsets of queries that would benefit that specific seller the most—all without disclosing the user-session data.

Frequent pattern mining from hidden data that is manifested in the above example is different from the existing works on privacy preserving pattern mining [48]. In the case of former, the data owner's main concern during pattern mining is to maintain the dataset's privacy as a whole, such that an approximate reconstruction of the hidden data from the released patterns is impossible. Another concern is that the patterns that the data owner releases are maximally beneficial to an analyst's specific knowledge discovery endeavor in that session. On the other hand, privacy-preserving pattern mining methodologies proposed in the existing works safeguard a subset of data entities so that their attributes cannot be inferred by observing the association of those entities with others in the released patterns.

Our proposed setup of frequent pattern mining from hidden data is as follows. For a given analyst, the data owner creates a sampling object, which samples frequent patterns from the data-set in an iterative manner. The sampling distribution from which the samples are obtained is computed dynamically based on the analyst's feedback on the selected samples, which makes the mining process user-interactive. The sampling object uses Markov Chain Monte Carlo (MCMC) sampling, an indirect sampling strategy so that the samples can be obtained without a complete instantiation of the desired sampling distribution. The sampling object also computes a global measure of data disclosure as it releases patterns, which guarantees that the analyst will not be able to reconstruct the dataset by using the released patterns. Note that, an approximate reconstruction of a transaction dataset from the list of itemsets and their corresponding support is $\mathcal{NP}$-complete [49, 50], which provides further assurance regarding the viability of the proposed setup.

Our solution for frequent pattern mining from hidden data utilizes sampling of patterns from the frequent pattern space. In literature, there exist several pattern sampling algorithms [51–55], but they do not fulfill the purpose of our task. The major difference of these algorithms with ours is that the sampling distribution of our method changes through user's interaction, whereas in the above-cited works the sampling distribution remains fixed throughout the sampling session. Such an interactive update of sampling distribution specifically suits the task of pattern mining from hidden data, because for a hidden dataset the analyst only has a vague idea about the data-set's content and she is unable to provide a precise definition of the patterns that will best serve her interest. Our system enables the analyst to provide feedback on the quality of the patterns that she receives. The sampling mechanism uses these feedbacks to bias the sampling towards the preferences of the analyst so that the patterns obtained from subsequent samples are more relevant. In this work, we consider a simple binary feedback mechanism, where a user of the system specifies whether the pattern she receives from the system is of her liking or not. Nevertheless, the method can be easily adapted to a complex feedback mechanism.

Even if the data is not hidden, the interactive sampling of frequent patterns is useful on its own merit. For example, in an interdisciplinary research, the data may not be hidden to the domain expert. However, the expert has little knowledge about the quality of the patterns that he obtains by mining those data. Effective discovery of useful patterns requires domain knowledge; on the other hand, efficient incorporation of domain knowledge requires an interactive console. Existing pattern mining algorithms are not interactive, and they generate an enormous list of frequent patterns, in which; a high degree of redundancy prevails. Such a tool is hardly helpful for a domain scientist. An interactive pattern mining approach that enables the domain scientist to sample a small set of relevant frequent patterns is a better alternative.

Figure 3.1 illustrates the framework that we devise to mine patterns from a hidden data-set. In this figure, we show a set of k analysts, $u_1, u_2, \ldots, u_k$, that are simulta-

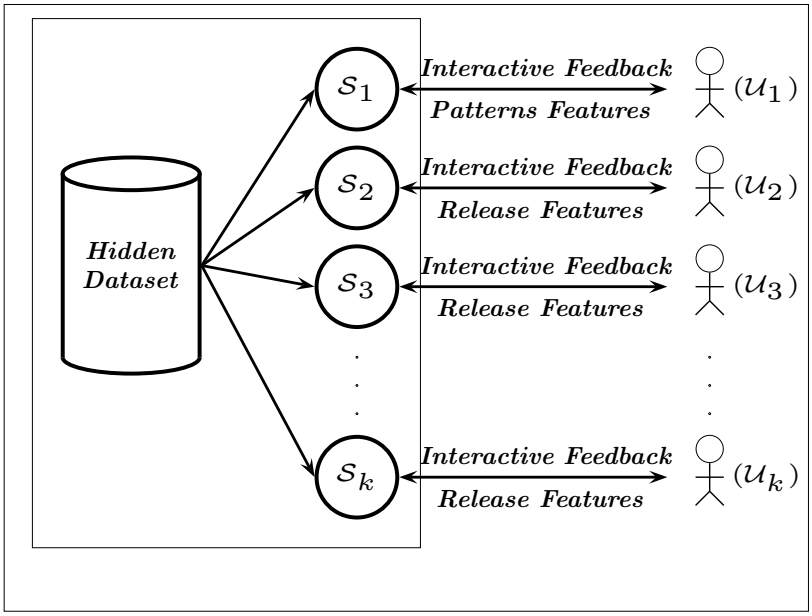


Fig. 3.1. Sampling-based interactive pattern mining on a hidden dataset

neously accessing frequent patterns from a hidden data-set. For a specific user, u_i, a dedicated MCMC sampler, s_i is assigned to facilitate the interactive mining session. Based on u_i's feedback (binary) on the sampled patterns, s_i updates the sampling distribution so that the analyst u_i is served in the most fruitful way. For a mathematical representation of the sampling distribution, we use a vector space model [56] of unit-size patterns. By appropriately defining the unit-size pattern, our solution can easily be applied for sampling patterns of various complexity, such as itemsets, trees, sequences, and graphs. The contributions of this work are summarized as below:

- We propose an interactive pattern mining system for mining frequent patterns from a hidden dataset. The system uses an MCMC sampler for selecting a small set of frequent patterns from a sampling distribution, which is updated iteratively through user's feedback on the sampled patterns. Thus, the proposed system overcomes the *information overload* problem, which is caused by a large

number of frequent patterns returned by a traditional frequent pattern mining algorithm.

- We show that our proposed pattern mining system protects the integrity of a hidden dataset by releasing frequent patterns in a restrictive manner such that the probability of entire data-set reconstruction using the released set of patterns with high accuracy is very low.

- We show experimental results on several real-life itemset and graph datasets to prove the effectiveness of our proposed method on two aspects: pattern selectivity, and dataset privacy disclosure.

The remainder of this chapter is organized as follows. In Section 3.2, we discuss related works in the area of interactive pattern mining and pattern sampling. In Section 3.3 we formally define the problem. In Section 3.4, we discuss the key concepts that are important to understand the proposed method. In Section 3.5, we discuss technical details of the proposed MCMC sampling based system; also include discussion of interestingness function design for set and graph pattern (Subsection 3.5.1, 3.5.2 and 3.5.3), feedback mechanisms (Subsection 3.5.4), convergence of the MCMC walk (Subsection 3.5.5), pseudo-code and parameter sensitivity 3.5.6. In Section 3.6, we present extensive empirical evaluation including pattern selectivity performance (Subsection 3.6.3), execution time (Subsection 3.6.7) and privacy disclosure (Subsection 3.6.8) based analysis of the proposed system on various real-world data-sets. We conclude this chapter with two real-world case studies in Section 3.6.9, and future research direction in Section 3.7.

3.2 Related Works

We discuss the related works under three different categories.

3.2.1 Data Analytics over Hidden Databases

Data analytics over hidden databases is an active research direction; typically, these hidden databases are part of the deep web that can be accessed only by a form-like query interface. Along this research direction, various tasks are considered, such as, crawling the hidden web [57], obtaining a random tuple from a hidden database [58, 59], retrieving top-k records by sampling [60], and estimating the size and other aggregates over a hidden database [61]. These works are related to our work in the sense that we consider frequent pattern mining, also over the hidden databases. However, we do not assume a form-like interface to access the database, rather we assume that the data owner provides a random sampler to interact with the database, and the sampler is guided by the data analyst through interactive feedback.

3.2.2 Privacy Preserving Pattern Mining

In existing literature, different privacy assumptions have been used for the task of privacy preserving pattern mining. For instance, several works [62–64] design the task of privacy preserving pattern mining with an assumption that a list of sensitive frequent patterns are known to the data owner, who sanitizes the dataset before its disclosure so that the sensitive patterns remain hidden (by being infrequent) on the sanitized dataset. In another setup [65], it is assumed that each of the transactions in the input dataset belongs to a distinct data owner who due to privacy concern modifies the transaction before sending it to a mediator that runs the frequent itemsets mining methodologies. In both the above setups the data owner reveals the dataset after making random modifications on it. Frequent pattern mining over vertically [66], and horizontally [67] partitioned datasets are also solved; these setups assume the

existence of multiple parties, where each of the parties owns a data partition, and the parties want to find globally frequent patterns without revealing any local information. Secure multi-party computation over encrypted data are generally used for solving the above problems. Our problem setup is clearly different from the above works as we consider the dataset to be hidden and during the mining phase, the entity(user) does not receive any part of the transaction data irrespective of its encryption status.

3.2.3 Frequent Pattern Sampling

For mining frequent patterns from a hidden data-set, we use an MCMC based random walk on the frequent pattern space. In recent years, various other works also follow a similar approach—notable among these are [51, 52, 54, 55]. In [51], the authors propose a Markov Chain Monte Carlo based randomized algorithm to count the number of frequent sets for a given minimum frequency threshold. Such a method is useful for finding an approximate count of frequent patterns—approximate count is useful for a dense data-set or for a low frequency threshold scenario, for which exhaustive counting algorithms fail to complete. In another work [54], the authors develop direct sampling methodologies for sampling patterns from various distributions, such as frequency, area, and squared frequency. For frequency based sampling, the proposed algorithm first samples a data record from the input data-set randomly with a probability proportional to the size of its power set. Then it returns a uniformly sampled subset of that data record. For other desired distributions, the method finds an effective approach for sampling from these distributions. An improved sampling framework is proposed in [55], which uses coupling from the past (CFTP) technique. CFTP helps to avoid super-linear preprocessing and memory requirements, and it can simulate more complex distributions.

In [52], Hasan et al. propose a mining framework, that they call *output space sampling*, which samples frequent patterns using a user-defined distribution. The sampling framework that we devise in this work is similar to the work in [52]. How-

ever, similar to other existing pattern sampling works, [52] also uses a fixed sampling distribution throughout the mining session, whereas in our work, the sampling distribution updates in response to the user's feedback. Moreover, in [52], the random walk is performed over the edge of the Partial Order Graph (POG) (defined formally in Section 3.4); our work deviates from this by performing a random walk over a state-transition graph in which a random graph is overlaid on top of the POG graph—the above modification improves the convergence of an MCMC walk significantly.

3.3 Problem Statement

In this section, we formally define the problem of interactive pattern mining from hidden data. Given a hidden data-set[1] $\mathcal{D}$, and a user u, an interactive pattern mining system (IPM) returns a sequence of randomly selected frequent patterns $p_1, p_2, \ldots p_t$ from $\mathcal{D}$ to u; optionally u can send feedback $f_1, f_2, \ldots f_t$ to the IPM regarding the quality of the patterns; in response to the feedback, IPM iteratively updates its sampling distribution so that the randomly selected patterns that it returns to u in subsequent iterations align better with u's interest. In the following paragraphs, we will detail the followings: (1) how does the IPM system find a pattern? (2) How does the user u form a feedback on a pattern? And (3) how does the feedback from u is incorporated in the mining process to obtain a better pattern in subsequent iterations?

Given that user u has no access to the data-set, and the data owner has no intention to share all the frequent patterns, traditional frequent pattern mining algorithms are of no use in this scenario. We also assume that the data-set is large; therefore, the complete enumeration of all the frequent patterns may be infeasible. So, the IPM adopts the output space sampling (OSS)-like [52] paradigm to obtain a frequent pattern and share it with the user u. However, OSS does not provide an option for interactive pattern mining; it merely allows the user to mine from a user-defined sam-

[1]the words *data-set*, and *database* are used synonymously this this document

pling distribution. So, to accommodate an interactive mining setup, IPM adapts the sampling distribution using u's feedback, so that it respects u's interest progressively. If we assume that u has an underlying *interestingness function*, γ, that maps each frequent pattern to a non-negative real value (higher value stands for more interesting), the adaptation of sampling distribution by using the feedback translates to the inverse problem of learning the γ function.

An important question that is yet to be answered is what is the composition of u's feedback on a pattern. In this work, we consider a simple feedback mode, where the response of the user u on a pattern p is purely binary: *interesting* and *non-interesting*. One can always consider a more complex feedback mechanism, may be in the form of a profile vector, in which the user u represents the pattern p as a composition of a set of sub-patterns and uses a probabilistic vector to define her interest in each of these parts; however, we do not explore this option in this work. Now, under the scope of a binary response, the set of frequent patterns can be divided into two groups: useful and non-useful, and it is left to the IPM to deduce the underlying discriminating function by which the user u is building her opinion.

We like to clarify that our problem formulation is generic, and it considers the task of mining all kinds of patterns, such as sets, sequences, trees or graphs. In most of our discussion in this chapter, we will refer to the term "pattern" and will dictate the discussion considering a graph or an itemset pattern, but with minor or almost no adjustment, the discussion will fit for different kinds of patterns. Experiment section also shows results involving both itemset and graph patterns.

3.4 Background

In this section, we define several key concepts that will be useful to understand our research work.

3.4.1 Frequent Partial Order Graph

Based on the sub-graph relationship, the set of all frequent patterns ($\mathcal{F}$) forms a partial order called *Subgraph partial order*. The lowest pattern in this partial order is the null ($\emptyset$) pattern. The partial order can be represented as a graph, called *partial order graph (POG)*. Every node in the POG corresponds to a distinct frequent pattern. Every edge in POG represents a possible extension of a frequent pattern to a larger (by one unit) frequent pattern. Algorithms for enumerating all frequent patterns typically traverse the POG in either depth-first or breadth-first manner, starting from the lowest pattern. Figure 3.2(a) shows the frequent subgraph partial order for the graph mining task shown in Figure 2.4. It is easy to see that for the case of mining other patterns, such as sets, sequences and trees, a similar POG can be built, using the subset, subsequence, or subtree relationship, respectively.

3.4.2 Markov Chains, Random Walk and Metropolis-Hastings (MH) Algorithm

A **Markov chain** is the sequence of Markov process over the state space S. The state-transition event is guided by a matrix T, called *transition probability matrix*. A Markov chain is said to reach a stationary distribution π when the probability of being in any particular state is independent of the initial condition. Markov chain is reversible if it satisfies the *reversibility condition* $\pi(i)T(i,j) = \pi(j)T(j,i), \forall i,j \in S$. A Markov chain is *ergodic* if it has a stationary distribution. If the state space (S) of a Markov chain is the vertices of a POG, and the state transition happens only between adjacent nodes in POG, then this Markov process simulates a random walk on the frequent pattern space ($\mathcal{F}$).

The main goal of the Metropolis-Hastings algorithm is to draw samples from some distribution $\pi(x)$, called the *target distribution*, where, $\pi(x) = f(x)/K$; here K is a normalizing constant which may not be known and difficult to compute and $f(x)$ is the likelihood function that we know how to compute. MH algorithm can be

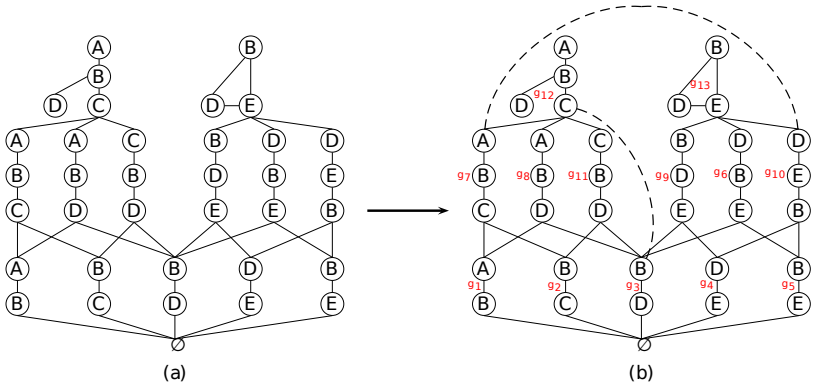


Fig. 3.2. (a) Partial Order Graph (POG) (b) State-transition graph

used together with a random walk to perform Markov Chain Monte Carlo (MCMC) sampling. For this, the MH algorithm draws a sequence of samples from the target distribution as follows:

i. It picks an initial state (say, x) satisfying $f(x) > 0$.

ii. From current state x, it samples a point y using a distribution $q(x, y)$, referred as *proposal distribution*.

iii. Then, it calculates the *acceptance probability*,

$$\alpha(x, y) = \min\left(\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}, 1\right) = \min\left(\frac{f(y)q(y, x)}{f(x)q(x, y)}, 1\right) \tag{3.1}$$

and accepts the proposal move to y with probability $\alpha(x, y)$. The process continues until the Markov chain reaches to a stationary distribution.

As shown in [52], a Metropolis-Hastings (MH) based MCMC algorithm can be used for sampling frequent pattern with the desired sampling distribution. In such a method, the set of frequent patterns ($\mathcal{F}$) is the state space of the random walk, and the edges of the POG define the possible state transitions. Importantly, the POG is generated locally as needed. For instance, when the random walk resides on a pattern x, the local neighborhood of the pattern x, which consists of all frequent super patterns (whose sizes are larger by one than the size of x), and all frequent sub-patterns (whose sizes are smaller by one than the size of x), is constructed. The random walk

then jumps to one of its neighbors using the acceptance probability (Equation 3.1) of the MH algorithm. The MH requires to choose a proposal distribution (q), which can simply be the uniform distribution, i.e., each of the neighbors of the pattern x in the POG is equally likely to be chosen. If y is a neighbor of the pattern x, d_x and d_y are the degrees of the pattern x and y in the POG, and γ is the target distribution, following equation 3.1, the acceptance probability for choosing the proposal move is as below:

$$\alpha(x, y) = \mathbf{min}\left(\frac{\gamma(y) \cdot (1/d_y)}{\gamma(x) \cdot (1/d_x)}, 1\right) = \mathbf{min}\left(\frac{\gamma(y) d_x}{\gamma(x) d_y}, 1\right) \tag{3.2}$$

The following Lemma holds:

Lemma 1 *The random walk on POG converges to a stationary distribution which is equal to the desired target distribution, γ.*

PROOF*: See [52]* ■

3.5 Interactive Sampling Algorithm

We also use MH as the underlying sampling algorithm to mine patterns from the hidden database, $\mathcal{D}$. Each sampler s_i in Figure 3.1 (b) runs an instance of the MH sampling with the objective that the MH's target sampling distribution (γ in Equation 3.2) matches the desired sampling distribution of the user u_i. Achieving the above objective is challenging because the sampler has no knowledge of the desired sampling distribution. To overcome this difficulty, we sample patterns using the MH with a default sampling distribution, and update the distribution using each of the user feedback so that the effective distribution converges towards γ with the updates.

3.5.1 User's Scoring Function

We represent the user's desired distribution as a scoring function, $f : \mathcal{F} \rightarrow \mathbb{R}_+$; f maps each pattern in $\mathcal{F}$ (frequent pattern-set) to a non-negative real number. From this scoring function, we can obtain user's desired distribution by $\frac{f(P)}{\sum_{q \in \mathcal{F}} f(q)}$, i.e. the

probability of a pattern in the desired distribution is directly proportional to the $f(\cdot)$ (f-score) of that pattern [2].

For the sake of learning, we must impose a bias on γ and assume that γ is taken from some family of functions [3]. For this, we use the similarity between interactive pattern mining (IPM) and the task of Query by Example in information retrieval (QEIR) [68]—*given a set of documents that a user has liked, retrieve other documents that he is likely to like*; for IPM, the set of patterns for which the user provides a positive feedback plays a role which is similar to the role of the given phrases to construct user's preference profile. Similarly, to model γ, we also adopt a vector space model of unit-size patterns. For example, unit-size patterns for a graph are its edges; for an itemset, they are the items. In this vector space model, each unit-size pattern has a weight, expressed as b^i where $b \in (1, +\infty)$ and $|i| \in 0, 1, \ldots, |\mathcal{F}|$. The γ-score of a pattern is simply the multiplication of these weights [4]. If P is a pattern of size k and P_j are the unit-size patterns that compose the pattern P, we have:

$$\gamma(P) = \prod_{j=1}^{k} \mathbf{w}[P_j] \; \forall P_j \in P \text{ and } |P| = k \tag{3.3}$$

Note that, if two patterns x and y are structurally similar, there will be higher overlap between their constituting unit-size patterns; hence, under the above assumption, they will have similar γ-score.

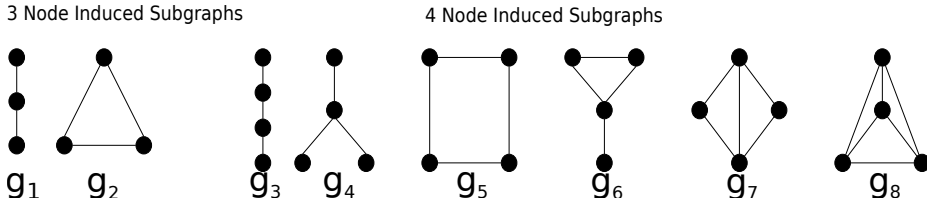


Fig. 3.3. Set of undirected graphical structures with 3 and 4 vertices.

[2] In this chapter, we will use the phrases *scoring function* and the corresponding *sampling distribution* synonymously.

[3] this argument is known as *No Free Lunch Theorem* in the machine learning literature

[4] An additive γ function may also be considered, but we notice that such a function has slow convergence compared to multiplicative functions.

To assign a weight on each unit-size pattern, we assume that the set of frequent patterns are partitioned into two sets: *interesting* ($\mathcal{I}$) and *non-interesting* ($\mathcal{N}$); so, $\mathcal{F} = \mathcal{I} \cup \mathcal{N}$. Furthermore, assume that $\mathcal{F}_1$ is the set of all unit-size frequent patterns. Now, consider a weight vector $\mathbf{w}$ of size $|\mathcal{F}_1|$; each entry $\mathbf{w}[\cdot]$, in this vector corresponds to one of the unit-size frequent patterns; for a unit-size pattern p, $\mathbf{w}[p]$ is equal to b^{u-v}, where $u, v \in [0, 1, \ldots, |\mathcal{F}|]$, and they are, respectively, the number of interesting and non-interesting patterns (according to a user) in which p is embedded. In this way, if a unit-size pattern is part of x interesting (from the set $\mathcal{I}$) and y non-interesting (from the set $\mathcal{N}$) patterns, its weight will be b^{x-y}.

Example: In Figure 2.4 (b), if $\mathcal{I} = \{g_1, g_2, g_7, g_{12}\}$ and $\mathcal{N} = \{g_3, g_4, g_5, g_6, g_8, g_9, g_{10}, g_{11}, g_{13}\}$. Suppose $b = 2$, then, $\mathbf{w}[A\text{-}B] = 2^{3-1}$ as the unit-size pattern A-B is part of three (g_1, g_7, g_{12}) interesting and one (g_8) non-interesting patterns. On the other hand, $\mathbf{w}[B\text{-}D] = 2^{1-6}$, as this edge appears in 1 interesting and 6 non-interesting patterns. Also, $\gamma(g_7) = 2^4$, which we obtain by multiplying the weights of each of its edges (unit-size patterns). ■

3.5.2 Updated Scoring Function(γ) for Graph Patterns

The scoring function (Equation 3.3) that we discuss in the previous subsection works for all different kinds of patterns, including, graphs and itemsets. In this scoring function, the γ-score of a pattern p is computed from a weight vector of unit size patterns. In the case of a graph pattern, we consider the edges to be the unit size pattern, so the scoring function of a graph pattern is influenced by the vertex and edge labels of its constituting edges. However, although the edges are the building block of a graph pattern, the above scoring function considers each edge independently, ignoring higher order topological structures, such as a triangle, a star, or a rectangle (see Figure 3.3). Hence, the presence or absence of some higher-order topologies in a graph pattern has no consequence on the γ-score of that pattern. Nevertheless,

from the user's perception of a graph pattern, interestingness of a graph pattern may be influenced by the fact, whether the pattern includes or excludes some specific graphical structure. To address this, we modify the earlier scoring function as below.

The scoring function for graph patterns incorporates the influence of higher-order topologies through an additional weight vector ($\mathbf{w}_g$) of appropriate size; every entry in this vector corresponds to one of the higher-order topologies. In this work, we consider eight topologies as shown in Figure 3.3 which are all undirected topologies that have 3 or 4 vertices. However, depending on the domain of the data-set, one can always add or discard entries in the $\mathbf{w}_g$ vector for other topologies. Now, each entry of w_g is equal to b^{u-v}, where $u, v \in [0, 1, \cdots, |F|]$ are the number of interesting and non-interesting patterns in which the corresponding graphical topology is an induced subgraph. Update and usage of w_g is similar to the weight vector of frequent edges. Each entry of w_g has an initial value of 1. Depending on the feedback's nature, sampler updates the weight of each induced subgraph by multiplying or dividing by b. Thus for a graph pattern P we will have two weight vectors containing weights of unit length patterns and weights on higher-order topologies, respectively. We denote this newly formulated score for graph pattern as γ_g-score; for the pattern P, it can be represented by the following equation:

$$\gamma_g(P) = \frac{1}{2}\left(\prod_{j=1}^{k} \mathbf{w}[P_j] + \prod_{g \in \mathcal{G}} \mathbf{w}_g[g]\right) = \frac{1}{2}\left(\gamma(P) + \prod_{g \in \mathcal{G}} \mathbf{w}_g[g]\right)$$
$$, \ \forall P_j \in P, \ \forall g_i \in P \ \ |P| = k \text{ and } \mathcal{G} \text{ is the set of all topologies} \tag{3.4}$$

As we can see, γ_g is simply the average of the scores of unit-length patterns, and higher-order topologies. Any other linear combination of these factors can also be used based on the domain knowledge to balance the influence of vertex/edge labels and higher-order topologies. For our data-set, we found average to be a good choice, and it improves the performance of interactive mining substantially.

3.5.3 User Interaction Design

The objective of interactive pattern mining is to sample each pattern such that their sampling probability is proportional to their score, where the score is defined by γ (or γ_g, for the case of a graph pattern). The normalized form of the score vector is the target probability distribution; as it was mentioned earlier, we use the same symbol (γ or γ_g) to denote both the scoring function and the target probability distribution. Unfortunately, the scoring function is unknown to the sampler. Rather the sampler (s_i) uses a proxy distribution (say β) as the effective target distribution, which the sampler updates with each of the feedback. If $\beta^0, \beta^1, \ldots, \beta^k$ are sequences of β's along with the updates, the limiting value of β converges to the actual scoring function. For simplicity, we will first use γ in the following discussion. Later, we will show that the discussion holds for γ_g as well.

The proxy distribution, β requires to converge to γ, i. e., $\mathbf{Lim}_{k\to\infty}\beta^k = \gamma$, so we ensure that each of the β^k distributions also belongs to the same family of distribution as of γ. Thus, the $\beta(\cdot)$ (β-score) of a pattern is computed as the multiplication of the weight of its constituting unit-size patterns, identically as it is computed for the γ-score (see Equation 3.3).

At initialization, all unit-size patterns have a weight of 1, and sampling using such a β (let's name it β^0) as the target distribution yields a random initial distribution. The sampler starts sampling from the initial distribution by using the MCMC based random walk; suppose, at some step of the walk, it (the sampler) picks a pattern P of length k for user's feedback. If the pattern P has already been used for feedback, the pattern is discarded and the process continues until another pattern is found for which the feedback has not been sought yet. Then, for the selected pattern, the user provides her feedback by indicating whether she likes the pattern P or not. If the feedback is positive, the sampler increases the weight of each of k length-one sub-patterns of P exponentially, i.e. it multiplies the current weight of those length-one sub-patterns by b. On the other hand, if the feedback is negative, the sampler decreases the weight

of those length-one sub-patterns by dividing the current weight by b. This weight modification updates the existing sampling distribution β. The sampling step and the interaction step are repeated intermittently as the β function iteratively moves closer to the user's desired distribution γ. The above weight updates also ensure that in every β^k, there exists a positive probability of visiting any frequent pattern from any other frequent pattern; which maintains the ergodicity of the random walk.

Lemma 2 $\boldsymbol{Lim}_{t\to\infty}\beta^t = \gamma$.

PROOF*:* β^t is the updated sampling distribution after t iterations; in each of these iterations, exactly one pattern is sent for user's feedback; hence, total number of patterns for which feedback is sought is t.

Now, consider, any unit-size pattern p, and assume that it appears in $r(\leq t)$ patterns out of those t patterns. Now, let's partition the r patterns into two set, based on their feedback status, interesting or non-interesting. If the sizes of these sets are r_i and r_n, respectively, according to the weight update mechanism, in β^t the weight of the pattern p, $\mathbf{w}[p]$, is equal to $b^{r_i - r_n}$.

Now, in all sampling distributions, $\beta^k : 0 \leq k \leq \infty$, every interesting/non-interesting pattern in $\mathcal{I}$ and $\mathcal{N}$ has a non-zero probability to be visited; hence, as t goes to infinity, all the patterns in the set $\mathcal{I}$ and $\mathcal{N}$ will be sent to the user (at least once) for feedback; Also, for each pattern, we consider its feedback exactly once. So, the weight of p in the limiting distribution β^∞ is exactly equal to b^{x-y}, where x and y are the count of interesting and non-interesting patterns of which p is part-of, which is identical to $\mathbf{w}[p]$ of γ. Hence, $\beta^\infty = \gamma$ ■.

The above proof easily extends to the case of γ_g. If ${\beta_g}^t$ is the proxy distribution for γ_g, and we use the identical update mechanism in β_g same as β, except that for the earlier, we also update the weights on the topology vector $\mathbf{w}_g$ based on the fact whether a topology is an induced subgraph of a given pattern or not.

The following lemma holds.

Lemma 3 $\boldsymbol{Lim}_{t\to\infty}{\beta_g}^t = \gamma_g$.

PROOF*:* From Lemma 2, the unit pattern weight component ($\mathbf{w}$) of β_g converges to γ. For the remaining part, let's consider, a graph topology p and assume that it is an induced subgraph of $m(\leq t)$ patterns out of the t patterns that are sent for feedback. Now, based on their feedback status, let's partition the m patterns into two sets: interesting and non-interesting and assume that the sizes of these sets are m_i and m_n, respectively. According to the weight update mechanism of ${\beta_g}^t$, $\mathbf{w}_g[p]$ is equal to $b^{m_i - m_n}$. As t approaches infinity, $b^{m_i-m_n}$ converges to b^{x-y}, where, x and y are the count of interesting and non-interesting patterns (over the entire frequent pattern space), in which p is an induced subgraph. β_g multiplies the weights of all the topologies that we consider. Hence, ${\beta_g}^\infty = \frac{1}{2}\left(\gamma + \prod_{p\in\mathcal{G}} \mathbf{w}_g[p]\right) = \gamma_g$ ■

Lemma 2 and Lemma 3 proves that the user interaction that we design works perfectly when the γ and γ_g function is constructed as was shown in Equation 3.3 and Equation 3.4. While this is a reasonable assumption about γ and γ_g, in real life an interestingness function can be different. Nevertheless, any interestingness function that a user use should have some structure; alternatively, it should be non-random, where the structural similarity between a pair of patterns should have a strong influence on the possible γ and γ_g-scores of these patterns. As a result, the above proxy distribution β and β_g should work reasonably well.

3.5.4 Feedback Mechanism

As the sampler walks over the frequent pattern space randomly, it sends a subset of patterns that it visits, to the user for feedback. Using feedback, it updates its sampling distribution of the objective that the active distribution converges to user's desired distribution. In this section, we will discuss two mechanisms for choosing the subset of patterns for which sampler seeks feedback.

Periodic Feedback. In this mechanism, the user sets a parameter called *miniter*, and the sampler chooses one pattern for feedback periodically after it performs *miniter* count of steps in its random walk over the frequent pattern space. With this approach, the sampler does not estimate the quality of the pattern for which it seeks feedback; however, it ensures uniqueness, i.e. it does not seek feedback for the same pattern more than once. The process stops once a desired number of feedback are collected. This method is fast as the only selection criteria is to maintain uniqueness, which can be done by computing a singular key for each frequent pattern.

Conditional Periodic Feedback. For an effective learning of user's desired distribution, diversity among the selected patterns is important while nominating them for feedback. Periodic feedback mechanism does not ensure diversity as the sampler does not perform any similarity computation over the visited patterns. As a result, it is possible that the sampler receives feedback for a set of patterns, which are similar to one another. This severely impedes the convergence towards the user's desired distribution. However, direct similarity computation is costly, especially for graph patterns. In this feedback mechanism, we devise an alternative method for promoting diversity.

In this method, the sampler computes whether the feedback (positive or negative) of a pattern alters the existing distribution significantly. It still uses *miniter* as one of the pattern selection criteria; nevertheless, before sending the pattern to the user, the sampler estimates whether feedback (positive/negative) of the chosen pattern changes the β distribution substantially. If this is true, the pattern will be sent to the user otherwise, the sampler waits for the next *miniter* cycle and performs the same selection test. The process stops once the desired number of patterns are chosen for feedback. Since β is composed of the product of weights on the unit-length patterns (see Equation 3.3), the change in weight vector $\mathbf{w}$ can be used to approximate the change in β. To compute the discrepancy between a pair of $\mathbf{w}$'s, we use KL-Divergence. It is defined as below.

Suppose Q and P are two distribution, KL-Divergence of Q from P can be written as

$$D_{KL}(P||Q) = \sum_i P(i) ln \left(\frac{P(i)}{Q(i)} \right) = \underbrace{\sum_i P(i) \ln(P(i))}_{\mathcal{E}_P} - \underbrace{\sum_i P(i) \ln(Q(i))}_{\mathcal{E}_Q} \quad (3.5)$$

In the last expression of the above equation, the first term is the entropy of the distribution P, and the second term is the entropy of the distribution Q relative to P. We use the value $\mathcal{E}_P$ and $\mathcal{E}_Q$ to represent these two terms.

Let's assume $\mathbf{w}$ is the existing weight vector when the sampler chooses a pattern p to get the feedback. Then assuming both types of feedback, i.e. positive and negative respectively, the sampler computes $\mathbf{w}^+$ and $\mathbf{w}^-$ by updating the score of the unit size patterns of p appropriately. By normalizing all the vectors $\mathbf{w}, \mathbf{w}^+$, and $\mathbf{w}^-$ so that they become a probability distribution vector, the sampler computes the percentage of divergences using following equations:

$$Divergence(\mathbf{w}, \mathbf{w}^+) = \left(\frac{|\mathcal{E}_{\mathbf{w}} - \mathcal{E}_{\mathbf{w}^+}|}{\mathcal{E}_{\mathbf{w}}} \right) * 100 \quad (3.6)$$

$$Divergence(\mathbf{w}, \mathbf{w}^-) = \left(\frac{|\mathcal{E}_{\mathbf{w}} - \mathcal{E}_{\mathbf{w}^-}|}{\mathcal{E}_{\mathbf{w}}} \right) * 100 \quad (3.7)$$

Finally, the sampler takes the average of above calculated percentage of divergences (Equation 3.6,3.7). If the percentage of divergence is higher than some threshold, say, η, the sampler will send the pattern to the user for feedback. Using empirical evaluation, we showed that, "Conditional Periodic Feedback" performs better than "Periodic Feedback". For graph patterns, sampler applies above technique for both the edge ($\mathbf{w}$) and topological ($\mathbf{w}_f$) weight vectors and takes the average percentage of divergence to compare against η. The conditional periodic feedback scheme safeguards the system from the user(adversary) who tries to gather as many patterns he can from the system in the name of providing feedback. Even if a user is willing to provide any number of feedback, the sampler only asks for feedback if the feedback help over a certain threshold on its current learning, so when the sampler converges

to a distribution with the help of relatively low number$(60 - 100)$ of feedback (More detail in Section 3.6) it stops asking for feedback.

3.5.5 State-transition Graph and Convergence of MH-based Random Walk

In [52], the authors use partial order graph (POG) as the state-transition graph for sampling frequent patterns. However, by construction, POG is a multi-stage graph—the patterns that have the same size belong to the same stage, and a pattern in stage i has neighbors only from stage $i + 1$ and stage $i - 1$. Such a graph has a large diameter (at least, as large as the size of the largest frequent pattern), which causes slow mixing for any random walk on it (the graph). If the mixing is slower then the sampling quality will be worse. In particular, for our task, fast mixing is crucial—we update the effective distribution (β^t) frequently, so we desire that the random walk to mix well before the effective distribution β^t is updated. The mixing time of a random walk is characterized by the number of steps the walk takes to reach its stationary distribution. For a known target distribution, the mixing time can be estimated by computing the spectral gap [69] of the transition probability matrix P. In short, the higher the spectral gap, the faster the convergence.

Table 3.1
Effect of random edges in POG on spectral gap

Dataset	spectral gap (μ)	
	POG + Random edge	POG
Mushroom	0.045	0.020
Chess	0.082	0.040
Connect	0.085	0.047

To achieve fast mixing, we overlay a random graph on POG, by introducing random edges in the partial POG in a periodic manner with the exploration of a batch

of new nodes of the POG. In Figure 3.2, we show two such random edges (shown as dotted line in the right Figure), between patterns (*A*-*B*-*C*, *D*-*E*-*F*) and (*A*-*B*-*C*-*D*, *B*-*D*). The addition of these two edges reduces the diameter of the POG in Figure 3.2 by 40% (from 5 to 3). The reason for overlaying a random graph is due to the well-known fact that a random graph exhibits rapid mixing property—for instance, the mixing time of an MCMC random walk on a Connected Erdos-Renyi random graph is $\Theta(\log^2 n)$ [70]. In all our experimental data-sets, introducing random edges increase the spectral gap almost by 100%. Table 3.1 shows the improvement in the spectral gap for Mushroom, Chess and Connect data-sets after setting the $randomedge\_factor$ (explained in Section 3.5.6) to 20%. Importantly, the addition of random edges maintains the ergodicity of the MCMC walk. This is because the random edges are symmetric (allow transition along both the directions of the edge), and the addition of these edges maintains the reversibility condition.

Lemma 4 *The random walk on POG with random overlay graph converges to a stationary distribution which is equal to the desired target distribution, γ.*

PROOF*:* In order to achieve a stationary distribution, a random walk needs to be finite, irreducible, and aperiodic [71]. First, POG with random overlay graph is finite since the incorporation of random edges does not introduce any new frequent patterns (states) than it was before in POG. Second, for any two nodes x and y in POG with random overlay graph, there still exists a positive probability of reaching from one to other because every pattern can reach and can in turn be reached from the ϕ pattern. Since at least one path exists between any two patterns via the ϕ pattern, the random walk is irreducible. Finally, the walk can be guaranteed to be aperiodic by allocating a self-loop probability at every vertex of POG with random overlay graph[5]. This proves that the random walk over POG with random overlay graph achieves a unique stationary distribution. Now, consider two adjacent vertices x, $y \in$ POG with random overlay graph. Using equation 3.2 the transition

[5]This is required only from a theoretical standpoint; in our experiment, we do not allocate any self-loop probability explicitly

probability from x to y, $P(x,y) = \frac{1}{d_x} * min\left(1, \frac{\gamma(y)d_x}{\gamma(x)d_y}\right) = min\left(\frac{1}{d_x}, \frac{\gamma(y)}{\gamma(x)d_y}\right)$. Assuming that the stationary distribution probability of the node x is $\pi(x) = \gamma(x)$, then $\pi(x)P(x,y) = min\left(\frac{\gamma(x)}{d_x}, \frac{\gamma(y)}{d_y}\right)$. Similarly, the transition probability from y to x, $P(y,x) = min\left(\frac{1}{d_y}, \frac{\gamma(x)}{\gamma(y)d_x}\right)$. Using $\pi(y) = \gamma(y)$, $\pi(y)P(y,x) = min\left(\frac{\gamma(y)}{d_y}, \frac{\gamma(x)}{d_x}\right)$. We have $\pi(x)P(x,y) = \pi(y)P(y,x)$. Thus the detail balance condition holds using γ as the stationary distribution over the POG with random overlay graph. ■

3.5.6 Algorithm

For an incoming user, the IPM creates a new sampler instance which executes the subroutine *Interactive_Sampling* shown in Algorithm 1. Besides the database $\mathcal{D}$, the method also takes four more parameters: (1) a minimum normalized support ($0 \leq \pi^{\mathbf{min}} \leq 1$) value, which defines whether a pattern is frequent or not; (2) a minimum number of walks (*miniter*) before the sampler returns a pattern to the user for feedback; (3) Total number of feedback ($feedback\_count$) and (4) a value for the base (b), which is used to update the weight of unit size patterns.

In Line 1 of Algorithm 1, the sampler starts with any arbitrary frequent pattern P; a unit-length frequent pattern suffices. Then, it computes all frequent super and sub-patterns of P (Line 2). The degree of P is the size of the union set of super and sub-patterns. Then it chooses a pattern (Q) from P's neighbors (Line 5) uniformly and computes the acceptance probability in Line 8. If the move is accepted, Q becomes the resident state; otherwise, the sampler chooses another neighbor identically and repeats the whole process. If the user aborts the sampling, the infinite while loop breaks.

During the MCMC walk, if the condition in Line 13, i.e., the ratio of the number of nodes in the current POG and the same in the previously marked POG is greater than 3, the sampler inserts a set of random edges to the current POG. We use a variable called $randomedge\_factor$ to control the number of inserted random edges. The value of $randomedge\_factor$ is not that important as long as a small fraction of

Algorithm 1: Interactive_Sampling

Input: $\mathcal{D}, minsup, miniter, feedback\_count, b$

```
1  P = generate_a_frequent_pattern(D, minsup);
2  d_P = compute_degree(P);
3  π(P) = compute_β-value(P);
4  while true do
5      choose a neighbor, Q, uniformly from, all possible frequent super and sub
         patterns;
6      d_Q = compute_degree(Q);
7      π(Q) = compute_β-value(Q);
8      acceptance_probability = min ((π(Q)d_P)/(π(P)d_Q), 1);
9      if uniform(0,1) ≤ acceptance_probability then
10         P = Q;
11         iter = iter + 1;
12     end
13     if ((#ofNodeInCurrentPOG)/(#ofNodeInOldPOG) >= 3) then
14         insert_random_edge();
15     end
16     if iter % miniter == 0 and feedback_count ≥ 0 then
17         insert_random_edge();
18     else
19         goto line 5;
20     end
21 end
```

edges in the state-transition graph is from the random graph; in all our experiments, we keep this fraction equal to 0.20.

The condition on Line 16 checks whether the sampler should send the resident pattern (P) to the user for feedback. Inside *get_and_process_feedback* procedure,

Algorithm 2: compute_degree

Input: $\mathcal{D}, P, minsup$

```
1 super_pat = all_frequent_super_pattern(P);
2 sub_pat = all_sub_pattern(P);
3 neighbor_list = super_pat ∪ sub_pat;
4 save the neighbor list in POG data structure;
5 return size(neighbor_list);
```

Algorithm 3: compute_β-value

Input: P

```
1 foreach single length p in pattern P do
2 |   value = value * w[p];
3 end
4 return value;
```

Algorithm 4: get_and_process_feedback

Input: P,b

```
1  if feedback is positive then
2  |   foreach single length p in pattern P do
3  |   |   w[p] = w[p] * b;
4  |   end
5  else
6  |   foreach single length p in pattern P do
7  |   |   w[p] = w[p] * 1/b;
8  |   end
9  end
10 feedback_count = feedback_count − 1;
```

sampler first applies conditional periodic feedback scheme and checks whether getting feedback for the pattern in hand is advantageous or not. If the sampler finds it favorable for its learning, then it proceeds with the feedback. After that, if condition succeeds, depending on the feedback's status, the sampler updates the weight of each of the unit-size sub-patterns of P, which forces a change in the target distribution (β).

The above pseudo-code is generic and can easily be adapted for sampling any kind of patterns, e.g. itemsets, trees, sequences or graphs.

Example: Assume a random walk on the POG in Figure 3.2(b). At some stage of this walk, the resident state of the random walk is the pattern $g7$ (see Figure 2.4). It has 4 neighbors ($d_{g7} = 4$), one super-pattern ($g12$), two sub-patterns ($g1$ and $g2$), and a random edge. Assume that the existing weights (chosen arbitrarily) of the unit-size patterns $g1, g2, g3, g4$, and $g5$ are $\langle 2, 1, 2, 1, 1\rangle$ in that order. So, the weight of the pattern $g7$ is 2. If the proposal distribution chooses the pattern $g12$, to compute acceptance probability, we first compute the followings: β-score$(g12) = 4$, $d_{g12} = 4$; now the acceptance probability of the proposal move using equation 3.2 is: $\textbf{min}\left(\frac{4*(1/4)}{2*(1/4)}, 1\right) = 1$. So, this move will always be accepted. If we send $g12$ to the user and the user likes it, the weight of pattern $g1, g2$, and $g3$ will be incremented by the multiple of b and the new weight vector for the unit-size patterns will be $\langle 2 * b, b, 2 * b, 1, 1\rangle$. ■

Choosing Feedback Interval (*miniter*):

The effective sampling distribution β^t changes once a feedback on a pattern is received. So before returning a subsequent sample, the sampler needs to perform a few walks on the frequent pattern space—the parameter, *miniter* decides the number of walks that are interleaved between two successive feedback collections. Now, exactly how many walks the sampler should perform depends on how fast the random walk mixes, which we can compute from the spectral gap. Nonetheless, this computation

requires the knowledge of P, which the user is unaware of. On the other hand, the sampler can't pre-compute P as it requires the effective target distribution β^t, which is always changing based on the user's feedback. When *miniter* is set to a small value, the random walk exhausts all of its feedbacks before sufficiently exploring the pattern space which causes the sampler to converge to a distribution pre-maturely which is far from the true sampling distribution. Such a distribution allows the sampler to wander around the entire sample space which results in lower precision as many non-relevant patterns are visited. However, such a walk also covers a larger part of the search space, so the recall metric is high as the walk visits many of the relevant patterns as well. Similar phenomenon occurs for a large *miniter* also. In this case, the sampler spends more time on exploring between feedbacks and convergence of the true sampling distribution occurs slowly. Between two consecutive feedbacks, the sampler explores a large part of the frequent pattern space in which many patterns are non-relevant, which yields low precision. For this case, the walk covers a large part of the search space, so recall metric is high. It is hard to find an optimal value for *miniter*. We found that *miniter* in the range of $(20, 30)$ works well for most of the data-sets. We use *miniter*=25 for all of our experiments. In experiment section, we will show how the choice of *miniter* value affects the precision and recall.

Choosing Base (b):

The base (b) has a substantial significance in regard to learning the user's desired distribution, γ. A large value of b sharply increases (or decreases) the weight of a unit-length pattern in response to a positive (or negative) feedback. If we consider the learning of γ as a multi-arm bandit problem, each unit-length pattern can be considered as an arm, and we are trying to find a weight for each arm so that using this weight in equation 3.3, we can approximate the user's desired distribution γ. Under this assumption, the parameter b is simply the exploration-exploitation trade-off parameter. Choosing the high value of b imposes higher weights on unit-length

patterns that are part of some of the interesting patterns, which have already been explored; as a result, the sampler is biased to choose patterns that are composed of these items. However, this bias restricts the sampler to explore other yet-to-be-explored patterns that may be interesting but are not composed of unit-length patterns that have already been explored. For this reason, a higher value for b increases the sampler's precision since the sampler will traverse patterns that are composed of (already known) interesting unit-length patterns; however, this yields poor recall as many other interesting patterns may remain unexplored. It is hard to choose a good operation point on the precision-recall curve because in a hidden data-set the user has no idea of the number of interesting patterns that exist in the hidden data-set—a value that is needed to compute the recall. We found that b in the range of $(1, 2]$ works well for most of the data-sets. We use b=1.75 for all our experiments.

Choosing Threshold for Percentage of Divergence(η):

Choosing a proper η is challenging. To set the divergence parameter we adopt the trial-and-error approach. We perform numerous empirical tests on different threshold values ranging from 5% to 50% over different data-sets. We found that percentage of divergence(η) set to 20% works for all data-sets.

3.5.7 Mining Patterns from Hidden Datasets

To mine a pattern from a hidden data-set, a user interface is used through which the user provides her feedback. At the initiation of an interactive mining session with a user u_i, the interactive system creates a new sampler process (say, s_i) that runs on the host machine (that stores the data-set) for managing dedicated communication between u_i and s_i (see Figure 3.1(b)). The sampler executes the interactive mining algorithm that is shown in Figure 1; the parameters of this algorithm, such as *minsup* can be obtained from the user. Through the user interface, the sampler sends a pattern to the user, gathers her feedback and updates the sampling distribution by

executing the method in Line 15 of Figure 1. The infinite loop terminates once the Connection is closed by the user. The above setup enables interactive pattern mining by considering a semi-honest model [72], where each party obeys the protocol and the user of the system cannot be whimsical, i.e. change interest over patterns during an execution.

3.6 Experiments & Results

We conduct several experiments to manifest the main purpose of interactive pattern sampling, which is to mine a set of interesting patterns by utilizing the binary feedback from the user. Our experiments show results both from graph and itemset mining.

3.6.1 Experiment Setup

For our experiment, we require defining an interestingness function (γ) by which the user makes an opinion about a pattern. We mark $m\%$ randomly selected unit-size patterns among all the frequent (for a given *minsup*) unit-size patterns as interesting according to a user. With these $m\%$ patterns, we build a set called I_p. We assume that the user is interested in patterns where the majority of the single length sub-patterns are from the set I_p. Thus, the scoring function, γ for a pattern P is defined as, $\gamma(P) = \frac{\sum_{i=1}^{k} I(p_i)}{k} \ * \ 100$

Here, $p_1, p_2, \ldots, p_k$ are the unit-length sub-patterns of P, and I is an indicator random variable that defines whether a pattern p_i is from the set I_p or not. The γ-score of a pattern varies between 0 and 100, and the patterns that have γ-score more than 50 are interesting to the user, for which the user provides a positive feedback; for the remaining patterns, the user provides a negative feedback. We set $m = 30\%$ in all of our experiments, unless specified otherwise.

Using the similar notion of defining an interestingness function (γ_g) explained above, for graph data, we mark $L\%$ randomly selected induced subgraph of size 3

and 4 as interesting according to the user and called it I_{gp}. We assume a user will be interested in those graph patterns that have majority induced subgraphs from I_{gp}. For graph pattern, final γ_g-score will be the average of $\gamma(P)$ computed from I_p and I_{gp}. In all our experiments, we set $L = 50\%$.

Table 3.2
Dataset statistics

Dataset	# of transactions	Avg. transaction size	$\pi^{\mathbf{min}}$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{F}_1\rvert$
Chess	3196	37	2100	99262	29
Mushroom	8124	22	1200	53628	45
Connect	67557	43	58000	109664	24
eBay Query	10000	27	30	41130	3368
HIV-1	9	474.4	6	58154	39
Mutagenicity-II	687	14	20	2935	13
Biodegradability	328	11	16	1820	8

3.6.2 Dataset

We show experimental results on seven real-life data-sets displayed in Table 3.2. The first five among them are itemset data-sets, and the remaining two are graph data-sets. We obtain Mushroom, Chess and Connect data-set from the UCI Machine Learning repository[6]. *eBay Query* is the remaining itemset data-set that we generate by scrapping queries from the *"Related Searches"*[7] a feature of eBay. We build a query-graph where related queries are neighbours. Afterwards, we populate the row of an itemset data-set by performing short random walks of this graph; In this way,

[6] `http://archive.ics.uci.edu/ml/`

[7] When a user searches for an item in `www.ebay.com` with a query phrase, the search result page shows (at most) 10 keywords related to the original query.

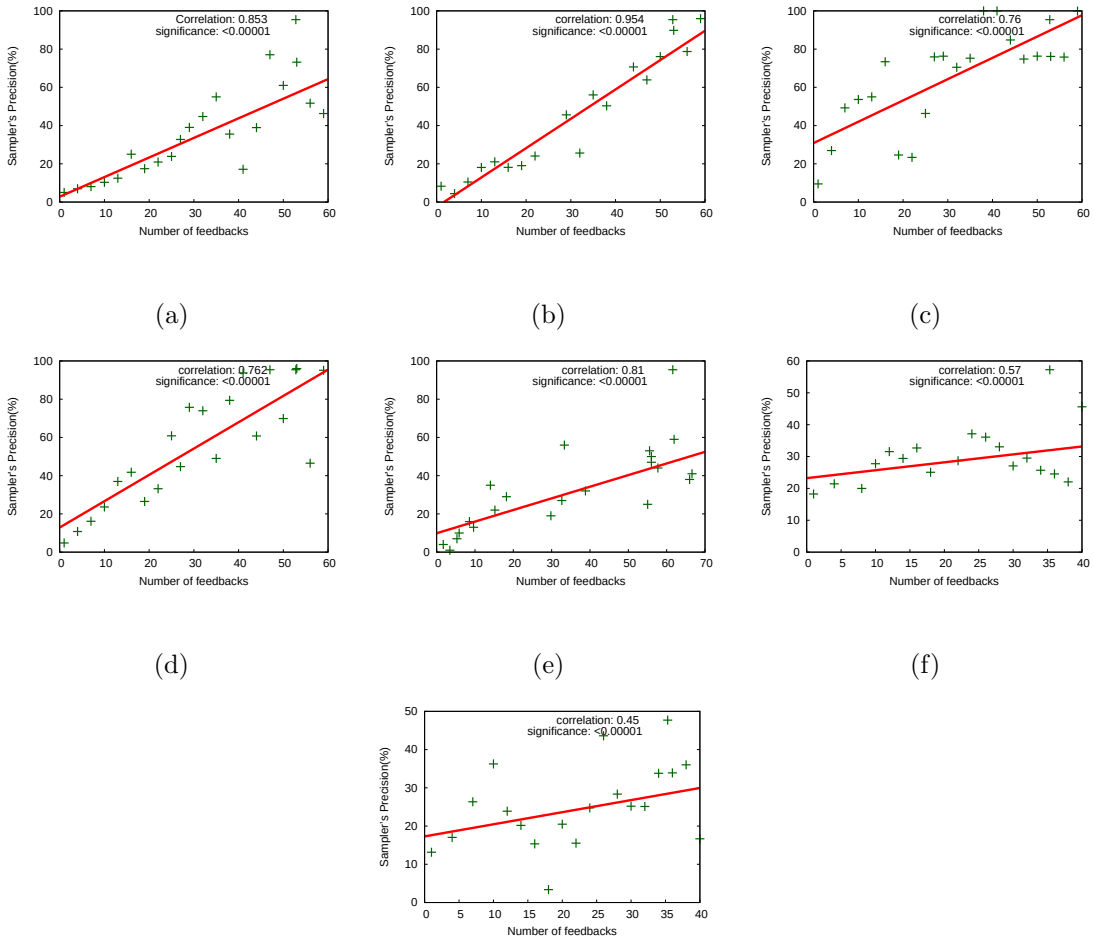


Fig. 3.4. Scatter-plot with a fitted straight line showing the relationship between the sampler's precision and the number of feedback in (a) Mushroom (b) Chess (c) eBay (d) Connect (e) HIV (f) Biodegradability (g) Mutagenicity-II with Periodic Feedback.

the list of the queries that such a walk visit makes a transaction in the eBay Query data-set. Since related search suggestion is built by using user session data [45], this data-set is a metaphorical user session data, where a transaction can be regarded as the set of queries that a user executes in a user session.

We also build a real-life transaction data-set from a life science domain. For this, we have collected HIV-1, human protein interaction data from HIV-1, Human-protein

interaction Database[8]. This data-set contains the information of nine HIV-1 proteins interacting with proteins in the host(Human). Each record in this data represents an interaction of an HIV-1 protein and a Human protein. There are 9 HIV-1 proteins and 3450 Human proteins. We convert this data into transactional itemset data. We assume each HIV-1 protein as a transaction and all interacting human proteins as items in the transaction.

The remaining two data-sets, Biodegradability and Mutagenicity-II are chemical (graph) data-sets that we obtained from the authors of [73].

Table 3.2 also shows some statistics of the data-sets. The second and third column shows the number of transactions and average size of each transaction. The fifth column shows the number of frequent patterns, considering the minimum support value listed on the fourth column. The sixth column shows the total number of distinct unit-size frequent patterns.

3.6.3 Analysis of Sampler's Performance

The objective of this set of experiments is to show the dependency of sampler's convergence with various parameters of the interactive sampling. We first show that with the increasing number of feedback, the sampler converges to a distribution, where the interesting patterns are more likely to be sampled. For this, we run the sampler for a large number of iterations and compute the percentage of iterations in which the sampler visits an interesting pattern; we call this percentage *samplers precision* (SP). We compute SP from many independent runs, each using a different value for the $feedback\_count$. Then we find the correlation between the number of feedback and SP. Since the process is randomized, SP value for each $feedback\_count$ is also computed by averaging over 10 runs.

We show our results in Figure 3.4. As we can see, the number of feedback and the sampler's precision are highly correlated—SP increases along with the feedback

[8] `http://www.ncbi.nlm.nih.gov/projects/RefSeq/HIVInteractions/`

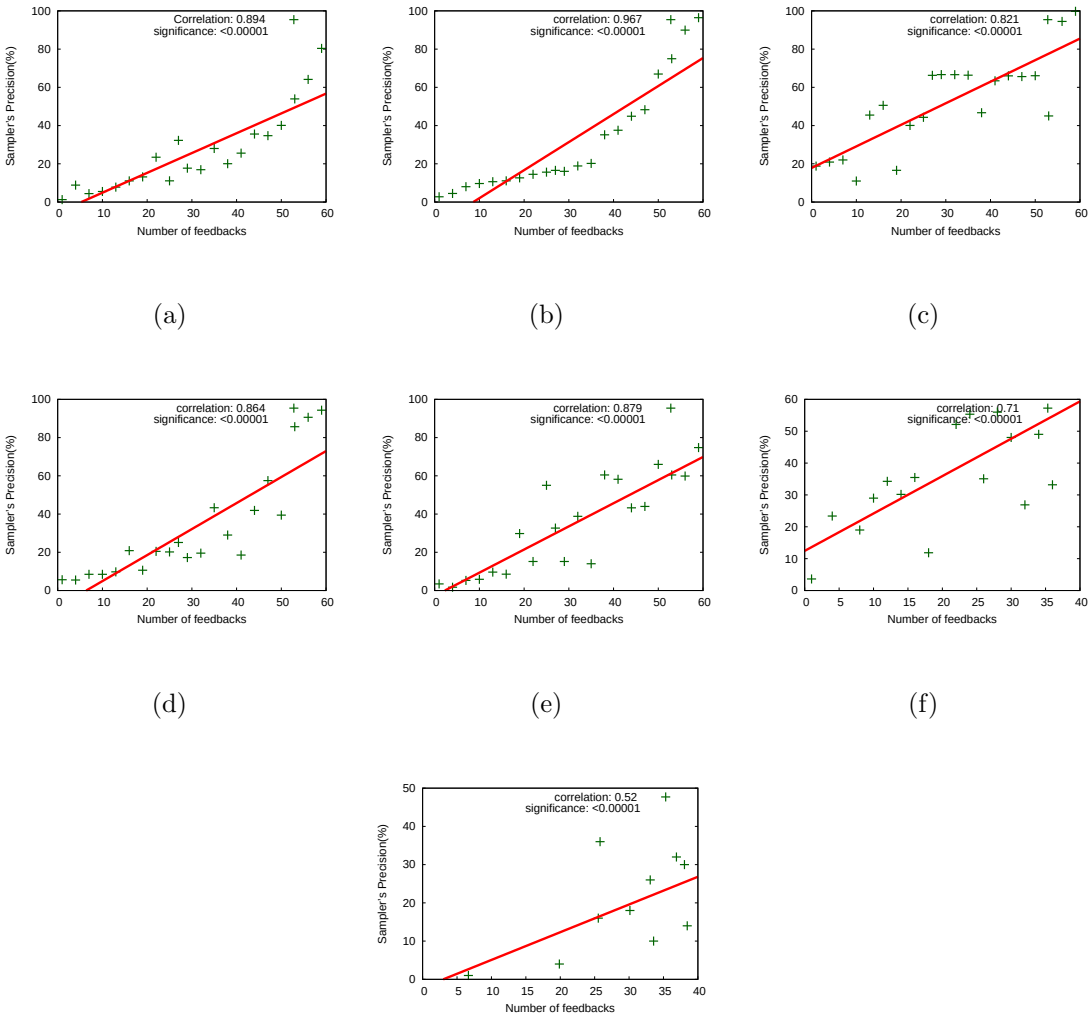


Fig. 3.5. Scatter-plot with a fitted straight line showing the relationship between the sampler's precision and the number of feedback in (a) Mushroom (b) Chess (c) eBay (d) Connect (e) HIV (f) Biodegradability and (g) Mutagenicity-II with "Conditional Periodic" Feedback scheme.

count. Note that, the strength of correlation differs across data-sets—specifically, the correlations are poor for graph data-sets, and good for itemset data-sets. This is expected because, in this experiment, we use γ as our scoring function, which treats

a graph as if it is simply a multiset of labeled edges (unit-pattern)—this assumption ignores the neighborhood structure of a graph, which may degrade the sampling quality. Another important observation is that, it does not take much feedback for the sampler to achieve a fairly good correlation. In all the plots, the maximum number of feedback that we use is less than 60, which is a negligible fraction of the entire frequent pattern set—see the fifth column of Table 3.2 for a count of frequent patterns in each of the data-sets. 60 is also a reasonable number considering the real-life scenario, where a human inspects the patterns and provides feedback on them.

To show the effect of b, the exploration-exploitation trade-off parameter, we run our sampling method for several b values and report average samplers precision and recall. Here the precision is as we defined earlier, and the recall is the fraction of interesting patterns that the sampler has explored out of all the interesting patterns. The result is shown in Table 3.3, where the precision and recall are averaged over 10 distinct runs; as we expect, an increment of b increases the sampler's precision with a reduction in the recall value. We also show the F-measure value, which is the harmonic mean of precision and recall. For $b = 1.75$, this value is the largest among the four choices that are shown in this Table.

To exhibit the influence of parameter *miniter*, we run our sampling algorithm for different *miniter* values and report average sampler's precision and recall as we did for the experimentation with the parameter b. We present the result in Table 3.4,where the precision and recall are averaged over 10 distinct runs. As discussed earlier in Section 3.5.6, we can see that both for larger and smaller value of *miniter*, the sampler suffers from low precision but enjoys a high recall. We also show F-Measure value for different choices of *miniter*. As we can see the largest F-measure is achieved for *miniter* equal to 5, but the precision for that choice is only 46.43%. For a hidden data-set, a user generally has no idea about the total size of the relevant patterns in the data-set, so he will not be able to estimate the recall. However, he will be able to compute the precision from the patterns that the system returns. This is

the justification for choosing *miniter* equal to 25 in all of our experiments as a value around 25 maximize the precision for most of the data-sets.

To show the effects of different values of divergence parameter η, we run our algorithm on Mushroom data-set for several times and report average sampler's precision and recall as we did for the experimentation with the parameter b and *miniter*. In Table 3.5, we show our results. As we can see, with the increase of the percentage of divergence, η, sampler's performance decreases as expected. With higher η, the sampler will only be able to ask for feedback for a very small number of patterns, hence performance decrease. On the other hand, as we lower the value of η, sampler asks feedback for patterns discarding the effect of knowledge gain that comes along with the feedback, which results in a decrease in the sampler's performance.

Table 3.3
Effects of parameter b on Mushroom data-set

base(b)	Avg sampler's precision(%)	Avg recall (%)	F-Measure
1.5	62.43	41.70	50.00
1.75	80.50	36.61	50.33
2	85.61	23.43	36.79
2.5	97.96	9.6	17.48

Table 3.4
Effects of parameter *miniter* on Mushroom data-set

miniter	Avg sampler's precision(%)	Avg recall (%)	F-Measure
5	46.43	77.23	57.99
25	80.50	36.61	50.33
50	72.91	41.43	52.83
250	40.53	85.12	54.91

Table 3.5
Effects of parameter percentage of divergence (η) on Mushroom data-set

percentage of divergence (η)	Avg sampler's precision(%)	Avg. recall (%)	F-Measure
5%	70.21	40.13	51.07
10%	75.34	37.32	49.91
20%	80.50	36.61	50.33
40%	35.40	68.15	46.58
50%	30.12	79.22	43.64

We also study, how the sampler's performance changes, as the fraction of interesting patterns in the frequent pattern set ($\mathcal{F}$) changes. For this, we vary the parameter m which defines what fraction of unit-size items are interesting to the user. For this experiment, we use the Mushroom data-set with a support value of 1200. The number of frequent patterns, i.e., $|\mathcal{F}| = 53628$ and the number of unit-size patterns ($\mathcal{F}_1$) is = 45. For $m = 10\%$, on average, 5 unit-size patterns from the set $\mathcal{F}_1$ will be chosen as interesting, and the number of interesting patterns is 56, on average. Sampler's precision for this m value is 5.8%, i.e. out of 1000 sampled patterns, 58 patterns are interesting. This seems to be a low number, but considering the fact that a uniform sampler will pick one interesting pattern from every thousand samples ($56 * 1000/53628 \sim 1$), the interactive sampler is 58 times more efficient. For $m = 20\%, 30\%, 40\%$, and 50%, the average sampler's precision is 26.5%, 56.7%, 63.2% and 74%, respectively. Sampler's precision increases with the increment of m, as there are more interesting patterns in the search space ($\mathcal{F}$). Results on other data-sets are similar, which are not shown due to the page restriction.

3.6.4 Interactive vs Uniform Sampling

Does user interaction really help? If we let the sampler perform a uniform sampling, will that achieve as a good result as an interactive sampling session? We somewhat answer the question in the last paragraph of the earlier subsection, here we answer it in more details. For this, we compare interactive sampling and uniform sampling. Note that, for uniform sampling, every pattern in $\mathcal{F}$ has equal probability to be sampled, i.e. the γ-score is $\frac{1}{|\mathcal{F}|}$ for all the patterns. In our implementation of interactive sampling, we simply turned off the feedback and use the above γ to obtain a version of MCMC sampling that would obtain a uniform sampling. For interactive sampling, we fix the $feedback\_count$ to 60 for the itemset data-sets i.e. Mushroom, Chess, Connect, eBay and 40 for the graph data-sets, i.e. Biodegradability and Mutagenicity-II. Our comparison metric in this experiment is sampler's precision, which is defined in the description of the previous experiment. The result is shown in Table 3.6; for all the data-sets the interactive sampling significantly improves the sampler's precision. We use Conditional Periodic feedback in this experiment. Hasan et al. [52] propose the uniform sampling of frequent patterns as a mechanism of frequent pattern summarization. However, the above experiment shows that while mining patterns from a hidden data, an interactive approach will provide more rewarding sampling experience. We could not be able to perform any accuracy experiment of One-click method by [10] because, in the chapter, the authors did not include any empirical experimentation to measure accuracy/efficiency of their method.

3.6.5 Analysis of Sampler's Performance for Conditional Periodic Feedback Mechanism

In Section 3.5.4, we explain how the sampler uses KL-Divergence to decide whether, for a given pattern, it will seek feedback. In this experiment, we want to empirically show that such scheme of gathering feedback is advantageous. Experiments described

Table 3.6
Average sampler's precision measure of uniform and interactive sampling.

Dataset	Sampler's Precision(%)	
	Uniform	Interactive
Mushroom	0.182	80.50
Chess	2.293	96.43
Connect	4.53	98.34
eBay Query	15.23	99.78
HIV-1	2.23	74.10
Mutagenicity-II	1.15	51.27
Biodegradability	1.247	74.64

Table 3.7
Comparisons of average sampler's precision measure while considering conditional periodic feedback VS periodic feedback

Dataset	Sampler's Precision(%)	
	Conditional Periodic Feedback	Periodic Feedback
Mushroom	80.50	56.70
Chess	96.43	95.74
Connect	98.34	94.32
eBay Query	99.78	98.50
HIV-1	74.10	65.23
Mutagenicity-II	51.27	35.00
Biodegradability	74.64	44.05

in Section 3.6.3, computes the correlation between SP and $feedback\_count$ in order to exhibit sampler's performance of learning the user's interestingness function. We also use the same correlation measure to demonstrate the sampler's performance with the improved feedback gathering scheme. The setup of the experiment is similar as in Section 3.6.3, where we run sampler multiple times for a large number of iterations for various $feedback\_count$. At the end of one run, we compute SP and find the final SP by averaging over-all runs. Finally, we compute the correlation between $feedback\_count$ and SP. In Table 3.7, we showcase the performance of a sampler when using the improved feedback gathering scheme against the scheme where the sampler receives feedback in a periodic manner. In Figure 3.5, we present the scatter plots of $feedback\_count$ and SP with a fitted straight line exhibiting correlations for different data-sets.

Table 3.8
Comparisons of average sampler's precision measure while considering topological information in scoring function and not.

Dataset	Sampler's Precision(%)	
	Topological information	No Topological information
Mutagenicity-II	51.27	45.66
Biodegradability	74.64	57.13

3.6.6 Analysis of Sampler's Performance with Updated Scoring Function for Graph Data

In this experiment, we show empirically that incorporation of topological information while defining a user's interestingness for graph pattern is favorable for the sampler. The experimental setup is similar as in Section 3.6.3. We run sampler for multiple times with different $feedback\_count$ and compute average SP with graph

data. Note that, we use Conditional Periodic feedback scheme in every execution. Table 3.8 supports the fact that topological information can make a positive impact on sampling performance when dealing with the graph data. Figure 3.5(f)(g) illustrates improved correlation because of updated scoring function for graph data.

An important observation for most of the data-sets is, there exist few frequent graph patterns, those have induced subgraph that are complex in nature, i.e. g_7 and g_8 (figure 3.3). Based on such observation, we decide to omit g_7 and g_8 from the weight vector of induced subgraphs of size 3 and 4.

3.6.7 Timing Analysis

In this experiment, we analyze the execution time of this algorithm for varying minimum support and data-set's size[9]. We run our algorithm on Mutagenicity-II data-set with various minimum supports and different database sizes and compute the total time interval between 20 consecutive positive feedback. We conduct 10 independent runs and take the average value of the total time interval. Figure 3.6(a-d) shows that, as expected, the time between consecutive positive feedback increases with an increase in the database size or a decrease in the support value; for example, a user's waiting time between successive positive feedback is 9.4 (188/20) seconds for 5% support, but it is just 4.2 seconds for 10% support for graph data-set(Mutagenicity-II, Figure 3.6(a,b)). However, the reader should note that, the majority of the time is spent in computing the state-transition graph for MH-based subgraph sampling, so for larger data-set, the sampler could pre-compute the state-transition graph (for typical support threshold) and save it for future use; in that situation, the timing performance of the interactive sampling will improve substantially. We conduct an experiment with Mutagenicity-II and Mushroom data-set. Without pre-computation, the execution of 1 million iterations on Mutagenicity-II data-set takes 1676 seconds,

[9]This experiment is performed in a 2GHz dual-core machine with a 2 GB physical memory.

but the same job executes in only 230 seconds if we pre-compute the state-transition graph.

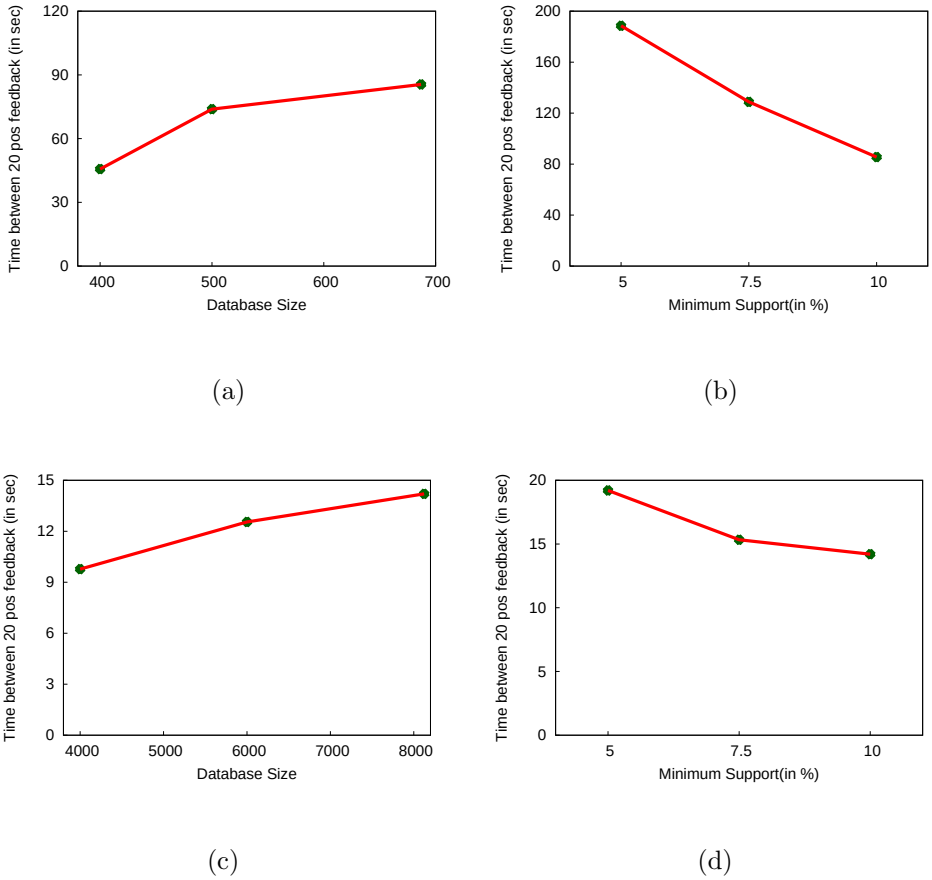


Fig. 3.6. Timing performance of IPM w.r.t different database size and minimum support for Mutagenicity-II(Graph) and Mushroom(Itemset) data-set

3.6.8 Empirical Evaluation of Disclosure of Hidden Itemset Dataset through IPM

In this section, we conduct an experiment to show whether the proposed interactive pattern Mining (IPM) system discloses sufficient information for an attacker to reconstruct the hidden itemset data-set. For this experiment, we use Chess and

Mushroom data-set. Before presenting the experimental results, we first discuss the attack model that an adversary uses for reconstructing the hidden data-set.

Attack Model: A Generative Process for Transaction Data Reconstruction

In this attack model, we assume that the hidden itemset data-set has a generative distribution $Dist$, and an adversary tries to approximate the distribution by using the released patterns. If the approximate distribution is $Dist'$, the data-set reconstruction risk can be estimated by the similarity between $Dist$ and $Dist'$.

In literature, several mixture distributions [74–76] have been proposed to model binary representation of an itemset data-set. In this work, we use the Bernoulli mixture model [74] for its simplicity and generality. The Bernoulli mixture model assumes that there exist latent types that control the distribution of the items in a transaction, and for a given type, the items in a transaction are independent. If we have K latent types, the transaction generating distribution can be expressed as a multinomial distribution with K parameters, $(\pi_1, \pi_2, \cdots, \pi_K)$ satisfying, $\sum_{k=1}^{K} \pi_k = 1$. Let, $\mathbf{Z} = (z^1, z^2, \cdots, z^N)$ is a type indicator, which represents the type of a transaction; i.e. $z^i = k$ states that the transaction i is of latent type k. Using the independence among items within a transaction for a given type, the probability of a transaction, $T_i = \{I_1, I_2, \cdots, I_{|\mathcal{I}|}\}$ is given by the following mixture model.

$$p(T_i|\theta) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{|\mathcal{I}|} p(I_j|(z^i = k), \theta) \tag{3.8}$$

Here, θ represents all the parameters of the model and $|\mathcal{I}|$ is the number of items in the data-set. Since each transaction is a binary vector, the conditional probability of appearance of an item I_j given a type k in a transaction T_i follows a Bernoulli distribution with success probability μ_{kj}:

$$p(I_j|(z^i = k), \theta) = \mu_{kj}^{t_j}(1 - \mu_{kj})^{1-t_j} \tag{3.9}$$

Here, $t_j \in \{0, 1\}$ indicates whether the item I_j exists in the transaction or not. The Parameters can be easily computed using Expectation-Maximization (EM) [76] algorithm.

An adversary, who knows that the transaction data follows a mixture of Bernoulli can attempt to estimate the parameters of the mixture distribution using the released patterns. Assuming $Dist$ is the original data distribution, and $Dist'$ is the constructed data distribution using the released set of patterns. In a successful reconstruction, l2-norm between the parameters of $Dist$ and $Dist'$ should be close to zero. In this experiment, we determine whether the probability of the l2-norm between the parameters of $Dist$ and $Dist'$ (l2-norm($Dist, Dist'$)) falls within a non-tolerable interval from 0 is bounded by ρ. We expect this ρ to be very small. Note that, during the execution of the proposed system the data owner can use this parameter ρ to express the level of entire data privacy risk he is willing to tolerate. Low and high value of ρ indicates tighter and relaxed security configuration of the IPM system.

For a set of released patterns by IPM, suppose X is a random variable representing the l2-norm($Dist, Dist'$). μ and σ^2 are the mean and variance of X, respectively. According to the one-sided Chebyshev-Cantelli's [77] inequality constraint, we can write

$$P(X \leq \mu - a) \leq \frac{\sigma^2}{\sigma^2 + a^2}, \text{where } a > 0.$$

To measure an adversary's effort on compromising privacy, we compute

$$P(X \leq 0) = P(X \leq \mu - \mu) \leq \frac{\sigma^2}{\sigma^2 + \mu^2}, \text{assuming } a = \mu.$$

In this experiment, we want to show that the exact reconstruction, $P(X \leq 0) \leq \rho$, where ρ is a small number.

We design two types of attackers here. Attacker-1 considers each released pattern as a transaction and attempts to reconstruct the true data distribution $Dist$ using the patterns directly. On the other hand, Attacker-2 knows the distinction between transaction and pattern space. For each released pattern p, attacker-2 can retrieve top 5 original transactions t_is for which $coverage(p, t_i) >= 0.8$, where $coverage(p, t_i) =$

$|p \cap t_i|/|t_i|$ and $i \in \{1 \cdots 5\}$. Attacker 2 attempts reconstruction using the retrieved original transactions. In the following section, we will discuss the performance of both types of attackers for Mushroom and Chess data-set.

Table 3.9
Chebyshev-Cantelli's probability bound on Attacker 2's data-set exact reconstruction performance measured by l2-norm. *DP=Data Points

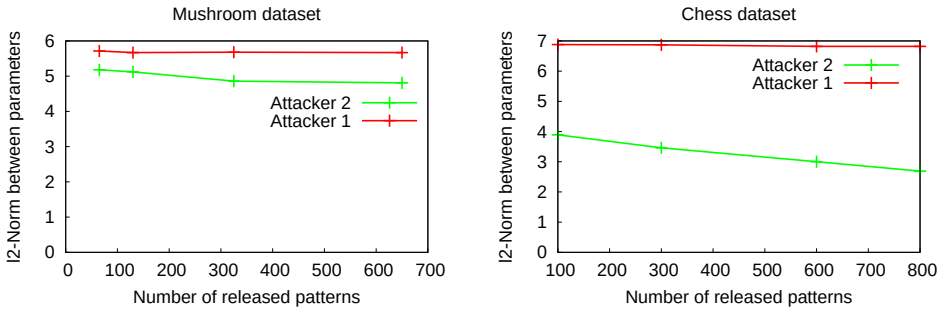
Mushroom				Chess			
DP*	Mean (μ)	Var (σ^2)	$P(X \leq 0)$	DP*	Mean (μ)	Var (σ^2)	$P(X \leq 0)$
65	5.18	0.383	0.0054	100	3.89	0.263	0.0045
130	5.12	0.289	0.0031	300	3.46	0.513	0.0215
325	4.86	0.263	0.0029	600	3.00	0.427	0.0197
650	4.81	0.296	0.0037	800	2.69	0.526	0.0368

Estimation of Disclosure Risk

In this experiment, we show an empirical estimation of the two types of attackers (discussed in Section 3.6.8) performance on the disclosure of the hidden itemset data-set. Specifically, we want to show how an attacker's ability to disclose the data-set's privacy changes as the number of released patterns increase. As discussed in Section 3.6.8, the data owner has a true mixture of Bernoulli distribution $Dist$ for the data-set, and the attacker constructs $Dist'$ from the released patterns. To simulate the attack model, we use all the patterns that are released by our method during feedback, and also the delivered patterns according to the user's interest in a session. We use l2-norm between parameter vector of $Dist$ and $Dist'$ to measure attacker's performance on disclosing overall data-set's privacy. We perform this experiment using the Mushroom and Chess data-set for the different number of released set of patterns. We tried several values of K (No. of mixture components in equation 3.8)

and found that for $K = 3$, reconstruction performance was comparatively better. In the x-axis of Figure 3.7(a)(b), we plot the number of released patterns and in the y-axis, we show the l2-norm($Dist, Dist'$) average over 30 runs. We use different colors to plot attacker 1 and 2's performance. For both data-sets, attacker-2's approach is more effective than attacker-1. As we can see, the l2-norm decreases as we increase the number of released patterns for attacker-2 but for attacker-1, it remains at the same level.

In Table 3.9, we show the calculated upper bound of $P(X \leq 0)$ (Exact reconstructing), where X is the random variable representing attacker 2's performance (measured l2-norm) on different executions of the algorithm. We show separate calculation for Mushroom and Chess data-set in the Table. In 4th and 8th column of Table 3.9, we present the calculated bound using $P(X \leq 0) = P(X \leq \mu - \mu) \leq \frac{\sigma^2}{\sigma^2+\mu^2}$ as per the discussion in Section 3.6.8. As we can see for both the data-sets, $P(X \leq 0)$ is significantly lower.



(a) (b)

Fig. 3.7. L2-norm between parameters of $Dist$ and $Dist'$ with the increasing number of released patterns over (a) Mushroom (b) Chess data-set

3.6.9 Real-life Utility of Interactive Mining

Results from eBay Datasets:

In this experiment, the user imitates a seller at the eBay marketplace, who is interested to find related queries in the smart-phone, tablet, and game console categories. To accomplish this, the seller performs an interactive pattern mining on the eBay query data-set by providing positive feedback on frequent patterns that are related to the above product categories; some of the frequent patterns that she samples from the hidden user session data-set are shown in Table 3.10.

Table 3.10
Some frequent patterns from eBay query dataset

google tablet, android tablet, tablet pc 10
iPhone 4, iPhone 3gs unlocked, iPhone 5s clean esn
samsung galaxy s, samsung vibrant, samsung captivate
nexus 5 unlocked, nexus 5 rooted, iPhone 5S jailbreak
ASUS Eee Pad, ASUS Transformer Prime

These patterns can educate her about the buyer demand trend at the eBay marketplace; further, they could assist her in choosing a more rewarding title for her product, which would boost her chance to market it. At eBay, a seller's item in surfaced on the search result page if the title of the item contains *all* the words in a user's query[10]. If the seller wants to sell a `samsung galaxy s` smart-phone, the knowledge of the pattern in the third row of Table 3.10 would encourage her to add the words `vibrant` and `captivate` with the title `samsung galaxy s`, so that her product would also surface against the queries like `samsung vibrant` and `samsung captivate`. These two words are chosen by the US cell phone career T-mobile and AT&T to market the `samsung galaxy s` phone with their phone plans. Similarly, a

[10]however, eBay allows 55 characters for the title, so experience sellers always optimize the title of their product

title like, `google tablet android` would match with more user queries than those that would match with the title `google tablet`. Equivalently, for a used iPhone-5s title like `iPhone 5s clean esn` would attract more buyers since the keyword "clean esn" establishes the fact that the iPhone has never been reported, lost or stolen.

If a seller wants to sell unlocked iOS or android based smart-phones, she has to know that for android devices, i.e. nexus 5 keyword `rooted` is being used equally to `jailbreak` for iOS devices, i.e. iPhones. Furthermore, putting only `unlocked` does not necessarily mean that the device is `rooted` or `jailbreaked`. "Rooting" gives access to the user of an android phone for altering and installing specialized applications, whereas "Jail-breaking" gives iPhone user access to install non-officially approved applications. Set of patterns presented in the fourth row of Table 3.10 certainly helps seller to add keywords `rooted`, `jailbreaked` in their product. Finally, patterns in the fifth row of Table 3.10 make seller knowledgeable about the fact that, `ASUS Eee Pad` is also known as `Transformer Prime`.

One can argue that, is it possible to discover above-mentioned patterns using existing off the shelf pattern mining/sampling techniques? The answer is, it is possible, but with hundreds of other patterns, which might not be as interesting to the user. Since using our technique, the user can explicitly specify what she likes as the sampling continues; the output patterns she receives will mostly consist of her choice of patterns. Moreover, the interestingness of the user is adhoc (in this case only likes smart-phone, tablet, and game console). It will be difficult to come up with an interestingness function satisfying this specific choice of patterns on the fly for existing pattern mining or sampling algorithm to use. In summary, by educating a seller through interactive pattern mining on the hidden user session data-set, the marketplace can increase sales transactions and eventually can boost their own revenue.

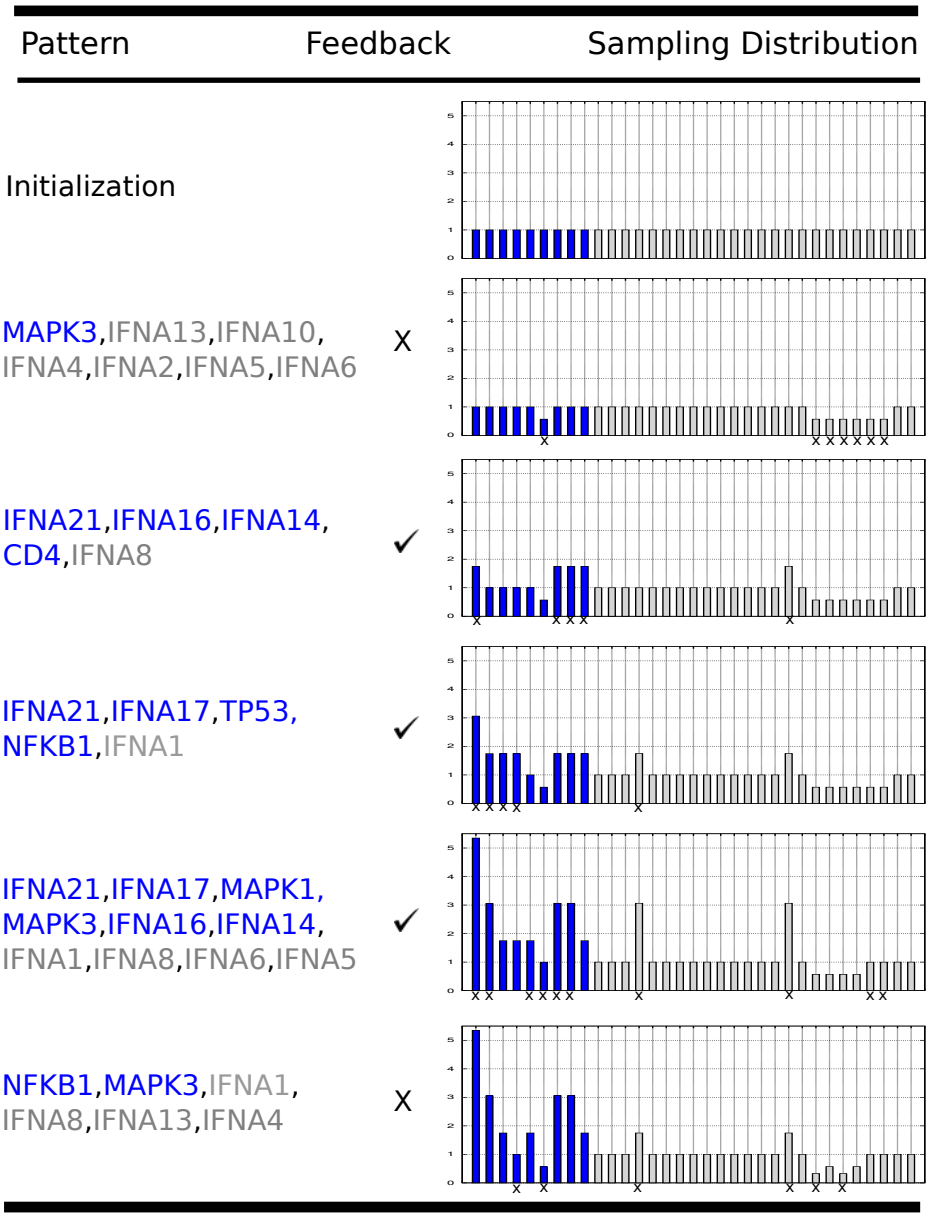


Fig. 3.8. Changing of sampling distribution with feedback for HIV-1 dataset

3.6.10 Illustration of Interactiveness over HIV-1 Dataset

In this section, we present another case study of our proposed algorithm using HIV-1 data-set. The motivation of this case study is to visualize the effect of interactiveness of our algorithm. We want to show how the sampling distribution changes

with feedback. For this, we build a compact version of HIV-1 data-set (mentioned in Section 3.6.2), where first we pick 33 human proteins out of 3450 that are interacting with nine HIV-1 proteins through regulation, i.e. up-regulation, down-regulation. Then we remove all other human proteins from each transaction of the original HIV-1 data-set those are not part of these 33 human proteins. At the end of the removal process, we have a compact version of HIV-1 data-set. To carry out the demonstration, we select 8 human proteins out of 33 randomly and mark as interesting for a user. In Figure 3.8, we use blue color text to identify these 8 proteins. Each row in Figure 3.8 shows the current sampled pattern, its feedback and the updated sampling distribution. We use X and $\surd$ to denote negative and positive feedback respectively.

The sampling process starts with a uniform distribution where each of the 33 items(human proteins) have the same score (1). As the sampling process continues and sampler collects feedback for the nominated pattern, the score of each human protein got updated hence the sampling distribution. Column 3 of Figure 3.8 shows the distributions. We use "x" mark across the x-axis of each distribution bar plots to denote the human proteins of the sampled pattern for which sampler asks feedback. For better visualization, we cluster 8 human proteins on the left of the bar plot and drawn in blue. Rests are on the right side and drawn in Grey. To decide what should be the nature of the feedback, i.e. positive or negative, we identify what percentage of items are from interesting item sets, i.e. 8 human proteins. If the percentage is more than 50%, we assume feedback to be positive else negative. We can see in Figure 3.8 that the score of interesting items' increases as the sampler receives positive feedback. Even though sampler starts with a negative feedback, it catches up as the positive feedback comes. Finally, we can see that, with-in few numbers of the feedback, sampler converges to a distribution, where the scores of interesting human proteins are comparatively higher than non-interesting proteins.

3.7 Discussion

This is a novel research problem, and our work is just the beginning, so the opportunity for future work is abundant. In this section, we will discuss some of the drawbacks of the proposed system and possible forthcoming works.

3.7.1 Safeguarding the System from Manipulative User

In this work, we have considered a semi-honest model whereas in real-life a malicious model may be needed. Specifically, an interesting follow-up research could be to understand how to safeguard the interactive mining system from a malicious user who intends to reconstruct the entire data-set by exploiting the sampling system. Even more interesting is the scenario when a group of malicious users jointly attack the system to achieve a similar goal.

3.7.2 Advanced Feedback Mechanism

Our work uses the binary feedback; an important future work is to consider a richer interaction model between the sampler and the user. One option is that the sampler returns a set of (random) patterns, and the user provide a preference-based ranking of those patterns as her feedback. Afterwards the sampler updates the sampling distribution by utilizing this feedback. Another interesting future work on the user's feedback can be to handle situation where a user is not sure to provide positive/negative feedback.

3.7.3 Making Learning and Sampling Phase Independent

In this work, the user interacts with a Markov Chain Monte Carlo (MCMC) sampling entity that samples frequent patterns from the hidden data. The target distribution of the sampler is iteratively refined based on binary feedback from the user. While the overall idea is novel, this solution has a crucial drawback-the learning

and the sampling are unified, which causes the sampler to converge to a sub-optimal distribution prematurely; hence the sampled patterns, sometimes, are obtained from a distribution which is not close to the true distribution. One possible future work in this direction would be to make sampling and learning phase independent from one another.

3.7.4 Robust Data Privacy Model

In this work, our proposed system assumes that the released patterns are not sensitive. It ensures that the data owner maintains the confidentiality of the data in the sense that using the released set of patterns a user cannot reconstruct the entire data-set with a high level of accuracy. In practice, such assumption might not be compatible to ensure the privacy for all kinds of private data. For example, in financial or medical data, patterns can contain sensitive information, hence prone to privacy violation for individuals in the data. As a future work, we want to extend our data privacy model to ensure both entire data-set's as well as individuals confidentiality.

3.8 Conclusion

Mining interesting patterns from a hidden data-set is an important research task; an effective solution to this task brings information to its end-user, thus democratize the information for its better usages. In this chapter, we show examples from an eCommerce domain, but such a solution will be useful in many application domains; for instance, in health informatics research, private mining of patient health record located at a remote server will enable a large number of researchers to find interesting patterns involving diseases, and medicines.

To conclude, in this chapter, we introduce a new research problem, i.e., mining frequent pattern from a hidden data-set through interactive pattern sampling. We formulate a solution to this problem by adopting interactive pattern sampling framework, where the effective sampling distribution is continuously updated by binary

feedback from the user. We provide real-life examples to illustrate the usefulness of this research task and also show experimental results to validate the feasibility and effectiveness of our solution. Specifically, we show that the number of feedback that a user is required to provide is practical for a real-life scenario.

4. PATTERN2VEC: NEURAL-NET BASED FEATURE REPRESENTATION LEARNING OF COMPLEX FREQUENT PATTERNS

Feature representation learning is an active line of research now-a-days. Majority of the works into this direction are motivated by the traditional or deep neural networks. Researchers were able to achieve superior performance in the domain of text [78], speech [79], images [80] and graph [81] by designing and developing efficient algorithms for feature representation learning. While motivating from these works, in this dissertation, we aim to develop an effective feature representation learning for frequent patterns; specifically for complex type i.e. sequence and graph. Using traditional single layer neural network we developed an algorithm called "Pattern2Vec" that can constructs d-dimensional feature vector representation for sequence and graph type patterns. We show effectiveness of "Pattern2Vec" in transaction (sequence,graph) and pattern level classification task.

"Pattern2Vec" leverages the neural-net based model similar to text, specifically [18]. The core idea of "Pattern2Vec" is to map a pattern of type sequence or graph into a sentence and pattern elements (items in a sequence or edges in a graph) as words. Afterwards, it leverages a language model for finding d-dimensional feature vector representation of the patterns. We experimentally show that such technique performs better than the existing ad-hoc metric embedding. This "Pattern2Vec" is also a significant module of IPIPD system, where it helps to achieve genericness of the discovery system.

The remainder of this Chapter is organized as follow. In Section 4.1, we discuss couple of recent feature representation learning in the domain of text and graph. In Section 4.2, we provide detail background on the language based models for words and paragraphs that we leverage in "Pattern2Vec". In Section4.3 and 4.4, we discuss in

detail how we compute feature representation for sequence and graph type patterns. In Section 4.5, we present empirical evaluation of "Pattern2Vec". Finally, we end the chapter with a conclusion(Section4.6).

4.1 Related Works

In recent years, researchers have taken a deep dive into the domain of effective representation learning of various kinds of data ranging from text, image, graph and speech. In the section, we will discuss few recent works on representation learning of text and graph data.

4.1.1 Representation Learning of Text

In NLP domain, effective representation learning is an active line of research. Researchers have used n-gram based representation for a long period of time. Due the sparsity issues related with n-gram based techniques and with the advancement of computational power researchers have concentrate on using traditional or deep neural-net based approach for text representation. Two of the recent state of the art method is Word2Vec [78] and Paragraph2Vector [18]. Word2Vec is a traditional neural-net based algorithm that find d-dimensional feature representation of a word in the text corpus. Word2Vec train a single layer neural network with objective function to find the probability of neighboring words given an input word. Paragraph2Vector is an extension of Word2Vec that finds d-dimensional feature representation of each text (paragraph) in the corpus.

4.1.2 Representation Learning of Graph

Graph is a complex data structure. Finding effective metric representation of graph is also an active line of research in representation learning. Majority of the work done in this direction concentrate on computing feature vector representation of

nodes or edges of the graph. Recently in [81,82], authors have shown the potential of neural-net based unsupervised feature representation of the vertices of an input graph for vertex classification [81] and link prediction [82]. [81] propose an algorithm called "DeepWalk" for finding neighborhood-based feature representation of a vertex in a graph. "DeepWalk" essentially performs multiple number of smaller length random walks originated from randomly chosen vertices of the input graph. It then map each of these random walk as a paragraph where each vertex in the walk as words in the paragraph. Then it leverage the Word2Vec model to find representation of vertices. DeepWalk shows its effectiveness in node classification domain. Mahmudur et.al. in [**?**] uses sparse auto-encoder to compute vector representation of edges of a graph and solve link predicting problem. Very recently, Grover et.al proposed an algorithm named node2vec [83] that learns continuous feature representation for nodes in a graph. They proposed a flexible notion of a vertex's neighborhood in their representation. They shows applicability of the proposed method in multi-label classification and link prediction domain. In the same time-frame, Wang et.al also proposed node representation learning algorithm for graphs [84]. This algorithm is capable to highly non-linear network structure. Authors of this work [84] shows applicability of the proposed method in the multi-label classification, link predicting and graph visualization domain.

4.2 Background

In this section, we discuss several key concepts related to the recent Neural Network-based unsupervised approach of language modeling. First, we discuss a model to find effective feature representation of words [78] in a corpus and later, the extension of the word model to compute feature representation of a paragraph [18].

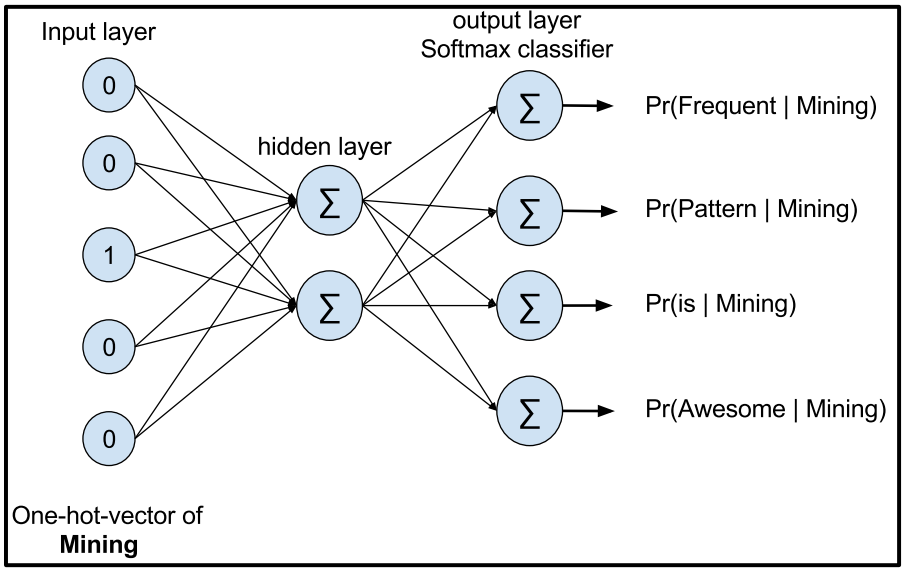


Fig. 4.1. Neural network of feature representation of words

4.2.1 Feature Representation of Words

Given a sequence of words $W = \{w_0, w_1, \cdots, w_N\}$, where $w_i \in \mathcal{V}$ ($\mathcal{V}$ is the dictionary), the objective of the word model is to maximize the average log probability:

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{-c \leq j \leq c \ j \neq 0} \log \Pr(w_{i+j}|w_i) \tag{4.1}$$

Here, c can be denoted as the training context window, describing the neighborhood words of the center word w_i. Larger size context (window) introduces more training sample for the model to use hence increases model accuracy in exchange of expensive computation. In the word model, the probability $\Pr(w_{i+j}|w_i)$ is designed by a softmax function:

$$\Pr(w_{out}|w_{in}) = \frac{exp\left({v'_{w_{out}}}^T v_{w_{in}}\right)}{\sum_{w=1}^{|\mathcal{V}|} exp\left({v'_w}^T v_{w_{in}}\right)} \tag{4.2}$$

where v_w and v'_w are the input and output feature vector representations of word w, and $|\mathcal{V}|$ is the size of the dictionary i.e. total number of words. Essentially,

the language model for word is a simple neural network with a single hidden layer. One-hot-vector representation of words are fed as input to this neural net, which has $|\mathcal{V}|$ components (one for every word in our dictionary $\mathcal{V}$) with a 1 in the position corresponding to the input word (w_i) in the dictionary and 0 in all other positions. With the task of maximizing the above probability, the output of the neural net is a single vector for size $|\mathcal{V}|$. Each index of the output vector will hold the probability that every word in the dictionary would appear in the context of the input word (w_i). In the process of learning such probabilities, the neural net produces effective feature vector representation of the word w_i in the hidden layer. Number of nodes in the hidden layers dictates the dimension of the feature vector representation of the input words.

Example: Suppose, we have a sentence `Frequent Pattern Mining is Awasome`. The size of dictionary is 5. Lets assume, word `Mining` as a center word for which we are going to compute vector representation and say size of the context window is 2. So the word model tries to compute Pr(`Pattern`|`Mining`), Pr(`is`|`Mining`), Pr(`Frequent`|`Mining`), Pr(`Awasome`|`Mining`). As discussed earlier, initially each word is represent using One-hot-vector encoding. For `Mining` the encoding looks like $[0, 0, 1, 0, 0]$. In Figure 4.1, we show the single layer neural net for this task. Note that, we use two nodes in the hidden layer. Once the model learns all the conditional probabilities mentioned in the output layer the model produces the 2-dimensional vector represent of word `Mining` in the hidden layer.

The motivation of adopting such technique to obtain effective vector representation of word is following. If two different words share similar context i.e. both of the words has similar neighboring words, the word model like the above will produce similar conditional probabilities for these two words. In order to produce close results i.e. predicting similar context, the vector representation of these two words need to be similar, that the model learns in the hidden layer. So if any two words share similar context then the network is encouraged to learn similar vector representation of these two words.

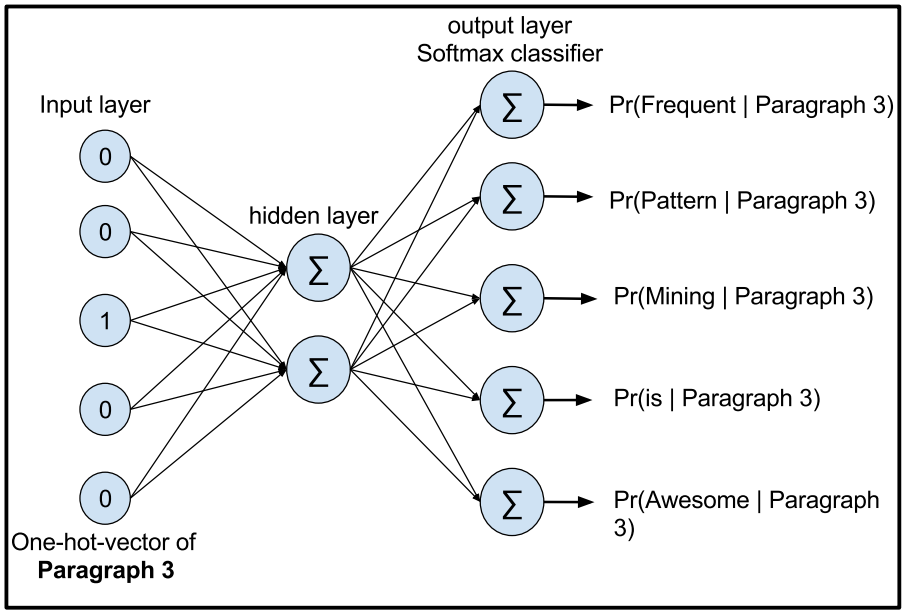


Fig. 4.2. Neural network of feature representation of paragraphs

4.2.2 Feature Representation of Paragraphs

Given a sequence of words $W = \{w_0, w_1, \cdots, w_N\}$ distributed into P paragraphs. $\mathcal{V}$ is the vocabulary. The objective of the paragraph model is to maximize the average log probability:

$$\frac{1}{N} \sum_{p=1}^{P} \sum_{j \in c} \log \Pr(w_j | p) \tag{4.3}$$

The paragraph model is an extension of the word model. Paragraph model considers each paragraph as another word in the dictionary and the objective function maximizes the conditional probability of a context c given the paragraph. Similar to the above model along with the task of maximizing the conditional probability, it outputs d-dimensional feature vector representation of each paragraph, P. The training of feature vector representation of the text is done using stochastic gradient descent and the gradient is obtained via back-propagation. Following the above example of word, suppose we have five paragraphs and size of the dictionary is five and we want

to find vector representation of 3rd paragraph. In Figure 4.2, we show the modified neural network to obtain effective vector representation of the 3rd paragraph.

4.3 Feature Representation of Sequence Pattern

In sequence dataset $\mathcal{D}$, each transaction T_i is an ordered list of events. For example, $ACCGA$ is sequence and A, C and G are events. Similar to set, One can view the dataset $\mathcal{D}$ as a text corpus and sequence of events in a transaction as a sentence in a language, where the events are the words. Given a set of sentences, our objective is to find a language model in which a sequence has a metric embedding in an appropriately chosen vector space. To obtain the language model, we apply `Paragraph Vector` [18]. It finds the d-dimensional (d is user defined) feature representation of all sequences in $\mathcal{D}$.

In Figure 5.2, we illustrate how we compute feature representation of the sequences in $\mathcal{D}$. As we can see, we treat sequences as sentences in a text corpus (step 1 in Figure 5.2). In the 2nd step, we pass the corpus in the language model (shown as a black box in Figure 5.2). As an output (step 3 in Figure 5.2), language model produces feature representation of a given length ($d-$dimension) for all transactions, i.e. sequences $(1, 2, \cdots N)$.

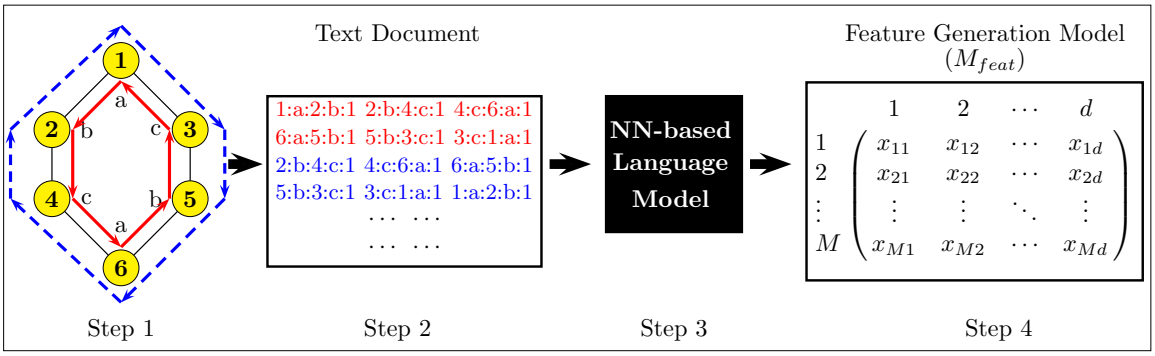


Fig. 4.3. Unsupervised feature construction of graph patterns

4.4 Feature Representation of Graph Pattern

In graph dataset $\mathcal{D}$, each transaction T_i is a labeled, undirected and connected graph. In order to use neural network based unsupervised language model, we have to map the dataset $\mathcal{D}$ to a text corpus and the graphs in $\mathcal{D}$ to sentences, on which we apply `Paragraph Vector` [18]. For converting a graph into sentence(s) we break the graph into a collection of simple paths, such that edge ordering is maintained in each path. Each path can then be viewed as a sentence and the edges in the path as words. To construct these paths from a graph T_i, for each node u in T_i, we perform a dfs-walk originated from u that traverse all the edges in T_i. The motivation of using dfs-walk is that it preserves topological information of a graph around the originated vertex; multiple dfs-walks originated from different vertices capture topological information of the entire graph. Besides, it provides an ordering of edges in the paths which serves as sentences. The number of paths that we generate from each graph is exactly equal to the number of vertices in the graph because each dfs-path is originated from a distinct vertex in the graph.

Note that, for a set or a sequence dataset, we maintain a one-to-one mapping between the transactions and the sentences, but for graph, it is a one-to-many mapping. The primary reason for this is that a complex object like graphs cannot be represented by only one sequence of vertices. More importantly, the number of transactions for graph data is usually small (Table 5.1) compared to a dataset of sets and sequences. So populating one sentence per graph is not an optimal choice. In many of the existing works on machine learning, specifically in deep learning, artificial random noises are inserted to create distorted samples [85, 86] which greatly increases the training set size. In [85], authors proposed a technique called "elastic distortion" to increase the number of image training data by applying simple distortions such as translations, rotations, and skewing. However in the existing works, no such mechanism is available for training data inflation for a graph data. The one-to-many mapping that

we use is such a mechanism which enables us to create many samples for a graph instance, thus increasing the learning ability of the model for graph patterns.

In Figure 4.3, we illustrate the entire process. In Step 1, we have a graph (transaction) with 6 nodes and 6 edges, where each of the nodes has a label. We assume all the edges share the same label 1 (not shown in the figure). Step 2 obtains six dfs-walk edge sequences, on which two are shown in the figure—one dfs-walk originated from vertex 1 (red ink) and the other originated from vertex 2 (blue ink and dashed). Note that, it is possible to have dfs-walk with different ordering from a vertex but in this work, we allow one walk per vertex. Once we have such representation of the entire data $\mathcal{D}$, we feed the text corpus to the language model (Step 3) and finally for a given feature length d, it trains the model (M_{feat}) and produces feature vectors of all M dfs-walks in the text corpus. During the classification stage for a pattern graph, we create only one dfs-walk from a randomly selected vertex and generate feature vector representation by using M_{feat}.

We give some perspective of adapting neural network based unsupervised language modeler [18] to model and find feature representation of a graph. When we map dfs-walks as sentences, estimated likelihood can be express as observing the walk d_{walk} given all the vertices $(v_0, v_1, \cdots, v_i)$ in a walk.

4.5 Experiments

In this Section, we empirically evaluate the performance of "Pattern2Vec" on several real-world dataset.

4.5.1 Data

The two sequence dataset Reuters1 and Reuters2 is collected according to the procedure mention in a sequence classification work [87]. These datasets are prepared by extracting specific keywords from the entries in the Reuters-21578 corpus [1]. Reuters1

[1] `http://web.ist.utl.pt/~acardoso/datasets/r8-train-stemmed.txt`

consists of word sequences labeled by either "earn" or "acq". Reuters2 consists of sequences labeled by "interest", "money-fx", "crude" and "trade". For more information see [87]. Out of the two graph datasets, one is called `pdb`, which is collected from predictive toxicology challenge[2]. The other graph dataset, `Mutagenicity-II` is obtained from the authors of [73]. In Table 5.1, we present some basic statistics about these four datasets; $\rho^{\min}$ is the minimum support threshold, and $\mathcal{C}$ is the set of all closed frequent patterns. For each dataset, we choose $\rho^{\min}$ in a way so that we can have a large number of closed frequent patterns for the validation of the system. In column four, we report the number of transaction class labels for each data.

Table 4.1
Dataset statistics

Type	Dataset	# of transactions	No. of class labels	$\rho^{\min}$	$\lvert\mathcal{C}\rvert$
Sequence	Reuters1	4335	2	86	54193
	Reuters2	900	4	36	48209
Graph	PDB	416	2	8	9246
	Mutagenicity-II	684	2	8	20771

Table 4.2
Sequence transaction classification

Dataset	N-gram	Neural-Net
Reuters1	0.80	0.91
Reuters2	0.74	0.82

[2] `http://www.predictive-toxicology.org/ptc/`

Table 4.3
Graph transaction classification

Dataset	Topological feature	Neural-Net
PDB	0.65	0.88
Mutagenicity-II	0.62	0.80

4.5.2 Experiment Setup

For pattern classification purposes, we need pattern mining algorithm that mines frequent pattern. Here, we use spmf [88] and CloSpan [89] for mining closed set, sequence and graph patterns, receptively. A closed frequent pattern is defined as a pattern that does not have a super-pattern (super-set) that is frequent with the same support count [1]. The set of closed pattern, $\mathcal{C}$, is a subset of the set of frequent patterns, $\mathcal{O}$. Redundancy among the patterns in $\mathcal{O}$ is substantially reduced in the patterns in $\mathcal{C}$, so learning a function over the patterns in $\mathcal{C}$ is better than learning a function over the patterns in $\mathcal{O}$.

For performance evaluation, we make 5-folds of the data of which 4 folds are used for interactive training and 1 fold for testing. We use, regularized softmax classification algorithm for both transaction and pattern level classification task. We use a grid search to tune regularization parameter λ and set it to 1. We run Gensim's [90] implementation of `Paragraph Vector` model with the default parameter settings, except for the parameter d (the dimension of feature vector), which we set it to 100 for both sequence and graph after tuning it using grid search. We report the performance result using average weighted F-score (calculate F-Score for each label, and compute their average, weighted by the number of true instances for each label) over the folds. We implement "Pattern2Vec" in python and run all experiments in 3 GHz Intel machine with 8GB memory.

The target labels of a pattern is derived from the class label of its predominating transactions. For example, if a pattern p belongs to t number of transactions among

which t_1 and t_2 are the numbers of transactions with label 1 and 2 respectively; then $t = t_1 + t_2$. The corresponding label/feedback of p is measured as 1 if $max(t_1, t_2) = t_1$ else 2.

4.5.3 Transaction Classification

In this experiment, we evaluate the effectiveness of Pattern2Vec using the task of transaction classification of type sequence and graph. Four datasets mentioned in Table 5.1 have class label associated to each transaction. For example, each transaction in Reuters1 and Reuters2 is labeled by 2 or 4 class respectively. In this experiment, first we train Pattern2Vec using the datasets and then get the d-dimension feature representation of each of the transactions in the dataset and setup a transaction classification task using softmax classifier. In Table 4.2 and 4.3, we show classification performance for sequence and graph data in-terms of weighted f-score. For all datasets, we have found dimension value 100 provide better transaction classification performance.

4.5.4 Pattern Classification

In this experiment, we evaluate the effectiveness of the proposed unsupervised feature construction based approach with alternative feature representation for sequence and graph patterns. For sequence pattern, we use n-gram based technique, where we extract all 2-grams and 3-grams from the data and use those as features. For graph patterns, we obtain 20 topological metrics compiled by [17] and consider these metrics as features for graph patterns. After mining the closed pattern from the datasets mentioned in Table 5.1, we apply Pattern2Vec to extract d-dimensional feature vectors of all patterns using the model trained on the entire corresponding dataset. Then we follow five fold cross-validation protocol to test the effectiveness of Pattern2Vec through a softmax classification algorithm. For all datasets, we have found dimension value 100 provide better pattern classification performance. In fig-

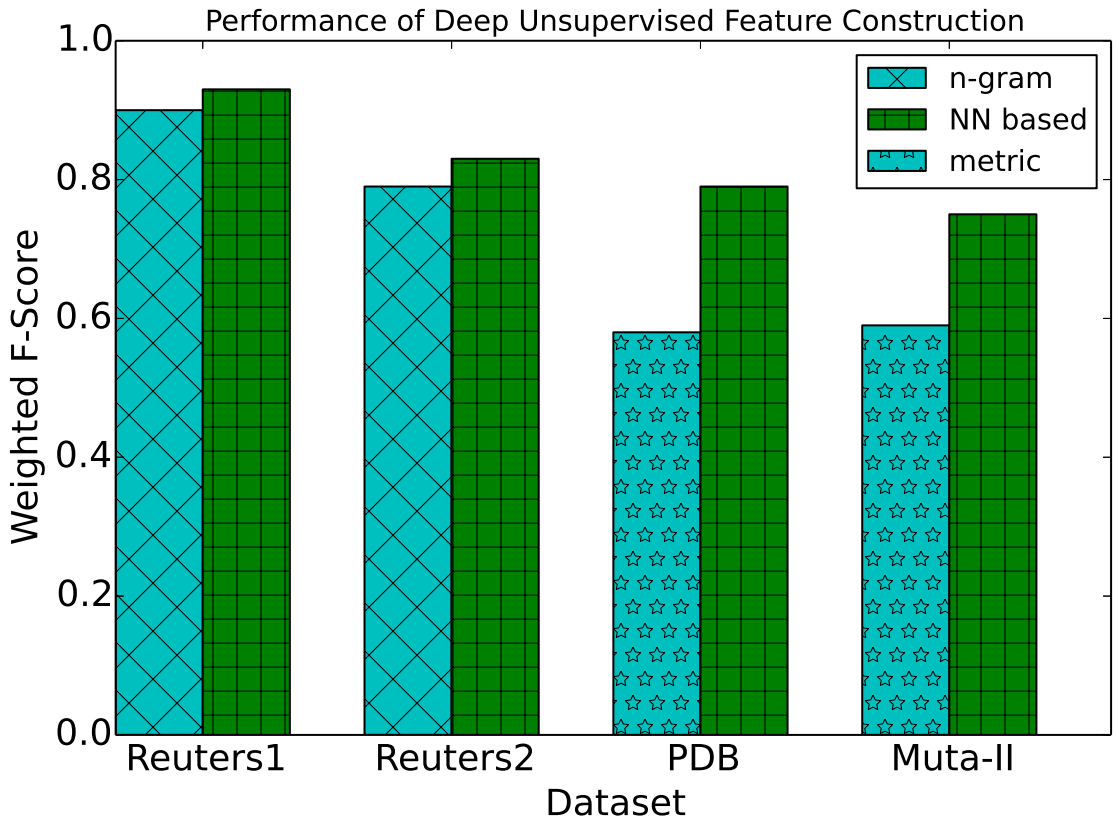


Fig. 4.4. Performance of unsupervised feature construction.

ure 4.4, we show the findings using bar chart. First two groups of the bar chart are for sequence datasets and the remaining two groups are for graph datasets. As we can see, Pattern2Vec performs better than the alternative feature representation of both sequence and graph patterns. Specifically, Feature embedding for graph datasets is significantly better than that of the competitors'.

4.6 Conclusion

In this work, we design and develop a representation learning algorithm for frequent patterns called "Pattern2Vec". "Pattern2Vec" is based on traditional single layer neural network used in NLP domain for language modeling. We show effec-

tiveness of "Pattern2Vec" in transaction and pattern level classification tasks. This proposed representation learning algorithm is also an integral part of IPIPD system proposed throughout this dissertation.

5. PRIIME: A ROBUST FRAMEWORK FOR INTERACTIVE PERSONALIZED INTERESTING PATTERN DISCOVERY

5.1 Introduction

In this work, we propose a generic framework for interactive personalized interesting pattern discovery (IPIPD) called PRIIME[1] by incorporating a user in the discovery pipeline. The proposed method uses preference rating of a small set of frequent patterns from the user to learn her interestingness criteria and recommend a set of patterns which incline the most towards her choice. To ensure genericness of PRIIME, we develop a neural net based unsupervised feature vector representation scheme for sequence and graph patterns. We propose a pattern selection approach for feedback to combat the *positive reinforcement* phenomenon. We also assume that feedback is given using positive real discrete values, where a higher value stands for more empathy towards the given pattern. For example, a user can have a three-class rating system, where 1, 2 and 3 means dislike, not sure and likes the pattern.

PRIIME also provides a robust pattern selection strategy for rating solicitation. Unlike earlier interactive pattern discovery works [9, 14], for the case of PRIIME, the coupling between the existing state of the learning model and the training set selection is significantly reduced which yields a much-improved learning model. In fact, PRIIME's strategy is a combination of exploration and exploitation which enables it to reduce the bias that existing state of the learning model imposes on pattern selection. Exploitation aspect of the proposed strategy prefers patterns (for feedback solicitation) that are likely to influence the learning model the most, i.e. have

[1]PRIIME is an anagram of the bold letters in gene**R**ic fra**M**work for **I**nteractive p**E**rsonalized **I**nteresting **P**attern discovery

the largest impact on model's parameters. On the other hand, the exploration aspect ensures that a broader span of the pattern space is covered. PRIIME maximizes expected gradient length (EGL) [91] (an active learning strategy) for exploitation and it chooses a diverse set of patterns using k-center [92] search for exploration. We empirically show that a combination of above two triggers better learning performance than the existing interactive pattern discovery frameworks.

We claim the following contributions:

• We propose a generic interactive personalized interesting pattern discovery framework called PRIIME that is based on iterative learning of a user's interestingness function using the supervised classification and regression models based on the nature of feedback (discrete and real number) design. The system does not require any prior knowledge of the user's interestingness criteria and needs a limited degree of user engagement.

• PRIIME leverages efficient unsupervised feature construction scheme for sequence and graph patterns named "Pattern2Vec" discussed in the Chapter 4 and propose a pattern selection strategy combining exploitation and exploitation for feedback collection.

• We perform exhaustive empirical validations of the proposed framework using real-world set, sequence, and graph datasets. We show that the proposed feature construction and pattern selection performs better than the existing ones.

The reminder of this chapter is organized as follows. In Section 5.2, we discuss in detail of the proposed framework. In Section 5.3, we talk about the learning algorithms of the framework including pattern representation (Subsection 5.3.1), classification model (Subsection 5.4.1) and feedback mechanism (Subsection 5.4.3). In Section 5.5, we present an extensive empirical evaluation including the performance of the learning algorithms (Subsection 5.5.7), neural net based unsupervised feature construction (Subsection 4.5.4) and comparisons with the existing methods (Subsection 5.5.5). We conclude the chapter a conclusion (Section 6.8).

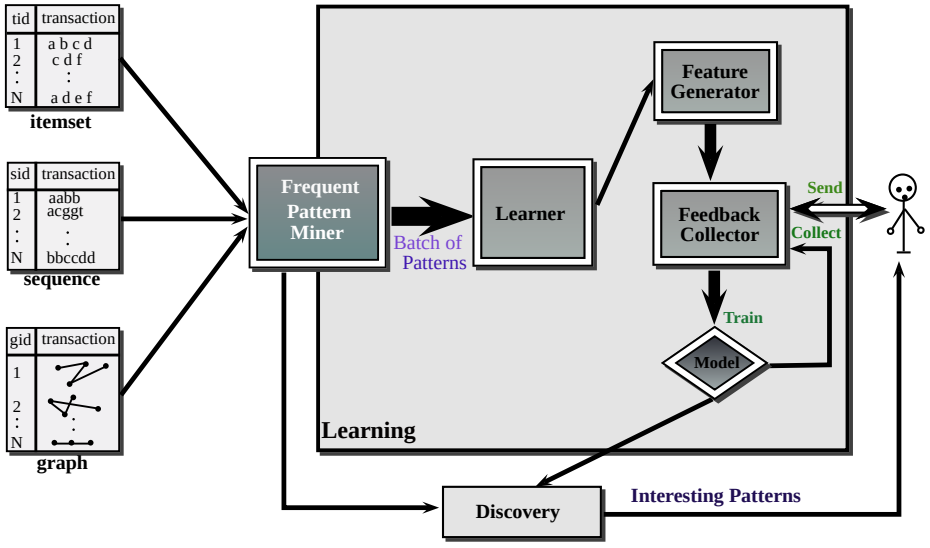


Fig. 5.1. Generic interactive personalized interesting pattern discovery framework

5.2 Problem Definition and System Architecture

We assume, u possesses an interestingness function (say, f) over the frequent patterns, $\mathcal{O}$). f maps each pattern in $\mathcal{O}$ to a score (discrete/non-negative real numer) indicating interestingness criteria of the user u i.e., $f : \mathcal{O} \rightarrow \mathbb{Y}$. Note that, $Y \in \mathbb{R}_+$ as well as $Y \in \mathbb{C}$ is true, where $\mathbb{R}_+$ is the set of non-negative real number and C is the set of class labels. PRIIME learns f through an active learning process over multiple learning iterations. During an iteration (say i), the system returns a set of patterns $\{p_t\}_{1 \leq t \leq l}$ to u, which is selected from a partition of $\mathcal{O}$ (which constitutes the i'th batch of pattern-set); u sends feedback $\{y_t = f(p_t)\}_{1 \leq t \leq l}$ using an interactive console; y_t's are interestingness score reflecting the user's empathy towards the selected patterns; using the feedback the system updates its current model of the user's interestingness function, f. An alternative to our batch learning framework can be to mine all the frequent patterns and select a relatively smaller subset of patterns for getting a user's feedback and use these for learning in one iteration. This approach is not scalable as the number of frequent patterns in $\mathcal{O}$ is typically very large even for a moderate-size dataset. Besides, if the learning entity receives the frequent patterns as a data

stream the one-iteration learning becomes infeasible, but a batch learning framework still works.

The IPIPD framework of PRIIME is partitioned into two main blocks: *Learning* and *Discovery* (Figure 5.1). *Learning* learns a model for a user; it contains following five modules: Frequent Pattern Miner (FPM), Learner, Feature Generator (FG) and Feedback-Collector (FC), and Model. The FPM works as a bridge between the Learner and the data. FPM module can be any off-the-shelf pattern mining algorithm that mines frequent patterns given a normalized minimum support threshold. Any of the existing off the shelf pattern mining algorithms ranging from sequential [93] to randomized [52], can be used. The Learner learns u's interestingness function (f) over the pattern-set $\mathcal{O}$ in multiple stages while utilizing user's feedback on a small number of patterns selected from a partition of $\mathcal{O}$ in each stage. The Feature Generator (FG) module is responsible for generating efficient and appropriate feature vector representation of the incoming patterns for the Feedback Controller and the Model to use. FG uses the bag of words method for set patterns and NN-based unsupervised language model for sequence and graph patterns. The Feedback-Collector (FC)'s responsibility is to identify the patterns using a exploitation-exploitation based strategy (explained in Section 5.4.3), that are sent to the user for feedback. Note that, patterns are sent to the user for feedback in its original form, not in the feature vector representation. *Discovery* delivers personalized interesting patterns to the user using the learned model.

We like to highlight some important facts about the learning task of the proposed framework. First, in the learning phase, the interaction between the PM and the Learner is one-way from PM to the Learner, and the Learner has no influence over PM's mining process. This setup works well for large datasets for which the mining task may take a long time yet the learning task can start as soon as the first batch of patterns is available. Moreover, if a user wants to end the learning session before PM finishes, the learner can still provide interesting patterns to the user based on the current learning model.

Algorithm 5: PRIIME_Learning

Input: $\mathcal{D}$:transaction dataset

```
M_feat = Train feature generation model(D);
Initialize model parameters w;
while true do
    B = Incoming batch of patterns from FPM;
    P_feat = Get feature representation(B,M_feat);
    P = Collect feedback(P_feat);
    w' = Train learning model(P,w);
    if |w'-w| < threshold then
        break;
    end
    w = w';
end
return w;
```

5.3 Learning Method

In PRIIME , the learning of interestingness function f and pattern mining by FPM proceeds in parallel over a few iterations. In the i'th iteration, PM generates a partition ($\mathcal{B}_i$) of frequent pattern set $\mathcal{O}$, and Learner selects l patterns $\{p_t\}_{1\leq t\leq l}$ from $\mathcal{B}_i$ and send them to u for rating. Meanwhile, FG obtains $\{\mathbf{x}_t\}_{1\leq t\leq l}$, the feature representation of the patterns $\{p_t\}_{1\leq t\leq l}$. Partitioning of frequent pattern enables model learning to proceed even if all the frequent patterns are not available, which is better than earlier works [9] where model learning is performed after the entire $\mathcal{O}$ is available. Learner collects u's rating $\{y_t = f(p_t)\}_{1\leq t\leq l}$ using an interactive console. Then the Learner updates its current model f using $\langle \mathbf{x}_t, y_t \rangle$.

In algorithm 5, we present the pseudo code of PRIIME's learning phase. PRIIME starts by training the unsupervised feature generation model (Pattern2Vec)

M_{feat} using the transaction data $\mathcal{D}$ and initializes the learning model $\mathbf{w}$. Afterward in each iteration, it uses M_{feat} to infer the feature representation of each pattern in the incoming batch $\mathcal{B}$. Then the algorithm selects patterns for feedback and sends them to the user. Once it receives the feedback, it updates the current learning model. The algorithm continues until the improvement of the learning model falls below a user-defined threshold. In the following, we discuss each of the tasks.

5.3.1 Pattern Representation

Say $\mathcal{D} = (T_1, T_2, \cdots, T_N)$ is a transaction dataset. $\mathcal{I}$ is the set of all single length frequent patterns. $\mathcal{I}$ is a set of item, a set of events, or a set of edges depending on whether $\mathcal{D}$ is an itemset, a sequence or a graph dataset, respectively. For a given support threshold, $\mathcal{O}$ is the set of all frequent patterns. For using a machine learning method for finding interesting patterns, PRIIME first finds a feature vector representation of these patterns. PRIIME leverages neural net-based effective feature construction algorithm called "Pattern2Vec" discussed in Chapter 4. Note that, training of such language model is computationally expensive and perform training in each iteration of PRIIME with the incoming batch of patterns is inefficient. As an alternative, PRIIME trains the model say, M_{feat} using the dataset $\mathcal{D}$ as training data (Line 1 in Figure 1). Afterward in each iteration, it populates d-dimensional feature vector of incoming patterns by using M_{feat}. For set type of pattern, PRIIME does not use "Pattern2Vec", since empirically we found Bag-of-words based model performs better for set type patterns. In the following, we discuss representation technique for set type patterns.

5.4 Feature Representation of Set Patterns

For a set data, each transaction T_i is a collection of non-repetitive items. Considering the dataset $\mathcal{D}$ as a text corpus and each transaction in the corpus as a document, the set $\mathcal{I}$ can be represented as the dictionary of frequent words i.e. items. Following

the bag of words model, we represent each pattern $p \in \mathcal{O}$ as a binary vector V of size $|\mathcal{I}|$, constructed as below:

$$V(i) = \begin{cases} 1, & \text{if item } i \in p \\ 0, & \text{otherwise} \end{cases}$$

Such a representation maps each pattern to a data point $\mathbf{x} \in \{0,1\}^{|\mathcal{I}|}$ space. For example, consider a dataset in which A, B, C, and E are frequent items. Thus $\mathcal{I} = \{A, B, C, E\}$; A frequent pattern AB will be represented by the vector $(1, 1, 0, 0)$; a pattern ACE will be represented by the vector $(1, 0, 1, 1)$ and so on.

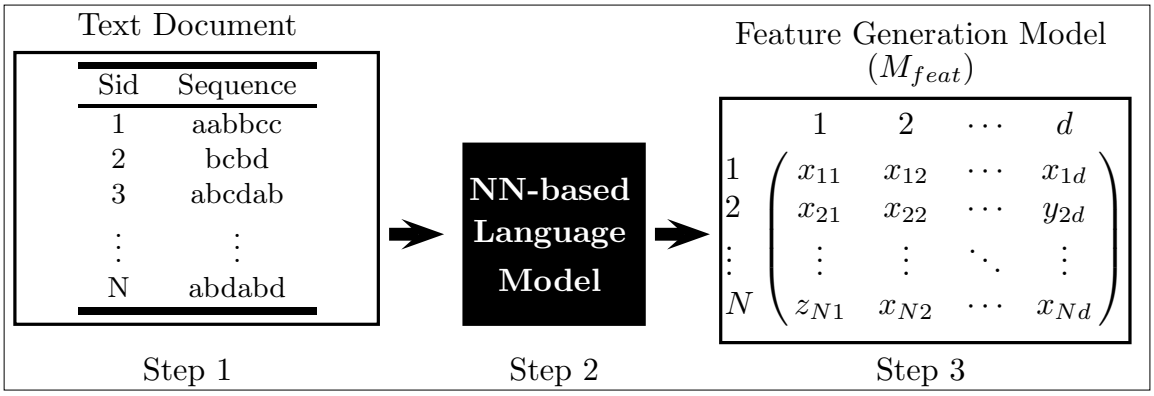


Fig. 5.2. Unsupervised feature construction of sequence Patterns

5.4.1 Classification Model

The PRIIME models the user's interestingness profile over the patterns as a classification problem and learns the model parameters through an interactive feedback mechanism. In each iteration, PRIIME executes a supervised training session of the classification model using the corresponding data points of the released patterns to the user. We use multinomial logistic regression as the classification model $H_{\boldsymbol{\theta}}$, which

is known as softmax regression in the literature. In multinomial logistic regression, the probability that a data point $\mathbf{x}_i \in \mathbb{R}^d$ belongs to class j can be written as,

$$H_{\boldsymbol{\theta}}(\mathbf{x}_i) = p(y_i = j|\mathbf{x}_i; \boldsymbol{\theta}) = \frac{exp({\boldsymbol{\theta}_j}^T \mathbf{x}_i)}{\sum_{l=1}^{c} exp({\boldsymbol{\theta}_l}^T \mathbf{x}_i)} \tag{5.1}$$

where, $j \in \{1, \cdots, c\}$ is the set of class labels and $\boldsymbol{\theta}_j$ is the weight vector corresponding to class j. The prediction of the model can be computed as, $\hat{y}_i = \text{argmax}_j \; p(y_i = j|\mathbf{x}_i, \boldsymbol{\theta})$. Given a set of labeled training data $X = \{(\mathbf{x}_1, y_1), .., (\mathbf{x}_m, y_m)\}$, the weight matrix $\boldsymbol{\theta} \in c \times d$ (c is the number of class and d is the size of a feature vector) is computed by solving the convex optimization problem as shown in Equation 6.2 using gradient descent. Note that, each data points $\mathbf{x}_i \in \mathbb{R}^d$ in the training set corresponds to a pattern.

$$\underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta}) \;=\; -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{c} \mathbf{1}_{\{y_i=j\}} \cdot \log \;\; p(y_i \;=\; j|\mathbf{x}_i; \boldsymbol{\theta}) + \frac{\lambda}{2} \sum_{j=1}^{c} \sum_{k=1}^{d} \theta_{jk} \tag{5.2}$$

Here $J(\boldsymbol{\theta})$ is the notation to represent the cost function in Equation 6.2, $\mathbf{1}_{\{y_i=j\}}$ is the indicator function indicating that only the output of the classifier corresponding to the correct class label is included in the cost. $\frac{\lambda}{2} \sum_{j=1}^{c} \sum_{k=1}^{d} \theta_{jk}$ is the regularization term to avoid over-fitting. In each iteration, PRIIME applies L-BFGS-B optimization function implemented in python's scipy package to train the model.

5.4.2 Regression Model

The proposed algorithm models the user's interestingness profile over the patterns set as a regression problem and trains the model through an interactive feedback mechanism. In each iteration, the algorithm executes a supervised training session of the regression model using the corresponding data points of the released patterns to the user. The algorithm uses the non-negative feedback provided by the user as the target function value for regression task. We use gradient boosted regression tree (an ensemble of high bias regression trees) as the regression model. In gradient boosted

tree regression, for a data point $\mathbf{x}_j \in \{0,1\}^{|\mathcal{I}|}$, the predicted score $f(\mathbf{x}_j) \in \mathbb{R}_+$ is computed as,

$$f_m(\mathbf{x}_j) = f_{m-1}(\mathbf{x}_j) \ + \ \gamma_m h_m(\mathbf{x}_j) \tag{5.3}$$

Here, m is the stage/iteration index used by the regression algorithm. h_m is the added single regression tree in the stage m that minimizes the differentiable loss function $L(y_j, f_{m-1}(\mathbf{x}_j) - h_m(\mathbf{x}_j))$. γ_m, also known as the step length, is the weight associated with the regression tree h_m chosen using line search that minimizes the loss function. To avoid overfitting of the model, a learning rate α is incorporated in the boosting update rule as a regularization parameter:

$$f_m(\mathbf{x}_j) = f_{m-1}(\mathbf{x}_j) \ + \ \alpha\gamma_m h_m(\mathbf{x}_j) \tag{5.4}$$

In this work, we use the trained regression model of iteration $i-1$ to initialize the boosted regression model of iteration i.

5.4.3 Selection of Representative Data-points for Feedback

As discussed in Section 5.1, one of the reasons of the superior performance of our proposed method over the existing ones is due to the exploitation-exploration based feedback collection. In each iteration, for an incoming batch, PRIIME exploits the currently learned model (M_{cur}) to identify the patterns that will have the largest impact on the parameters of the learning model. We can identify those patterns by checking whether these patterns make the maximum change in the objective function. Since we train softmax classifier using gradient descent, we include the patterns to the training set if it creates the greatest change in the gradient of the objective function,

$$\bigtriangledown_{\theta_j} J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[\mathbf{x}_i\left(\mathbf{1}_{\{y_i=j\}} - p(y_i = j|\mathbf{x}_i;\boldsymbol{\theta})\right)\right] + \lambda\theta_j \tag{5.5}$$

However, gradient change computation requires the knowledge of the label, which we don't have. So, we compute the expected gradient length [91] of $\mathbf{x}_i$ as shown in Equation 6.4,

$$EGL(\mathbf{x}_i) = \sum_{j=1}^{c} p(y_i = j|\mathbf{x}_i;\boldsymbol{\theta})||\bigtriangledown_{\theta_j} J(\boldsymbol{\theta})|| \tag{5.6}$$

After exploitation step, PRIIME retains the top 50% of patterns ($\mathbf{x}_i s$) according to the expected gradient length (EGL) as candidates for feedback. However, Expected gradient length based approach enables PRIIME to identify patterns that can make greatest changes in the learning model, but it does not ensure diversity among the patterns in terms of its composition, i.e. two patterns can be made of similar items, events or edges. To ensures that a broader span of the pattern space is covered, PRIIME uses k-center search (exploration) to identify k patterns for feedback from the candidates set.

k-Center. k-center method finds k data points as centers so that the maximum distance from each data point to its nearest center is minimized. This k-center problem is NP-complete, but there exists a fast 2-approximation greedy algorithm that solves this problem in $O(kn)$ time [92]. We set k to 10 in all our experiments. For choosing 10 representative patterns, we use the greedy 10-center algorithm considering the Jaccard distance for set patterns and euclidean distance for sequence and graph patterns.

5.4.4 Stopping Criteria

The stopping criterion is an important element of our proposed system. The execution of PRIIME can be halted from the system side as well as the user side. PRIIME sets a minimum default iteration counts to 10 and after 10th iteration, it keeps track whether getting additional feedback impact significantly on the learning model or not. If the improvement falls below to a threshold ($1E-4$), the execution of the feedback session halts.

5.5 Experiments and Results

5.5.1 Data

We use five set datasets, two sequence datasets, and two graph datasets to validate the performance of PRIIME. Two of the set datasets, `Chess` and `Pumsb` and `Mushroom`, are real-life datasets collected from FIMI repository[2]. The third one is an artificially generated electronic health record (EHR) dataset created by Uri Kartoun from Harvard University[3], which has health record of $10,000$ patients. To generate a class label for each patient, we identify whether the patient is diagnosed with arthritis, type-I/II diabetics or not [4]. The last set dataset is a drug side effects dataset collected from CIDER repository[5]. This data contains 996 drugs and associated side effects. We consider each drug as a transaction and side effects as the items. Each drug also has a class label based on the primary disease which it cures: cancer, heart diseases, brain diseases, pain, and others.

The two sequence dataset Reuters1 and Reuters2 is collected according to the procedure mention in a sequence classification work [87]. These datasets are prepared by extracting specific keywords from the entries in the Reuters-21578 corpus [6]. Reuters1 consists of word sequences labeled by either "earn" or "acq". Reuters2 consists of sequences labeled by "interest", "money-fx", "crude" and "trade". For more information see [87]. Out of the two graph datasets, one is called `pdb`, which is collected from predictive toxicology challenge[7]. The other graph dataset, `Mutagenicity-II` is obtained from the authors of [73]. In Table 5.1, we present some basic statistics about these four datasets; $\rho^{\min}$ is the minimum support threshold, and $\mathcal{C}$ is the set

[2] `http://fimi.ua.ac.be/data/`

[3] `http://bit.ly/1Db1yHo`

[4] Generally class label is not needed for pattern mining. We use the class labels to simulate user's interestingness over the patterns.

[5] `http://sideeffects.embl.de/download/`

[6] `http://web.ist.utl.pt/~acardoso/datasets/r8-train-stemmed.txt`

[7] `http://www.predictive-toxicology.org/ptc/`

of all closed frequent patterns. For each dataset, we choose $\rho^{\min}$ in a way so that we can have a large number of closed frequent patterns for the validation of the system.

Table 5.1
Dataset statistics

Type	Dataset	# of transactions	Average transaction size	$\rho^{\min}$	$\lvert\mathcal{C}\rvert$
Set	Chess	3196	37	2100	99262
	Pumsb	49046	73	27000	91453
	EHR	10000	43	3000	104297
	Drug-Side-effects	996	99	220	117544
	Mushroom	8124	22	1100	53628
Sequence	Reuters1	4335	56	86	54193
	Reuters2	900	108	36	48209
Graph	PDB	416	15	8	9246
	Mutagenicity-II	684	24	8	20771

5.5.2 Experimental Setup

An essential part of PRIIME is a pattern mining algorithm that mines frequent patterns and send them to the Learner in batches. We use LCM [94], spmf [88] and CloSpan [89] for mining closed set, sequence and graph patterns, receptively. A closed frequent pattern is defined as a pattern that does not have a super-pattern (super-set) that is frequent with the same support count [1]. The set of closed pattern, $\mathcal{C}$, is a subset of the set of frequent patterns, $\mathcal{O}$. Redundancy among the patterns in $\mathcal{O}$ is substantially reduced in the patterns in $\mathcal{C}$, so learning a function over the patterns in $\mathcal{C}$ is better than learning a function over the patterns in $\mathcal{O}$.

For performance evaluation, we make 5-folds of the data of which 4 folds are used for interactive training and 1 fold for testing. We use a grid search to tune regulariza-

tion parameter λ and set it to 1. We run Gensim's [90] implementation of `Paragraph Vector` model with the default parameter settings, except for the parameter d (the dimension of feature vector), which we set it to 100 for both sequence and graph after tuning it using grid search. We report the performance result using average weighted F-score (calculate F-Score for each label, and compute their average, weighted by the number of true instances for each label) over the folds. In each iteration, we approximately use 2% of the training data to construct the batch and select 10 patterns from the batch for feedback. Note that in the first iteration, PRIIME uses the k-center search to find 10 patterns for feedback. We implement PRIIME in python and run all experiments in 3 GHz Intel machine with 8GB memory.

5.5.3 Interestingness Criteria

Following earlier works [9, 14], for validating the proposed system we simulate the interestingness function of a user. In the following, we discuss two categories of interestingness function i.e. nature of feedback, discrete and non-negative real number.

Discrete

The interestingness criteria of a pattern is derived from the class label of its predominating transactions. For example, if a pattern p belongs to t number of transactions among which t_1 and t_2 are the numbers of transactions with label 1 and 2 respectively; then $t = t_1 + t_2$. The corresponding label/feedback of p is measured as 1 if $max(t_1, t_2) = t_1$ else 2.

Non-negative Real Number

The first interestingness score is the normalized representative class score. For example, if the pattern p belongs to t number of transactions among which t_1, and t_2

are the number of transactions with label 0 and 1 respectively; then $t = t_1 + t_2$. The normalized class score is computed as $\max(t_1, t_2)/t$. Odds ratio and Jaccard index are used as interestingness criteria of a pattern to reflect the fitness of the pattern as a feature for predicting a class. If p is a pattern and c is a class label, the Odds ratio is defined as,

$$Oddsratio(p \rightarrow c) = \frac{nsup(p, c) * nsup(\neg p, \neg c)}{nup(p, \neg c) * nsup(\neg p, c)} \tag{5.7}$$

The Jaccard index can be computed as,

$$Jacc(p \rightarrow c) = \frac{nsup(p, c)}{nsup(p) + nsup(c) - nsup(p, c)} \tag{5.8}$$

Here, $\neg p$ is the absence of the pattern p, $nsup(p)$ is the normalized support of p and $nsup(p, c)$ is the support of p within the transactions of class c.

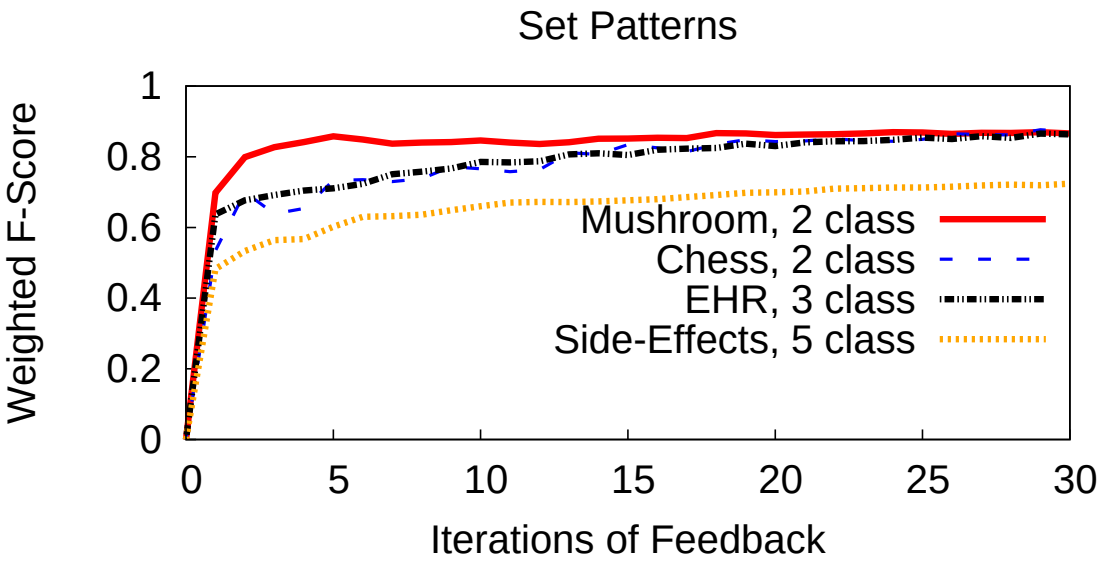


Fig. 5.3. Weighted F-Score of the learner across iterations of feedback in set dataset

5.5.4 Experiment on the Learner's Performance

In this experiment, we evaluate the performance of softmax based learning algorithm using weighted F-Score metric. we show how the performance of the learning

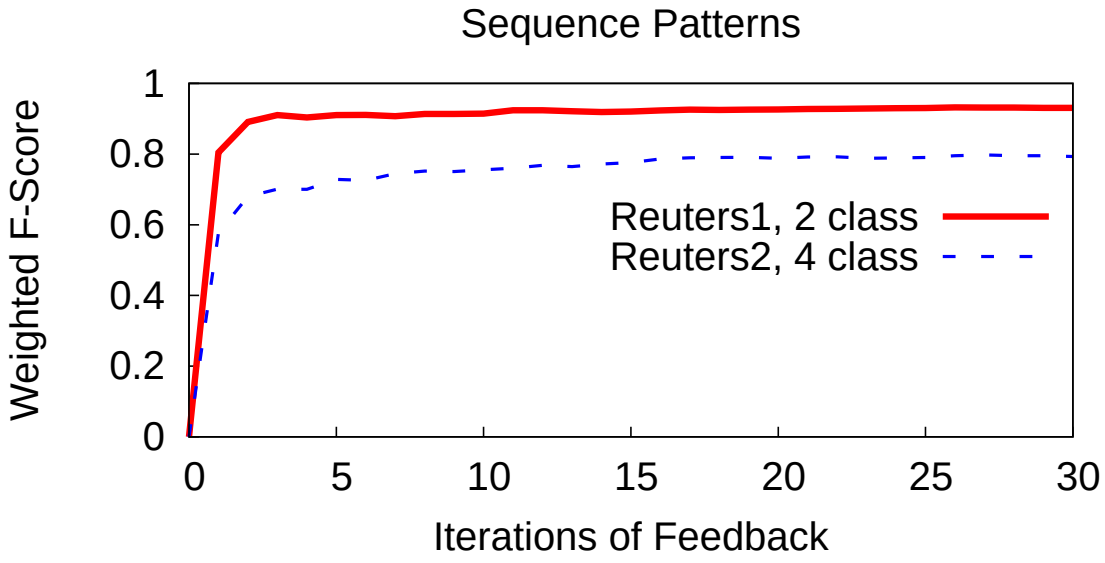


Fig. 5.4. Weighted F-Score of the learner across iterations of feedback in sequence dataset

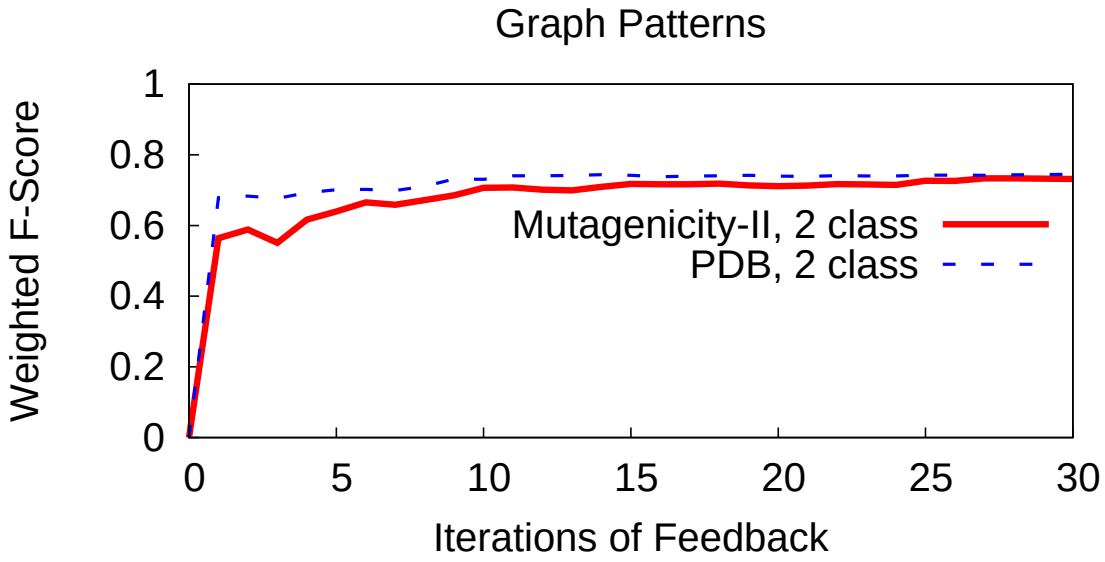


Fig. 5.5. Weighted F-Score of the learner across iterations of feedback in graph dataset

model improves with the iterations of feedback. As discussed earlier in each iteration, the learner selects ten patterns for feedback and iteratively updates the current

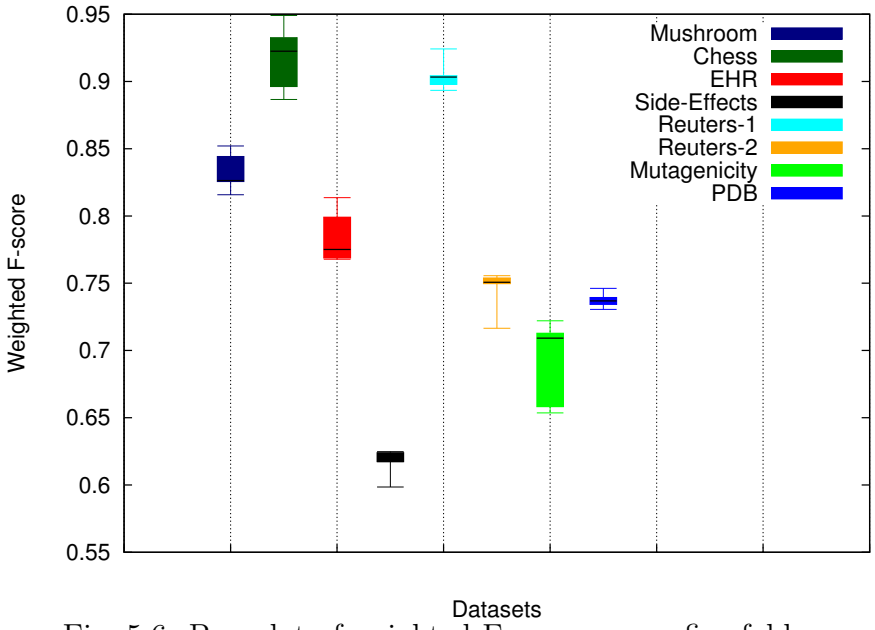


Fig. 5.6. Box-plot of weighted F-score across five folds.

model. After each iteration, we measure weighted F-Score of our currently learned model with the test fold. Then we plot F-score of the learner across the number of iterations executed so far. In Figure 5.3, 5.4 and 5.5, we show the progression of F-score across iterations for the set, sequence and graph datasets. The performance on set and sequence datasets is relatively better than that on graph datasets. The reason behind this is that effective feature vector construction for graph patterns is comparatively difficult than set and sequence patterns. For all the datasets, within five to ten iterations of feedback (each iteration has 10 patterns), learning method converges and stay stable. Compared to the entire (closed) frequent pattern space, the number of feedback that PRIIME use is almost negligible, yet the performance of the learning model is satisfactory.

In Figure 5.6, we show the box-plots of weighted F-score for each folds of training for all set, sequence and graph datasets. As we can see, except Chess and Mutagenicity data, variance between model predictive performance is comparatively low. In the plot, we report learner's performance after 10 sessions of feedback.

Table 5.2
Comparison on percentage accuracy of our algorithm with the existing ones

Dataset	Accuracy PRIIME's	Accuracy ([9]'s)	Accuracy [14]'s
Chess	90.8 %	42.7%	55.5%
Mushroom	84.5%	45.3%	61.4%
EHR	78.8%	46.6%	40.6%
Drug-Side-Effects	71.9%	36.3%	54.50%
Reuters1	91.1%	35.9%	NA
Reuters2	76.6%	38.2%	NA
PDB	73.1%	NA	50.7%
Mutagenicity-II	66.8%	NA	44.7%

5.5.5 Comparison with the Existing Algorithms

To compare with the existing works we implement [9]'s interactive learning method in python and compute the percentage accuracy experiment of their method for the set and sequence data by following the instructions provided in the chapter [9]. We also compare with [14]'s sampling algorithm (executable collected from the authors) for the set and graph data; we pick sampler's recall metric mentioned in [14] as it computes the percentage of interesting patterns recovered by the sampler. For all algorithms, we use 100 feedback and use the percentage of accuracy as a comparative metric. Table 5.2 shows the comparison, which clearly demonstrates that our algorithm performs substantially better than the existing methods. We validate that this performance boost can be credited to efficient feature construction for complex patterns and intelligent exploitation-exploration strategy for feedback pattern selection. We will discuss more on this in the following sections.

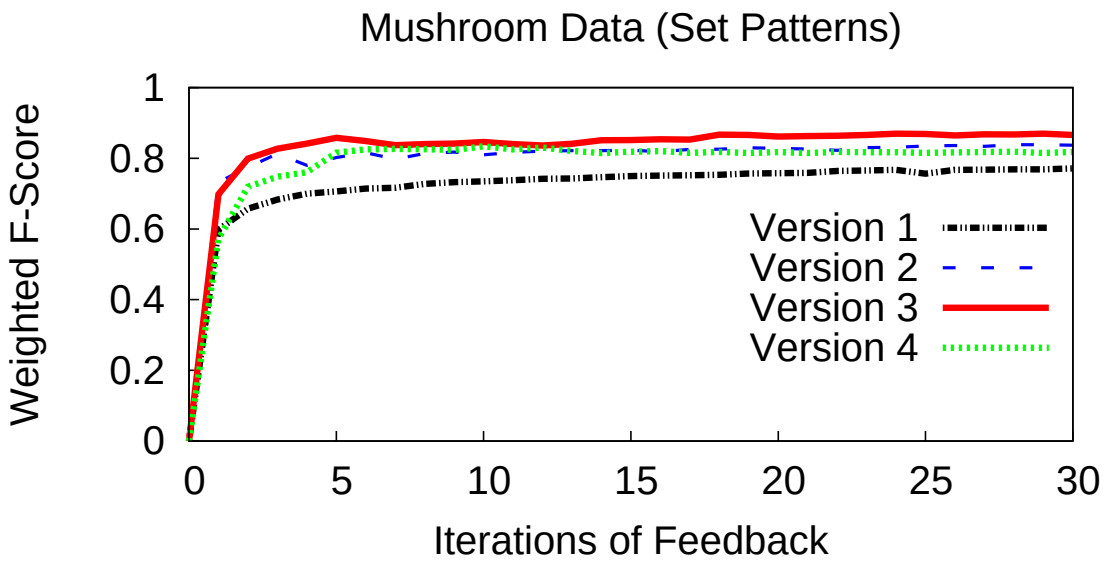


Fig. 5.7. Performance of the learner with different feedback collection scheme in Mushroom set dataset

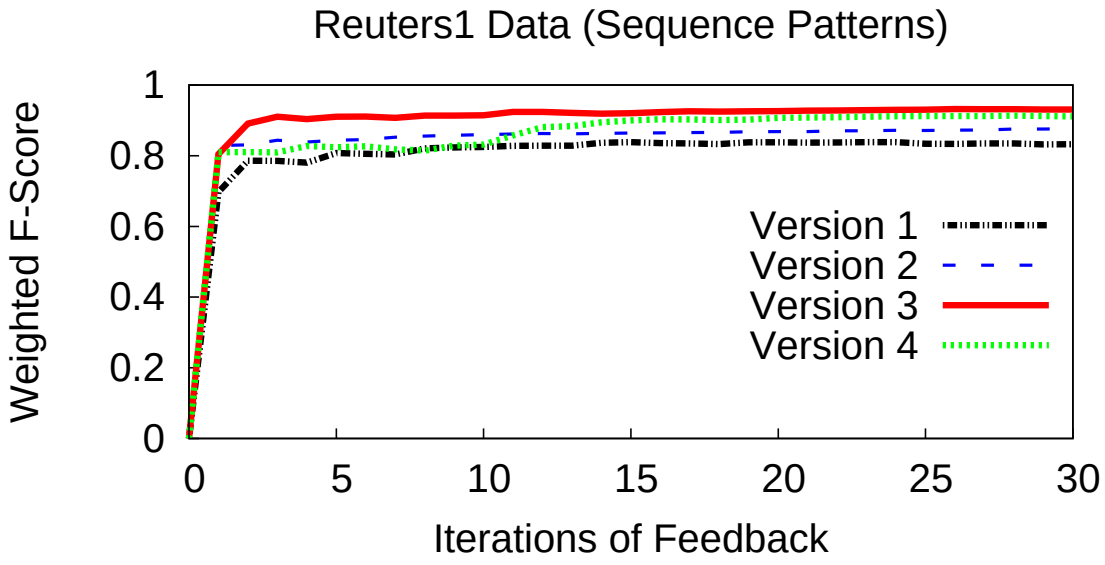


Fig. 5.8. Performance of the learner with different feedback collection scheme in Reuters1 sequence dataset

5.5.6 Representative Patterns Selection

As discussed in Section 5.1, our proposed solution is different than the existing works on how we select patterns for feedback. To show that our approach is more

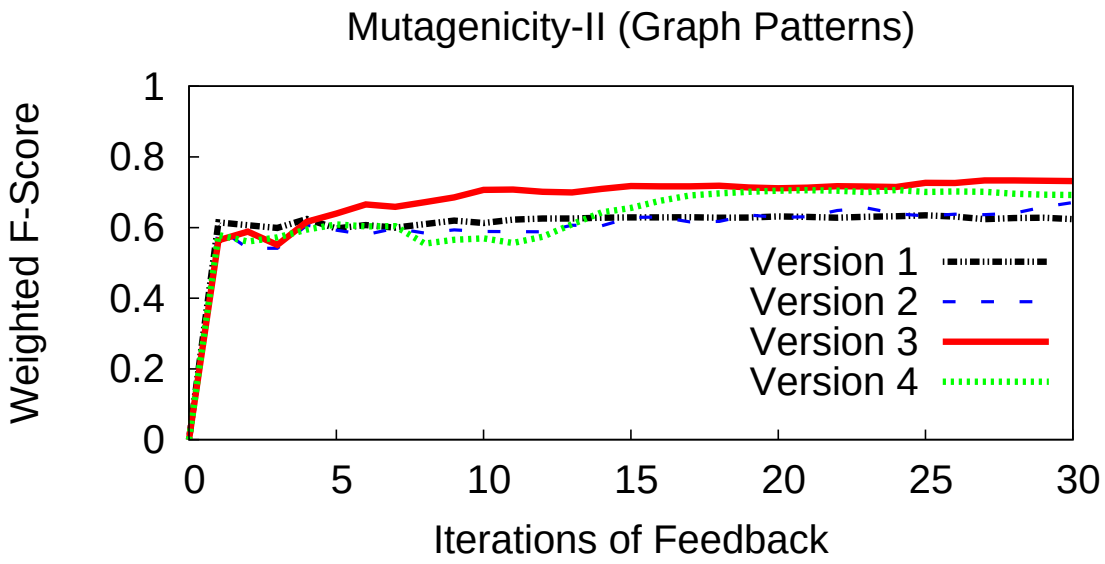


Fig. 5.9. Performance of the learner with different feedback collection scheme in Mutagenicity-II graph dataset

effective, we create four different versions of PRIIME . In one, we solely rely on exploitation i.e pick top 10 patterns for feedback from a batch according to the expected gradient length (EGL). In second, we concentrate on exploration and select 10 most diverse patterns for feedback using only k-center from the entire batch. The third version is the one that we proposed in this work, where we retain top 50% patterns from a batch according to the EGL then apply k-center to find 10 most diverse patterns for feedback. In fourth, we use only k-center (exploration) in the first 10 interactive sessions, then we use the combination of EGL and k-center (exploitation-exploration) like third version for the next 10 iterations, and in final 10 iterations, we only use EGL (exploitation). In Figure 5.7, 5.8 and 5.9, we show the performance of the above four versions of PRIIME for a set, sequence, and graph data. As we can see, for all the cases, our proposed exploitation-exploration (Version 3 curve in the figures) strategy performs better than the other three options. Our's performance better than the fourth version (Version 4 in the figures) establish the fact that in interactive pattern discovery, balancing exploration and exploitation in all iterations

of learning is beneficial than gradually changing from exploration to exploitation like the fourth version. More importantly, the performance of the learning methods that solely depends on the current learning (Version 1 in the Figures) is the lowest among all three, which suggests that high exploitation of the current model for selecting patterns for feedback may yield sub-optimal model.

5.5.7 Experimental Results of Gradient Boosted Regression Tree Model

In this section, we will explain empirical evaluations of GBRT based learning model for first four itemset dataset from Table 5.1.

Experiment Setup

The experimental setup for this version is similar to the classification model. We make 5-fold of the data of which 4 folds are used for interactive training and 1 fold for testing. We report the performance as the average of the results of the five folds. In each iteration, we use 2% of the training data to construct the batch and select 10 patterns from the batch for feedback. In all the experiments, we cross-validate the learning-rate (α) and number-of-stage (m) using the validation set and fix $\alpha = 0.3$ and $m = 2000$. We use the least square as a loss function to train the gradient boosted regression tree model. We implement the algorithm in python using scikit-learn [95] package.

Experiment on the Learner's Performance

In this experiment, we evaluate the performance of gradient boosted regression tree (GBRT) based learning algorithm using R^2 coefficient and Mean absolute error (MAE) metric over the entire closed frequent pattern set. The objective is to show whether

Table 5.3
R-squared (R^2) and Mean absolute error of our algorithm on Class score , Jaccard index and Odds ratio based interestingness.

Dataset	Class Score		Jaccard		Odds ratio	
	R^2	MAE	R^2	MAE	R^2	MAE
Chess	0.91	0.023	0.97	0.018	0.86	0.145
Pumsb	0.98	0.009	0.99	0.021	0.95	0.192
EHR	0.99	0.001	0.97	0.003	0.64	0.685
Drug-Side-Effects	0.58	0.12	0.85	0.043	0.88	0.062

GBRT is a good learning model for learning the above interesting metrics over the frequent patterns. We compute R^2 using,

$$R^2 = 1 - \frac{\sum_{i=1}^{M}(y_i - t_i)^2}{\sum_{i=1}^{M}(y_i - \bar{y})^2} \tag{5.9}$$

Here, y_i, t_i and $\bar{y}$ are the observed, predicted and mean of the observed data, respectively. M is the number of test instances. R^2 coefficient indicates how well the trained model fits the data. R^2 coefficient equals to 1 indicates perfect training, we want R^2 coefficient to be close to 1. Mean absolute error (MAE) is computed as,

$$\text{MAE} = \frac{1}{M}\sum_{i=1}^{M}|y_i - t_i| \tag{5.10}$$

We expect MAE to be reasonably small i.e close to zero. In Table 5.3, we report the average R^2 and MAE over 5-fold cross validation. We use all three interestingness metrics that we have discussed earlier. According to the R^2 coefficient, we see that for all the datasets and all different metrics (except class score in Drug-Side-Effects and odds ratio in EHR data), GBRT based learner able to model/fit at least 85% of the data. MAE is also significantly low for almost all the cases.

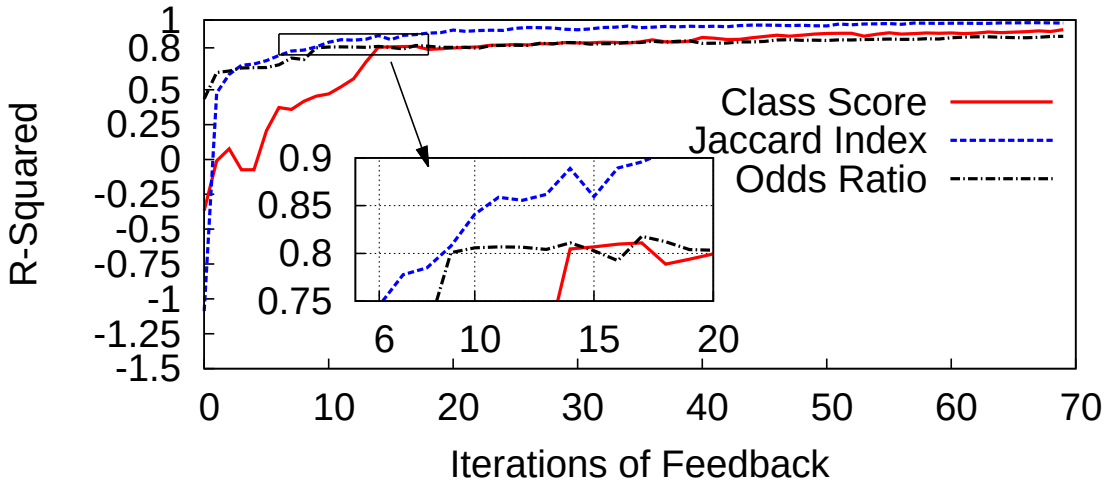


Fig. 5.10. R^2 of the learner across increasing number of feedback in Chess dataset

Learner's Performance with the Iterations of Feedback

In this experiment, we show how the performance of the learning model increases with the number of iterative sessions of feedback. As discussed earlier, in each iteration, the learner selects 10 patterns for feedback and iteratively updates the current model. After each iteration, we measure R^2 of our currently learned model with the test fold. Then we plot R^2 of the learner across the iterations of feedback passed so far. In Figure 5.10, we show the progression of R^2 coefficient with the iterations of feedback for all the interestingness metrics on `Chess` data. As we can see, for Jaccard index and Odds ratio, R^2 reaches to 0.8 only after 10 iterations of feedback and for the class score, it takes around 15 iterations of feedback to reach 0.8. For `Pumsb` data (Figure 5.11(a)), R^2 reaches to 0.9 and stay stable after 10 iterations for all three scores. In EHR data (Figure 5.11(a)), R^2 reaches to 0.9 and stay stable after 10 iterations for all but Odds ratio score. For Jaccard index and Odds ratio, R^2 reaches to 0.75 after 10 iterations and stay stable after 20-25 iterations in Drug-Side-Effects data (Figure 5.11(c)).

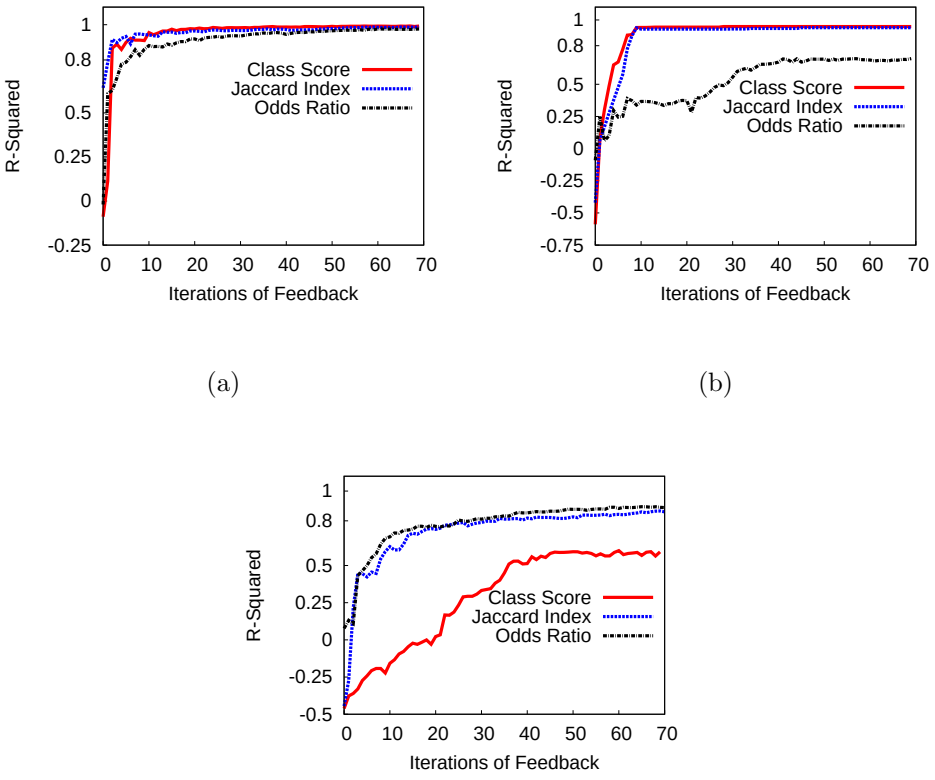


Fig. 5.11. R^2 of the learner across increasing number of feedback in (a) Pumsb (b) EHR and (c) Drug dataset

5.5.8 Comparison with Different Regression Algorithms

We compare GBRT based learner with SVM and Linear regression on all four datasets using odds ratio as interestingness criteria. As we can see in Figure 5.12, GBRT based learner is able to achieve reasonably good performance within 10 iterations of feedback, whereas it requires almost 25 to 30 iterations of feedback for SVM and linear regression based learner to catch the performance of GBRT for all datasets but `Pumsb` (Figure 5.12(b)). So, GBRT is probably the best choice for regression on itemset based transaction data with limited input instances.

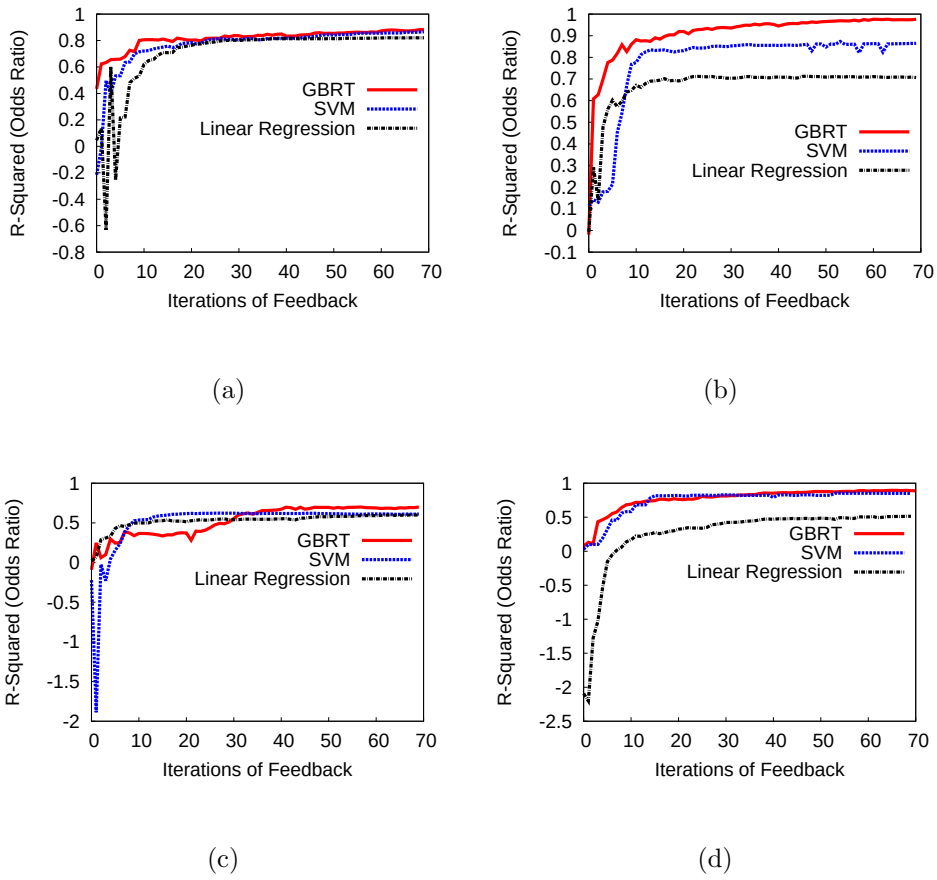


Fig. 5.12. Comparison of GBRT based learner with SVM and Linear regression using odds ratio on (a) Chess (b) Pumsb (c) EHR (d) Drug dataset

5.6 Conclusion

In this work, we propose a generic framework of interactive personalized interesting pattern discovery called PRIIME. The proposed method uses user's rating on a small collection of patterns for learning a user profile model, which at a later stage can be used for recommending patterns that best align with user's interests. Such a method is highly useful for identifying a small number of *interesting* patterns where interestingness is defined through user's rating.

6. RAVEN : WEB-BASED SMART HOME EXPLORATION THROUGH INTERACTIVE PATTERN DISCOVERY

6.1 Introduction

In recent times, online systems are increasingly being used as an integral part of a house hunter's decision-making process. Potential buyers visit various real estate listing sites; they subscribe in online forums related to home market and also read blogs for home buying advice. A recent study by Google and the National Association of Realtors [19] found that nine out of 10 home buyers depend on the Internet as their primary research source. 52% percent of new home buyers start their adventure of buying a dream home from the web. So, an Internet-based system (as different from the listing sites) that actively contributes toward the decision-making process of a home buyer is of great interest. For the US home market, there exist several online products for real-estate search, such as `trulia.com`, `zillow.com`, and `redfin.com`. Each of these products is a home listing resource at their core, augmented with an interface for the user to search the listing repository. Besides the traditional keyword-based search through a search textbox, they also provide faceted search option for a user to apply different filtering criteria, such as zip code, home type, build year, price range, and size in sqft.

However, existing online home search products are generally not fruitful for most of the users for several reasons. First, for a simple search query existing systems produce a long list of potential houses and it is difficult for a buyer to keep patience as she analyzes all the hundreds of returned houses. The process also includes a mental optimization exercise as the user weighs various home features one against another considering her budget. The second issue of existing systems is that for some buyers

construction of a well-defined search query is difficult because of their non-familiarity with all different home features—this is mostly true for the first time home buyers, and certainly true for the immigrant home buyers who, respectively, constitute around 32% and 8% of the home buyers in the USA market [96]. For example, a buyer may be interested into houses with few or no internal walls or partitions but does not know the real-estate jargon `open concept` to express her interest when formulating a search query. So the overall process of selecting the right home takes time; actually, for 40% of first-time home buyers, the lag time between research and action (buying a home) is around 120 days [19].

In recent years, recommendation systems are on the rise to help online buyers to find the most desired products in e-commerce marketplaces (amazon.com, walmart.com) and video streaming (netflix.com, hulu.com). Such systems utilize user feedback in the forms of reviews, ratings, previous purchases, views, and browsing history for learning users' interest profile and then recommend products that the user is likely to purchase [97, 98]. However, recommendation systems are effective for day-to-day consumable items for which an identical product is experienced by many users, such as, books, movies, and electronics, but such a system is not so effective for a once-in-lifetime purchase like a home. Apparently, both collaborative filtering and content-based recommendation fail in the real-estate domain; the former does not work because a home cannot be experienced by many users; the latter does not work, because explicit ratings or reviews on a specific home or home features are not available. In 2013, trulia.com experimented a recommendation system service called `trulia suggests`, which never came out of beta [99].

In this work, we present RAVEN [1], an intelligent real-estate discovery system. RAVEN is superior over the existing products, because existing products work by using information retrieval principles that rank homes in a database by matching up the home description against the keywords in the buyer's query. On the other hand, RAVEN works as a classifier which learns a discriminative function to distinguish

[1]RAVEN is an anagram of the bold letters in **V**irtual **R**eal **E**state **A**ge**N**t

buyer's preferred home from the remaining homes in the database. To train the classifier, RAVEN engages with the buyer in an interactive session where the buyer provides feedback on a collection of home features set, and RAVEN utilizes these feedbacks as supervised labels for training purpose. The learned model is used for generating a small list of homes as a recommendation for the buyer. RAVEN overcomes all the challenges of the existing home discovery products. First, using RAVEN the buyer does not have to master the art of query construction, and reformulation, he simply needs to provide binary feedback on a small number of home-feature sets. Second, RAVEN returns only a small list of highly relevant homes, thus saving the user from the tedious scavenging of the long list of search results to find the desired home.

For practical purposes, RAVEN works as a virtual real estate agent for the home buyers. In real-life for a new buyer, an agent presents an initial set of representative houses with a mix of different features e.g. interior, exterior, neighborhood, school location, and crime. The buyer then visits the potential houses and provides feedback to the agent on the houses; specifically on the features of the houses that she likes or does not like. Over time, the agent learns which features attract buyer the most and in subsequent iterations, he presents houses to the buyer that she is most likely to buy. RAVEN simply mimics the above iterative process. The value proposition of RAVEN comes from the benefit of discovering preferred homes by using less effort, reduction of lag time of a home buyer and the buyer's satisfaction in terms of the quality of recommendation that she receives.

The contributions of this work are the followings:

- We model efficient home search as an interactive itemset pattern discovery problem, and show experimental results on real-life home data to validate its effectiveness.

- We propose a novel algorithm for interactive itemset discovery that we use in RAVEN. The algorithm learns a user's interestingness function by utilizing her

feedback on a small set of itemsets. The framework is iterative and the level of user's interaction with the system is minimum.

- We build a web-demo[2] of RAVEN and perform exhaustive empirical validation of the proposed framework on real-world housing data that we created from `trulia.com`.

The rest of the chapter is organized as follows. In Section 6.2 we discuss the architecture of RAVEN. In Section 6.3, we discuss the underlying algorithmic detail of the proposed system. In Section 6.4, we discuss implementation, deployment and usage details of RAVEN. In Section 6.5, we discuss our methodologies for real-life real-estate data collection and preparation from `trulia.com`. In Section 6.6, we provide empirical evaluation of RAVEN by simulating its effectiveness on three different population demographics. Section 6.7 provides a detailed discussion of the related works. Section 6.8 concludes the chapter with a discussion of future directions.

6.2 Problem Formulation and System Architecture

RAVEN models the home exploration task as an interactive itemset pattern discovery problem. Each house is represented as an itemset, where each itemset is a set of features that the house possesses. RAVEN considers many features for a house, including those that are extracted from the textual description of a house, so this dataset is high dimensional and sparse. To discard insignificant or rare features, RAVEN only considers itemsets that are maximal [1] for a system-defined minimum support threshold. Then the problem of selecting the features-set of the desired house becomes a *pattern classification* problem, where the itemsets representing a desired home belong to the positive class, and the remaining itemsets belong to the negative class. RAVEN solves this classification task in an interactive setup. It gives a user a small collection of itemsets, and she provides a binary feedback on each of the itemsets (likes or does not like) in the collection. Leveraging the feedback as a

[2] `http://mansurul11.pythonanywhere.com/Raven/`

label, the interactive pattern discovery task learns a classification model to classify all the itemsets in the search space. The performance of interactive pattern discovery model is measured by the number of feedback it needs to achieve the expected level of accuracy in the recommendation. Below, we formalize this problem in the context of home exploration.

Consider a dataset, $\mathcal{D} = \{T_1, T_2, \ldots, T_n\}$, where each $T_i \in \mathcal{D}, \forall i = \{1 \ldots n\}$ represents a house. Assume, $\mathcal{I} = \{a_1, a_2, \ldots, a_k\}$ is a set of items where each item a_j is a feature of a house. `3-Bedrooms, 2-Bathrooms, Open Concept, Walkin Closet` are examples of such features. We represent a house T_i as a subset of home features, i.e., $\forall i, T_i \subseteq \mathcal{I}$. Now consider a home buyer u, who has an interestingness profile over the house feature set $\mathcal{I}$, and based on this interestingness profile, she makes a decision regarding whether she wants to buy a house or not. The objective of RAVEN is to learn u's interestingness profile in an interactive fashion by assigning more weights over the items that are in her preferred-list compared to the rest of the items in $\mathcal{I}$ which are not in her preferred-list, and eventually, use those weights for recommending houses that u will like with a higher probability. For example, if {`3-Bedrooms, 2-Bathrooms, Open Concept, Walkin Closet`} is a set of features of *House* 1 and {`3-Bedroom, 2-Bathrooms, Walkin Closet`} is a set of feature of *House* 2, and u has an interestingness profile made of the following features {`3-Bedrooms, 2-Bathrooms, Open Concept`}, u will like the *House* 1 with a higher probability than the *House* 2.

In real-life a house have many different features, which are categorical. It is difficult for a user to analyze such a long list of features of a house and make a decision. So RAVEN provides the user a compact summary of the house features. For achieving this, RAVEN considers the dataset $\mathcal{D}$ as an itemset dataset where each house is a transaction and each feature of the house is an item in that transaction. Frequent pattern mining techniques [1] are effective tool for selecting correlated feature-sets from such a dataset, which RAVEN uses. There are different variants of frequent itemset mining methods, known as frequent, closed and maximal patterns. In this

version of PRIIME, we use the set of maximal itemset patterns ($\mathcal{M}$) as a summary of the dataset $\mathcal{D}$ and present these itemsets to the user for feedback. These summaries are short in length and easy to analyze to constitute a feedback.

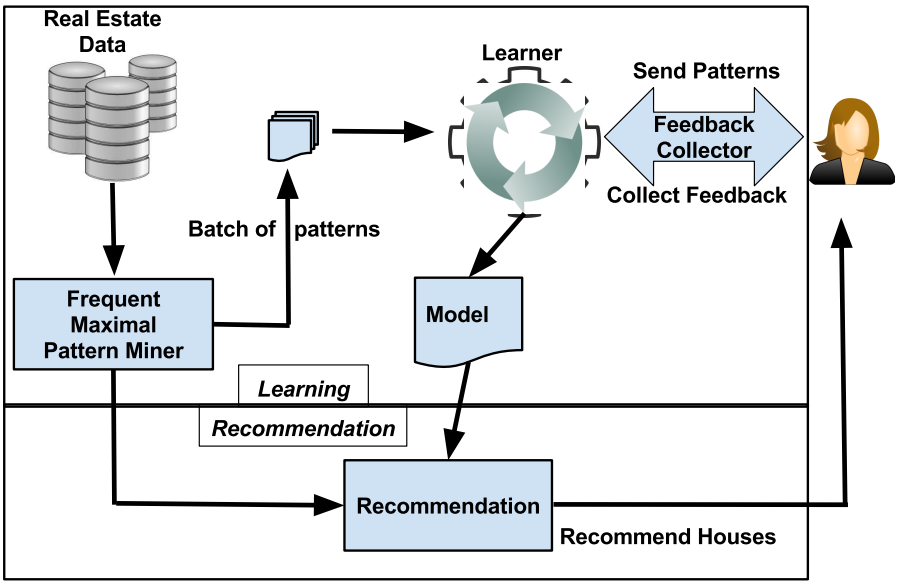


Fig. 6.1. Architecture of RAVEN

We solve the interactive pattern discovery problem as a batch learning task. It learns an interestingness function f which maps a maximal itemset of features to a yes/no decision by using a linear predictor utilizing u's feedback. In our solution, during a learning iteration (interactive session), say i, the system presents a set of frequent maximal patterns $\{P_t\}_{1\leq t\leq l}$ to u, which it selects from a partition of $\mathcal{M}$ (constitutes the ith batch of a pattern set); u sends feedback y_t using an interactive console; we consider y_t's to be a binary number reflecting the quality of the selected patterns. RAVEN updates its current model of the user's interestingness function, f using the feedback. An alternative to our batch learning framework can be to mine all the frequent maximal patterns and select a relatively smaller subset of patterns for getting a user's feedback and use these for learning in one iteration. This approach is not scalable as the number of frequent maximal patterns is typically large even for a moderate-size dataset in a low support setting. Besides, if the learning entity receives

the frequent patterns as a data stream, the one-iteration learning becomes infeasible, but a batch learning framework still works.

The framework of RAVEN is same as the PRIIME's as discussed in Section 5.2 in Chapter 5. For the ease of reader we redraw the framework of RAVEN with real-estate notation in Figure 6.1 The proposed system has two main blocks: *Learning* and *Recommendation*, which are shown in the upper and lower box of Figure 6.1, respectively. *Learning* learns a model for a user; it contains the following three modules: Frequent Maximal Pattern Miner (FMPM), Learner, and Feedback-Collector (FC). The FMPM module is an off-the-shelf pattern mining algorithm that mines maximal frequent patterns given a normalized minimum support threshold. Any of the existing pattern mining algorithms ranging from sequential [94, 100] to randomized [53], can be used. The Learner learns u's interestingness function (f) over the maximal pattern set $\mathcal{M}$ (summary of the dataset) in multiple stages such that each stage uses the user's feedback on a small number of patterns selected from a partition of $\mathcal{M}$. The Feedback-Collector (FC)'s responsibility is to identify the patterns that are sent to the user for feedback. The *Recommendation* suggests houses to the user using the learned model.

6.3 Method

The learning model of RAVEN is similar to PRIIME explain in Section 5.3 in Chapter 5. Nevertheless, for better understanding in this section we explain the learning and feedback collection module of PRIIME along with pattern representation from the view point of house recommendation.

6.3.1 Data and Pattern Representation

Considering the dataset $\mathcal{D}$ (discussed in Section 6.2) as a text corpus and each transaction in the corpus as a document, the set $\mathcal{I}$ can be represented as the dictionary

of frequent words i.e. items. Following the bag of words model, we represent each pattern $p \in \mathcal{O}$ as a binary vector V of size $|\mathcal{I}|$, constructed as below:

$$V(i) = \begin{cases} 1, & \text{if item } i \in p \\ 0, & \text{otherwise} \end{cases}$$

Such a representation maps each pattern to a data point $\mathbf{x} \in \{0,1\}^{|\mathcal{I}|}$ space. For example, in a data $\mathcal{D}$, $\mathcal{I}$ = {`3-Bedrooms, 3-Bathrooms, Open concept, Walkin Closet`}. A pattern {`3-Bedrooms,3-Bathrooms`} will be represented by the vector $(1,1,0,0)$, but a pattern {`3-Bedrooms,Open concept, Walkin Closet`} will be represented by the vector $(1,0,1,1)$ and so on. For the remaining sections, we will use $\mathbf{x}$ to denote a pattern P.

6.3.2 Classification Model

The RAVEN models the home buyer's interestingness profile over the house features set as a classification problem and learns the model parameters through an interactive feedback mechanism. In each iteration, RAVEN executes a supervised training session of the classification model using the corresponding data points of the released patterns to the user. RAVEN uses the binary feedback (0/1) provided by the user as the label of the training data. We use logistic regression as the classification model $H_{\boldsymbol{\theta}}$. In logistic regression, the probability that a data point $\mathbf{x}_i \in \mathbb{R}^d$ belongs to class y_i, where $y_i \in \{0,1\}$ can be written as,

$$H_{\boldsymbol{\theta}}(\mathbf{x}_i) = p(y_i|\mathbf{x}_i;\boldsymbol{\theta}) = \frac{1}{1 + exp(-\boldsymbol{\theta}^T\mathbf{x}_i)} \tag{6.1}$$

$\boldsymbol{\theta}$ is the weight vector. The prediction of the model is computed as, $\hat{y}_i = \text{argmax}_j\, p(y_i = j|\mathbf{x}_i,\boldsymbol{\theta})$, where $j \in \{0,1\}$. Given a set of training instances $X = \{(x_1,y_1),\cdots,(x_m,y_m)\}$, the weight matrix $\boldsymbol{\theta} \in d$ (d is the size of a feature vector) is computed by solving the convex optimization problem as shown in Equation 6.2 using gradient descent. Note that, each data points $\mathbf{x}_i \in \mathbb{R}^d$ in the training set corresponds to a maximal itemset pattern.

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log\ p(y_i|\mathbf{x}_i;\boldsymbol{\theta}) + \ (1-y_i)\log\ (1-p(y_i|\mathbf{x}_i;\boldsymbol{\theta})) + \frac{\lambda}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} \quad (6.2)$$

$\frac{\lambda}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$ is the regularization term to avoid over-fitting. In each iteration, RAVEN applies L-BFGS-B optimization function implemented in python's scipy package to train the model.

6.3.3 Selection of Representative Data-points for Feedback

As discussed in Section 6.2, the Feedback-Collector (FC) is responsible for selecting a small set of patterns for feedback. Considering the ease of a user while providing feedback, we choose to present five patterns to the user at a time. RAVEN uses a exploitation-exploration based criteia to choose patterns. Exploitation ensures that the selected patterns have high impact on the training and exploration ensures that the patterns are diverse. In each iteration, for an incoming batch, RAVEN exploits the currently learned model to identify the patterns that will have the largest impact on the parameters of the learning model. We can identify those patterns by checking whether these patterns make the maximum change in the objective function. Since we train the logistic classifier using gradient descent, we include the patterns to the training set if it creates the greatest change in the gradient of the objective function,

$$\bigtriangledown_\theta J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[\mathbf{x}_i\left(y_i - p(y_i|\mathbf{x}_i;\boldsymbol{\theta})\right)\right] + \lambda\boldsymbol{\theta} \quad (6.3)$$

However, gradient change computation requires the knowledge of the label, which we don't have. So, we compute the expected gradient length [91] of $\mathbf{x}_i$ using Equation 6.4.

$$EGL(\mathbf{x}_i) = \left(p(y_i|\mathbf{x}_i;\boldsymbol{\theta}) + (1-p(y_i|\mathbf{x}_i;\boldsymbol{\theta}))\right)|| \bigtriangledown_\theta J(\boldsymbol{\theta})|| \quad (6.4)$$

After exploitation step, RAVEN retains the top 50% of patterns ($\mathbf{x}_i$s) according to the expected gradient length (EGL) as candidates for feedback. However, Expected

gradient length based approach enables RAVEN to identify patterns that can make greatest changes in the learning model, but it does not ensure diversity among the patterns in terms of its composition, i.e. two patterns can be made of similar items. To ensures that a broader span of the pattern space is covered, RAVEN uses k-center search (exploration) to identify k patterns for feedback from the candidates set.

k-Center. k-center method finds k data points as centers so that the maximum distance from each data point to its nearest center is minimized. This k-center problem is NP-complete, but there exists a fast 2-approximation greedy algorithm that solves this problem in $O(kn)$ time [92]. We set k to 5 in all our experiments. For choosing 5 representative patterns, we use the greedy 5-center algorithm considering the Jaccard index as the distance between two patterns.

6.4 RAVEN: The System

To bring our ideas of efficient house exploration into reality, we build an Internet-based system for RAVEN live at `http://mansurul11.pythonanywhere.com/Raven/`. Code and data is available at `https://github.com/mansurul11/Raven`. Please visit `https://youtu.be/x-5bEXeGtxA` for a demo video of the system. The developed system has three primary modules: first level feedback, second level feedback, and house recommendation. In the following, we discuss each of these modules.

6.4.1 First Level Feedback Module

First Level Feedback is the most important module of RAVEN. Using this module, it models a home buyer's interestingness profile from the explicit feedback provided by the buyer. As we can see in Figure 6.2, RAVEN uses HTML `checkbox` component to capture the home buyer's positive (1) feedback on a pattern. To help a home buyer for constructing appropriate feedback, it also provides the price (Median and Average) of the houses associated with this pattern. For a new home buyer, it might be difficult to understand a house feature just from the keyword. we use `image`

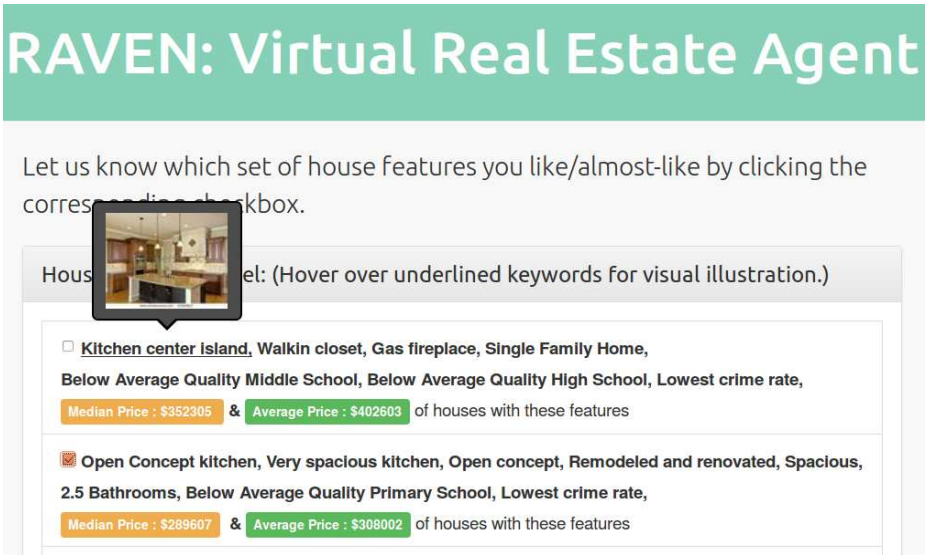


Fig. 6.2. User interface of first level feedback module of RAVEN.

`tool-tip` technique to show the corresponding image of the feature. At the end of pattern list, the system plants another `check box` (not shown in Figure 6.2) for the user to specify whether the current feedback session is the last one. Note that in the first iteration (initialization), RAVEN uses the k-center search to find 5 patterns for feedback.

The First level feedback session can be halted from the system side as well as the user side. RAVEN sets a minimum default iteration count to 10. After 10th iteration, it keeps track whether getting additional feedback impact significantly on the learning model or not by computing $||\boldsymbol{\theta}^i - \boldsymbol{\theta}^{i-1}||$, where $\boldsymbol{\theta}^i$ and $\boldsymbol{\theta}^{i-1}$ are the trained model parameter vector of iteration i and $i-1$, respectively. If $||\boldsymbol{\theta}^i - \boldsymbol{\theta}^{i-1}||$ falls below a threshold $(1E-4)$, execution of the first level feedback session halts and execution of the second level feedback session starts. In addition to above criterion, the user can also signal feedback termination by marking a checkbox in the GUI of RAVEN.

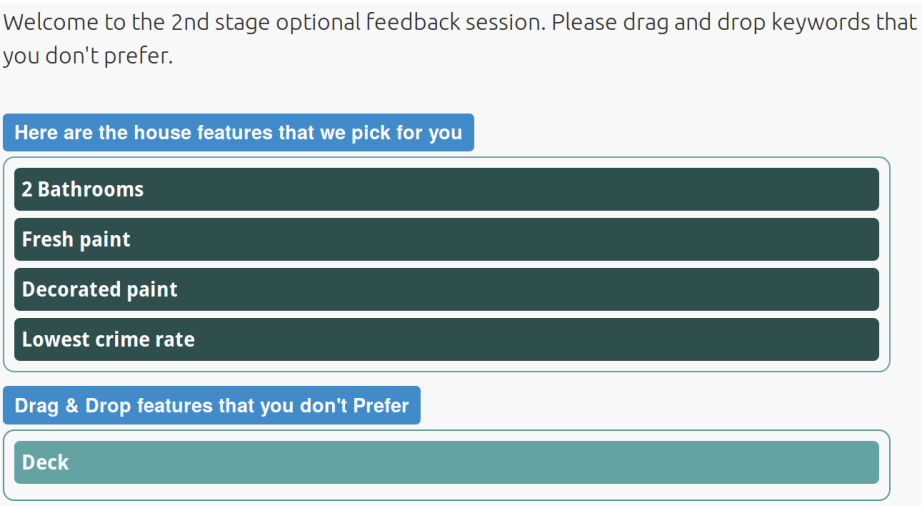


Fig. 6.3. Partial user interface of second level feedback module of RAVEN

6.4.2 Second Level Feedback Module

During the first level feedback, RAVEN has no assumption on whether the user who provides a positive feedback for a pattern likes all the features in that pattern or not. If a home buyer is strict, then she might give positive feedback for a pattern only if she likes all the features mentioned in a pattern. On the other hand, the casual home buyer might give positive feedback for a pattern if she likes some of the features. In the second level feedback, RAVEN presents a current compiled list of house features to the user. The user performs a final assessment and decide if she wants to remove any of the features. It is up to the user if she wishes to proceed without filtering. In Figure 6.3, we present the UI (User Interface) for second level feedback module. RAVEN uses `drag & drop` like component for second level feedback module. The system reflects the second level feedback by identifying the filtered out house features from the drop list and set the corresponding learned weights of these features to zero. That way, the dropped features do not contribute towards selecting the houses for the recommendation.

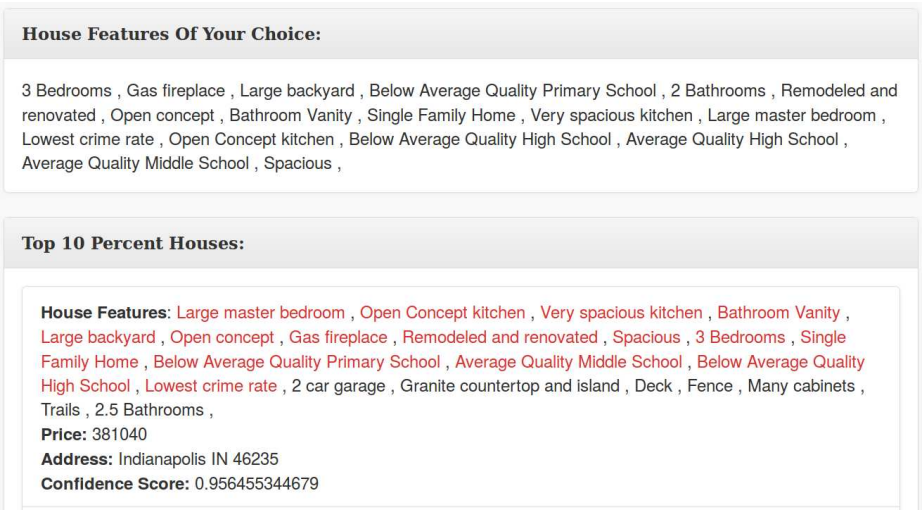


Fig. 6.4. Partial user interface of final house recommendation module of RAVEN

6.4.3 House Recommendation Module

The recommendation is the final module of the system. After the second level feedback, the system has the final trained model of a user's interestingness over house features. RAVEN applies the learned model over the house data and identify top 10 percent houses according to the positive class probability. To present the result, the system first mentioned what are final features compiled by the algorithm according to the user's feedback. After that, it lists top 10 percent houses along with the price, confidence score (probability) and location. In Figure 6.4, we present the UI for this module. RAVEN marks the features of a house using red color if that feature is coming from final features set.

6.4.4 Implementation Detail

RAVEN is built using python's `Django` [101] framework. The background learning algorithm is written in python using numpy and scipy math packages. For UI design, we use `Bootstrap` [102]. We use `jQuery` [103] to implement `tooltip` and `drag & drop` component. To facilitate the live demo, we host RAVEN in

pythonanywhere.com. In the live demo we set the batch size to 2000 and send 5 patterns to the user for feedback, which are configurable system parameters.

6.5 Data

To validate the effectiveness of RAVEN we crawl trulia.com to build a real-world housing dataset. In this section, we discuss the data collection, and cleanup methodologies. We also provide some basic data statistics.

6.5.1 Data Collection

We crawl trulia.com from November 2015 to January 2016 (3 months) for five major cities: Carmel, Fishers, Indianapolis, Zionsville and Noblesville in the Central Indiana. The information that we crawl is the basic house information, text on house detail, school, and crime information. We extract the address of a house from the URL. We use python to write the data crawler. We choose Central Indiana home market because of our familiarity with the region.

6.5.2 Data Cleanup

To clean the crawled text data we use standardized data cleaning approach, i.e. remove stop words, stemming and lemmatization of words. We have manually cleaned some keywords that are written in unstructured abbreviated form. For example, keyword fireplace is written as frplc or firplc in many house details. In order to compile house features set, we use Rake [104] to find the key phrases. Next, we group the key phrases, where we keep two key phrases together if they share at least one keyword. That way, we get all the features of a category together. For example, all the key phrases related to basement are grouped together by using this step. Finally, we manually investigate these groups and identify specific house features. In the end, we retain only those house features that appear in at least 25 houses.

Table 6.1
Number of house in each city

City	Number of House
Carmel, IN	597
Fisher, IN	525
Indianapolis, IN	5,485
Zionsville, IN	211
Noblesville, IN	389

6.5.3 Data Statistics

As mentioned earlier, we crawl house data of five major cities in Central Indiana, USA region. In total, we crawl $7,216$ houses. In Table 6.1, we show the breakdown of the house count in five cities. The total number of house features we extracted from the crawled data is 123. The house features are categorized into three groups. The first group of house features are about the interior and exterior of a house. Interior features can be divided into several categories, i.e. details of bedrooms, bathrooms, kitchen, garage, different amenities, etc. Information regarding fence, yard, porch, etc. are considered as exterior features. The second group consists of neighborhood features i.e. nearby schools, shopping malls, restaurants, trails, etc. The final group is the school quality and crime rate information of the house neighborhood.

6.6 Empirical Evaluation on Housing Data

Due to unavailability of the real-estate purchase data by actual buyers, we empirically evaluate RAVEN by demonstrating its performance over three different demographic groups of home buyers. We identified these groups from a yearly report of `realtor.com` [105]. These groups are: (1) young married couple without kids; (2) median-income couple with kids; and (3) high-income professional couple. We

also extrapolate the typical home features of these groups from the same information source. In the following, we provide more details of our selection.

6.6.1 Demographic Group 1

People in this demographic group are of age 34 or younger. According to [105], 80% people in this group prefer `Single Family Home` and 8% prefer `Townhome`. So, we select these two features. 21% of young peoples of age less than 34 prefer to be close to city/downtown, 29% want their home to be close to restaurants, because of that, we include `Close to Downtown`, `Nearby Restaurants` and `Nearby shopping-stops` in the preferred feature set. 58% people of these group want to stay close to their friends and family. They want features in the house appropriate for hosting parties. We choose some of the home features that are suitable for this purpose, such as, `Kitchen center island`, `Granite countertop and island`, `Breakfast bar and nook`, and `Balcony`. According to the report [105], 52% of the young peoples prefer to avoid any kinds of renovation, issues with paint, etc. So we choose `Movein ready` and `Fresh paint`. In general, most home buyers prefer a location with `Lowest crime rate`. This group typically have no kids, `good-school` is not a concern for them.

6.6.2 Demographic Group 2

People in this demographic group are of age between 35 to 49. According to the report [105], 85% of peoples in this group prefer single family home and 68% of them prioritize suburb area. 43% of the individuals in this group want good quality, nearby schools, so we choose `Nearby schools`, `Above Average Quality Primary School`, and `Above Average Quality Middle School` and `Single Family Home` as preferred house features. 24% of families with kids prefer nearby parks, playground, because of that, we include `Neighborhood playground`, `Trails`, and `Community pool` in the feature set. 50% of this demography have kids and most likely they prefer houses with open concept design, spacious, easy to maintain, common and work

area, and large kitchen with necessary amenities. With that in mind, we extend the feature set by, `Open concept`, `4 Bedrooms`, `3 Bathrooms`, `2 car garage`, `finished basement`, `Large master bedroom`, `Hardwood floor`, `Open Concept kitchen`, `Steel appliance`, `Granite countertop and island`, `Dual sink`, `Many cabinets`, `Common area`, `Work area` and `Large backyard`. Finally, `Lowest crime rate` location is a natural choice for this group.

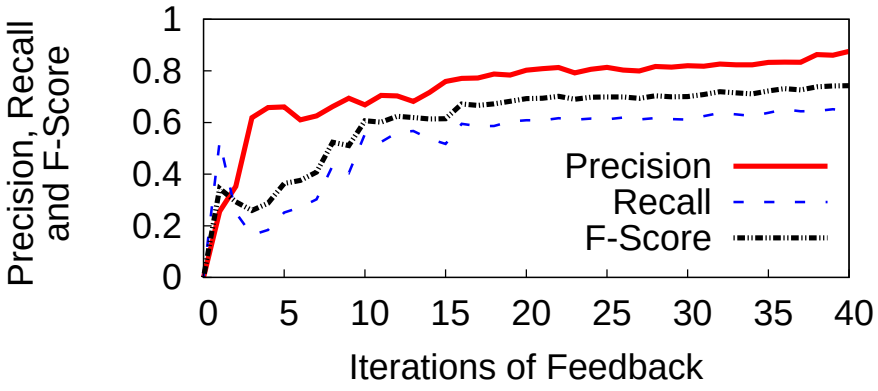


Fig. 6.5. Performance of the learner with across iterations of feedback for demographic group 1

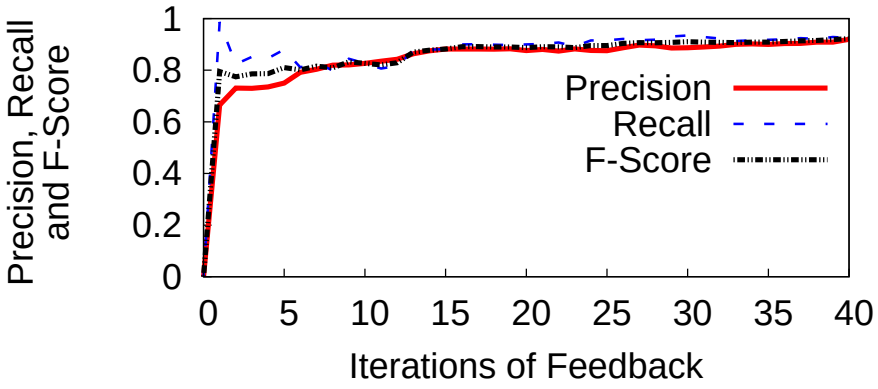


Fig. 6.6. Performance of the learner with across iterations of feedback for demographic group 2

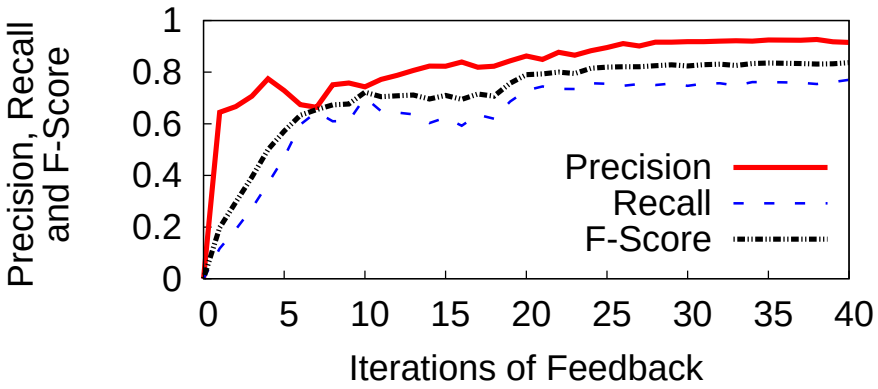


Fig. 6.7. Performance of the learner with across iterations of feedback for demographic group 3

6.6.3 Demographic Group 3

In this particular demographic group, we consider wealthy couples, interested to buy luxurious, high-end houses. Since, for this group luxury is the primary concern, we choose following house features that usually indicate a high-end house. `Large Master Bedroom`, `Marble floor`, `Gourmet kitchen`, `4 car garage`, `Walkin closet`, `Wood fireplace`, `Kitchen center island`, `Bathroom Vanity`, `Recreation area`, `Crown molding`, `High ceiling`, `Jacuzzi hot tub`, `Large backyard`, `Quartz countertop and island`, `Celling windows`, `Deck`, `Boat dock`, `Formal dining`, `Wet bar`, `Walkin pantry`, `Neighborhood golf course`, `5 Bedrooms`, `5 Bathrooms`, `Low crime rate`.

6.6.4 Experimental Setup

After generating features-set associated with a demographic group, we first create labels for the maximal patterns mined from `trulia.com` data with 25 support. We choose support threshold in a way so that RAVEN can leverage a large number of maximal patterns to learn the user's choice. Support threshold 25 also allows not to filter any of the 123 house features. We assign label 1 to a pattern if it contains

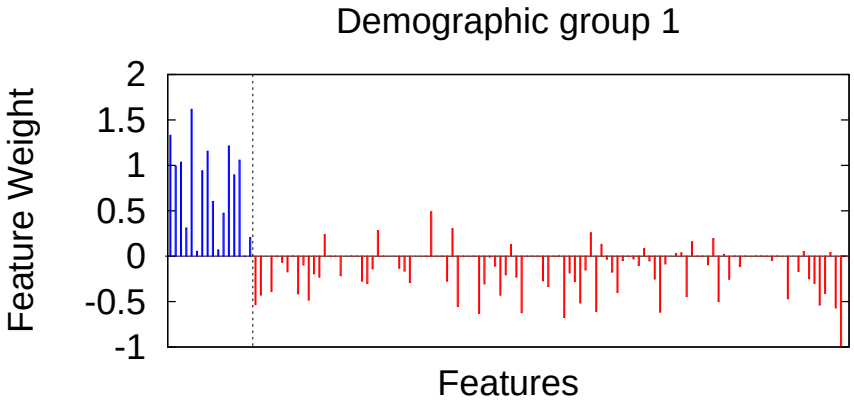


Fig. 6.8. Quality of training on the demographic group 1

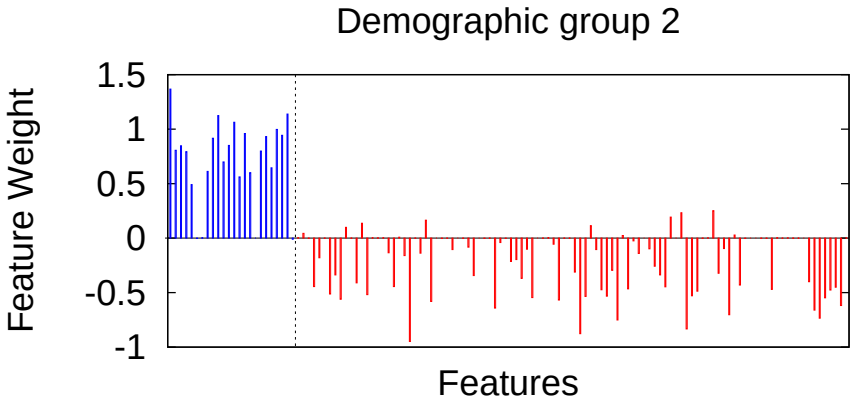


Fig. 6.9. Quality of training on the demographic group 2

80% or more features from the devised features set. We follow similar label creation protocol for all groups. Next, we split the pattern set into train and test and execute RAVEN over the training split. For all cases, we set the batch size to 2000 and consider 5 patterns for feedback in each iteration. Once training is done, we compute classification performance of RAVEN in terms of "precision", "recall" and "F-Score".

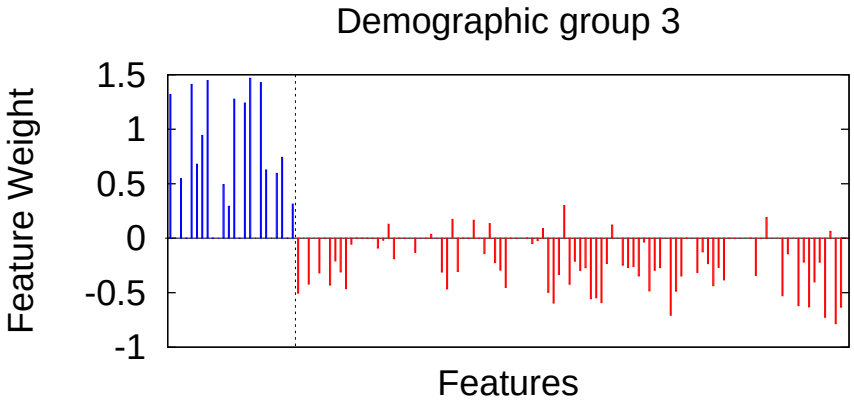


Fig. 6.10. Quality of training on the demographic group 3

6.6.5 Observations

For each of the demographic groups, we observe how PRIIME's performance changes as more and more feedback comes in. Figure 6.5 shows the observation for group 1, Figure 6.6 and Figure 6.7 shows the same for group 2 and group 3, respectively. In each figure, x-axis shows the number of feedback iterations and the three curves represent precision, recall, and F-Score over test fold for different iteration count. As we can see, in all cases the performance of the system improves with the increase in the number of feedback. After 10 interactive sessions, RAVEN reaches to a reasonable performance; for group 1, 2 and 3, the precision value reaches approximately 0.7, 0.8, and 0.75, respectively. RAVEN reaches to a stable position in terms of performance within 10 to 15 iterations of feedback for all three cases. For group 1, 2 and 3, after 20 iterations of feedback, (recall, F-Score) become (0.69, 0.6), (0.89, 0.88), and (0.79, 0.73), respectively. When we compare accross demographic groups, group 2 and 3 produce better performance than group 1. The reason is that the total number of positive patterns according to the user's choice is comapartively low for group 1 than 2 and 3.

To illustrate the training quality of learning module in PRIIME, we create a bar chart with learned features weights for all the cases. To visually show the distinction

between weights of the selected features set from weight of the rest of the features, we divide the plotting area into two regions, in the left region we display weights of the selected features set in blue ink and in the right region we show weight of the rest of the features in red ink. In Figure 6.8, 6.9 and 6.10, we show the observation. As we can see, PRIIME's learning algorithm puts positive weights on all the features from the selected set and negative weight to the majority of other features. Few features in the right side of the plots have positive weight and we investigate this scenario for the demographic groups. For group 1, top of these features (according to weight) is `Remodeled and renovated`, which is related to the selected `movein ready` and `Fresh paint` features. Next for group 2, we found that top two (according to weight) of these features are `many windows` and `counter space` which are related to the selected `open concept` and `open concept kitchen` feature. Finally for group 3, top of these features is `boat dock`, which is a very common luxurious feature of houses in Carmel, IN. Note that, the learned features weights used in the bar chart is collected after 10 iterations of feedback.

6.6.6 Analysis of Recommendation Quality

In this experiment, we analyze the location of top 20 recommended houses for each of the groups. In Table 6.2, we present our finding. For all groups the majority of the recommended houses are from Indianapolis area. The reason is that, city-wise our dataset is highly imbalance where 76% of houses in our dataset are from Indianapolis. However, if we consider the second most recommend cities we see that for group 2, recommended houses are from Fishers and Carmel. In central Indiana area, these two cities have the best school and family environment. For the group 3, we observe that Carmel is the second most recommend city and Carmel, IN has most of the high-end houses among the five cities.

Table 6.2
Recommend city for demographic group 1, demographic group 2, and demographic group 3 (Indy is the short form in Indianapolis)

Demographic group 1	Demographic group 2	Demographic group 3
Carmel IN 46032	Carmel IN 46032	Carmel IN 46032
Fishers IN 46037	Carmel IN 46033	Carmel IN 46032
Indy IN 46237	Carmel IN 46074	Carmel IN 46032
Indy IN 46237	Fishers IN 46037	Carmel IN 46033
Indy IN 46222	Fishers IN 46037	Carmel IN 46033
Indy IN 46227	Fishers IN 46038	Carmel IN 46033
Indy IN 46236	Fishers IN 46038	Fishers IN 46037
Indy IN 46220	Indy IN 46201	Indy IN 46203
Indy IN 46268	Indy IN 46203	Indy IN 46203
Indy IN 46256	Indy IN 46203	Indy IN 46208
Indy IN 46218	Indy IN 46203	Indy IN 46218
Indy IN 46208	Indy IN 46208	Indy IN 46219
Indy IN 46208	Indy IN 46208	Indy IN 46220
Indy IN 46201	Indy IN 46217	Indy IN 46229
Indy IN 46203	Indy IN 46219	Indy IN 46231
Indy IN 46202	Indy IN 46227	Indy IN 46236
Indy IN 46208	Indy IN 46234	Indy IN 46237
Indy IN 46202	Indy IN 46239	Indy IN 46260
Indy IN 46280	Indy IN 46241	Noblesville IN 46060
Zionsville IN 46077	Zionsville IN 46077	Noblesville IN 46060

6.7 Related Works

We discuss the related works under three different categories.

6.7.1 Real Estate Price Modeling

There exist several works on real-estate price modeling. The primary objective of these works is real estate appraisal, which is the process of estimating the market value for an estate or property. Krainer et al. [106] model real estate price by building metrics which are inspired from financial theory, such as, price to income ratio, and price to rent ratio. Nagaraja et al. [107] leverage financial time series data to identify different patterns in estate prices i.e. periodicity, and volatility. Downie et al. [108] observe that the low investment-valued estates are more volatile than the high investment-valued estates. Hedonic regression [109] technique is one of the popular methods for estate appraisal in real-life.

In recent years, there are multiple works [110–112] on real-estate ranking using investment value index. Fu et al. [110] propose a ranking algorithm called ClusRanking that leverages the characteristics of a neighborhood to estimate the values of its nearby estates. In another work [112], the authors propose the sparse estate ranking algorithm which uses opinion about real estates from online reviews and mobility behaviors (e.g., taxi traces, smart card transactions, check-ins). The most recent work [111] on estate ranking incorporates the functional diversity of user communities in a locality. Thus, it leverages different temporal urban functions (when most people in a community going to work or shop, eat, etc.), and user mobility information (public transportation traces, taxi traces) to rank real estates. All the above methods are beneficial to a person who is in real-estate business i.e. flipping houses[3]. However, in their model, they do not use house features, so they are not useful for a home buyer who wants to buy a house to live in it.

6.7.2 Commercial Real Estate Search Products

`realtor.com`, `trulia.com`, `zillow.com` and `redfin.com` are the most popular commercialized online real-estate search products that are available in the US market.

[3]Buy a house, renovate it and sell at a higher price

These are basically listing sites, where they provide customize search tools to find houses for rent, sell or buy. They also provide information on mortgage, real-estate agent, etc. In all of these sites, a registered user can save homes and get notifications of any updates on the save homes as well as similar homes. redfin.com can populate a recommend list of houses for a registered user based on the "save homes" feature.

6.7.3 Real Estate Recommendation

There exist a few works on real estate recommendation [113,114]. In [113], authors proposed a system called JOURA (short of Journey-Aware) that can recommend houses solely based on transportation information. For example, using JOURA a user can specify the locations she requires to commute frequently and the system finds houses optimizing the commuter time while considering price as a parameter. In [114], authors developed a case based reasoning house search system that can find houses based on a user's wish-list populated on the basis of location. The system also takes consideration of house price, bedroom, bathroom, size information to narrow down the search. Both of the methods only consider basic house features i.e price, size, bedroom, bathroom along with location and transportation based information. But for a home buyes other housing features i.e. school, neighborhood, entertainment, house interior/exterior features is important to consider that we do in the RAVEN.

6.8 Future Directions and Conclusion

Future directions of RAVEN are abundant. In the current version, we only consider the binary form of feedback. In the next version, we want to incorporate a more elaborate form of feedback, where a home buyer can express her empathy towards a house summary in multiple levels. For example, a home buyer can choose feedback from `yes`, no and `maybe` for a house summary set. We can add more features in the user interface of RAVEN. For example, On the recommendation page, we can present houses with more information i.e. images, text description, additional meta

data. Currently, we are showing top 10% houses without price filtering. We plan to introduce a price range selector, that enables the user to select a price range to filter the recommended houses. We also plan to present houses in a table fashion where the user can sort the results based on different house features. Mobile application of RAVEN is another exciting future works.

To conclude, in this chapter we introduce a new house discovery tool called RAVEN that uses frequent maximal itemset mining and interactive feedback from the user to learn the user's interestingness profile over the house features. In the end, RAVEN recommend houses to the user with a higher probability of liking it.

7. FUTURE WORKS AND CONCLUSION

Interactive Personalized Interesting Pattern Discovery is a novel research problem, and our works are just the beginning, so the opportunity for future work is abundant. In this section, we will discuss possible forthcoming works on the frameworks we developed.

7.1 Sampling Based Solution

One of the crucial contribution of this framework is that, it can sure entire dataset reconstruction privacy while allowing personalized patten discovery. From the privacy point of view followings can be potential future works.

Safeguarding the system from manipulative user: In this work, we have considered a semi-honest model whereas in real-life a malicious model may be needed. Specifically, an interesting follow-up research could be to un- derstand how to safeguard the interactive mining system from a malicious user who intends to reconstruct the entire data-set by exploiting the sampling system. Even more interesting is the scenario when a group of malicious users jointly attack the system to achieve a similar goal.

Robust data privacy model: In this work, our proposed system assumes that the released patterns are not sensitive. It ensures that the data owner maintains the confidentiality of the data in the sense that using the released set of patterns a user cannot reconstruct the entire data-set with a high level of accuracy. In practice, such assumption might not be compatible to ensure the privacy for all kinds of private data. For example, in financial or medical data, patterns can contain sensitive information, hence prone to privacy violation for individuals in the data. As a future work, we want to extend our data privacy model to ensure both entire data-sets as well as

individuals confidentiality and provide a complete theoretical contribution using the state of the art privacy models i.e. differential privacy.

7.2 Pattern Representation:

In this dissertation, we design the "Pattern2Vec" model using the existing neural net-based model from NLP domain. As an imminent future work, we aim to improve the current model and develop a neural net-based solution while incorporating pattern specific information in the optimization function. We also want to explore the applicability of such techniques in the graph classification domains.

7.3 RAVEN : The Home Discovery System

In the current version, we only consider the binary form of feedback. In the next version, we want to incorporate a more elaborate form of feedback, where a home buyer can express her empathy towards a house summary in multiple levels. For example, a home buyer can choose feedback from yes, no and maybe for a house summary set. We can add more features in the user interface of RAVEN. For example, On the recommendation page, we can present houses with more information i.e. images, text description, additional meta data. Currently, we are showing top 10% houses without price filtering. We plan to introduce a price range selector, that enables the user to select a price range to filter the recommended houses. We also plan to present houses in a table fashion where the user can sort the results based on different house features. Mobile application of RAVENis another exciting future works.

We also want to explore other business domains where IPIPD framework can be used to solve information overload problem. One of such business domain can be the auto industry. Similar to home, new car buyer faces the similar dilemma on deciding what car to buy with their budget, since there are so many options available. We believe like home industry, buyers in the auto industry will spend their time to

leverage such interactive system and gather a smaller set of potential cars that they might bought.

To conclude, in this thesis, we introduce several frameworks for interactive personalized interesting pattern discovery problem. We developed adopting interactive pattern sampling framework, where the effective sampling distribution is continuously updated by binary feedback from the user and allow personalized discovery from hidden dataset. We also developed supervised learning based generic framework utilizing neural net based efficient feature construction scheme of different types of pattern i.e. sequence, graph. To show real-life application of the proposed framework, we build a smart house discovery tool called RAVEN that can be helpful to find houses in an interactive manner.

LIST OF REFERENCES

LIST OF REFERENCES

[1] M. J. Zaki and J. Wagner Meira, *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 1st ed., 2014.

[2] M. Mampaey, J. Vreeken, and N. Tatti, "Summarizing data succinctly with the most informative itemsets," *Transactions on Knowledge Discovery from Data*, vol. 6, no. 4, pp. 1–42, 2012.

[3] J. Vreeken, M. Leeuwen, and A. Siebes, "Krimp: Mining itemsets that compress," *Data Mining Knowledge Discovery*, vol. 23, no. 1, pp. 169–214, 2011.

[4] X. Yan, H. Cheng, J. Han, and D. Xin, "Summarizing itemset patterns: A profile-based approach," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 314–323, ACM, 2005.

[5] A. K. Poernomo and V. Gopalkrishnan, "CP-summary: A concise representation for browsing frequent itemsets," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 687–696, ACM, 2009.

[6] G. Liu, H. Zhang, and L. Wong, "Finding minimum representative pattern sets," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 51–59, ACM, 2012.

[7] G. I. Webb, "Self-sufficient itemsets: An approach to screening potentially interesting associations between items," *Transactions on Knowledge Discovery from Data*, vol. 4, no. 1, p. 3, 2010.

[8] H. Heikinheimo, E. Hinkkanen, H. Mannila, T. Mielikäinen, and J. K. Seppänen, "Finding low-entropy sets and trees from binary data," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 350–359, ACM, 2007.

[9] D. Xin, X. Shen, Q. Mei, and J. Han, "Discovering interesting patterns through user's interactive feedback," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 773–778, ACM, 2006.

[10] M. Boley, M. Mampaey, B. Kang, P. Tokmakov, and S. Wrobel, "One click mining: Interactive local pattern discovery through implicit preference and performance learning," in *Proceedings of the Workshop on Interactive Data Exploration and Analytics*, pp. 27–35, ACM, 2013.

[11] B. Settles, "Active learning literature survey," *Technical Report of University of Wisconsin, Madison*, vol. 52, pp. 55–66, 2010.

[12] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proceedings of the International Conference on Very Large Data Bases*, vol. 1215, pp. 487–499, ACM, 1994.

[13] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proceedings of the International Conference on Management of Data*, vol. 29, pp. 1–12, ACM, 2000.

[14] M. Bhuiyan, S. Mukhopadhyay, and M. A. Hasan, "Interactive pattern mining on hidden data: A sampling-based solution," in *Proceedings of the International Conference on Information and Knowledge Management*, pp. 95–104, ACM, 2012.

[15] M. Bhuiyan and M. A. Hasan, "Interactive knowledge discovery from hidden data through sampling of frequent patterns," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 9, no. 4, pp. 205–229, 2016.

[16] C. C. Aggarwal, *Data Classification: Algorithms and Applications*. CRC Press, 1st ed., 2014.

[17] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Effective graph classification based on topological and label attributes," *Statistical Analysis and Data Mining*, vol. 5, no. 4, pp. 265–283, 2012.

[18] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents.," in *Proceedings of the International Conference on Machine Learning*, vol. 14, pp. 1188–1196, 2014.

[19] A. J. S. from The National Association of Realtors and Google, "The digital house hunt: Consumer and market trends in real estate," *National Association of Realtors*, 2013.

[20] R. J. Bayardo Jr, "Efficiently mining long patterns from databases," *ACM Sigmod Record*, vol. 27, no. 2, pp. 85–93, 1998.

[21] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *Proceedings of the International Conference on Database Theory*, pp. 398–416, Springer, 1999.

[22] J. Han, J. Wang, Y. Lu, and P. Tzvetkov, "Mining top-k frequent closed patterns without minimum support," in *Proceedings of the International Conference on Data Mining*, pp. 211–218, IEEE, 2002.

[23] F. Afrati, A. Gionis, and H. Mannila, "Approximating a collection of frequent sets," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 12–19, ACM, 2004.

[24] D. Xin, H. Cheng, X. Yan, and J. Han, "Extracting redundancy-aware top-k patterns," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 444–453, ACM, 2006.

[25] L. Guimei, L. Hongjun, and Y. Jeffrewxu, "CFP-tree: A compact disk-based structure for storing and querying frequent itemsets," *Information Systems*, vol. 32, no. 2, pp. 295–319, 2005.

[26] E. Galbrun and P. Miettinen, "Siren: An interactive tool for mining and visualizing geospatial redescriptions," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 1544–1547, ACM, 2012.

[27] E. Galbrun and P. Miettinen, "Interactive redescription mining," in *Proceedings of the International Conference on Management of Data*, pp. 1079–1082, ACM, 2014.

[28] V. Dzyuba, M. Van Leeuwen, S. Nijssen, and L. De Raedt, "Active preference learning for ranking patterns," in *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 532–539, IEEE, 2013.

[29] M. van Leeuwen, "Interactive data exploration using pattern mining," in *Interactive knowledge discovery and data mining in biomedical informatics*, vol. 8401, pp. 169–182, 2014.

[30] V. Dzyuba and M. van Leeuwen, "Interactive discovery of interesting subgroup sets," in *Advances in Intelligent Data Analysis XII*, vol. 8207, pp. 150–161, 2013.

[31] B. Omidvar-Tehrani, S. Amer-Yahia, and A. Termier, "Interactive user group analysis," in *Proceedings of the International Conference on Information and Knowledge Management*, pp. 403–412, ACM, 2015.

[32] C. Bucilă, J. Gehrke, D. Kifer, and W. White, "DualMiner: A dual-pruning algorithm for itemsets with constraints," *Data Mining and Knowledge Discovery*, vol. 7, no. 3, pp. 241–272, 2003.

[33] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, "ExAnte: Anticipated data reduction in constrained pattern mining," in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 59–70, Springer, 2003.

[34] B. Goethals, S. Moens, and J. Vreeken, "MIME: A framework for interactive visual pattern mining," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 757–760, ACM, 2011.

[35] M. Mampaey, N. Tatti, and J. Vreeken, "Tell me what I need to know: Succinctly summarizing data with itemsets," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 573–581, ACM, 2011.

[36] T. De Bie, "Maximum entropy models and subjective interestingness: An application to tiles in binary databases," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 407–446, 2011.

[37] K. McGarry, "A survey of interestingness measures for knowledge discovery," *The Knowledge Engineering Review*, vol. 20, no. 01, pp. 39–61, 2005.

[38] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *SIGMOD Record*, vol. 26, pp. 255–264, ACM, 1997.

[39] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *SIGMOD Record*, vol. 26, pp. 265–276, ACM, 1997.

[40] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53–87.

[41] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine learning*, vol. 42, no. 1-2, pp. 31–60, 2001.

[42] M. J. Zaki, "Efficiently mining frequent trees in a forest: Algorithms and applications," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 8, pp. 1021–1035, 2005.

[43] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proceedings of the International Conference on Data Mining*, pp. 313–320, IEEE, 2001.

[44] X. Yan and J. Han, "GSpan: Graph-based substructure pattern mining," in *Proceedings of the International Conference on Data Mining*, pp. 721–724, IEEE, 2002.

[45] M. A. Hasan, N. Parikh, G. Singh, and N. Sundaresan, "Query suggestion for e-commerce sites," in *Proceedings of the International Conference on Web Search and Data Mining*, pp. 765–774, ACM, 2011.

[46] N. Mishra, R. Saha Roy, N. Ganguly, S. Laxman, and M. Choudhury, "Unsupervised query segmentation using only query logs," in *Proceedings of the International Conference Companion on World Wide Web*, pp. 91–92, ACM, 2011.

[47] S. Jagabathula, N. Mishra, and S. Gollapudi, "Shopping for products you don't know you need," in *Proceedings of the International Conference on Web Search and Data Mining*, pp. 705–714, ACM, 2011.

[48] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," *Information Systems*, vol. 29, no. 4, pp. 343–364, 2004.

[49] T. Calders, "Computational complexity of itemset frequency satisfiability," in *Proceedings of the Twenty-third ACM Symposium on PODS*, pp. 143–154, 2004.

[50] Y. Wang and X. Wu, "Approximate inverse frequent itemset mining: Privacy, complexity, and approximation," in *Proceedings of the International Conference on Data Mining*, pp. 8–pp, IEEE, 2005.

[51] M. Boley and H. Grosskreutz, "Approximating the number of frequent sets in dense data," *Knowledge and Information Systems*, vol. 21, no. 1, pp. 65–89, 2009.

[52] M. Al Hasan and M. J. Zaki, "Output space sampling for graph patterns," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 730–741, 2009.

[53] M. Al Hasan and M. J. Zaki, "MUSK: Uniform sampling of k maximal patterns.," in *Proceedings of the SIAM Data Mining*, pp. 650–661, SIAM, 2009.

[54] M. Boley, C. Lucchese, D. Paurat, and T. Gärtner, "Direct local pattern sampling by efficient two-step random procedures," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 582–590, ACM, 2011.

[55] M. Boley, S. Moens, and T. Gärtner, "Linear space direct pattern sampling using coupling from the past," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 69–77, ACM, 2012.

[56] S. K. M. Wong and V. V. Raghavan, "Vector space model of information retrieval: A reevaluation," in *Proceedings of the International Conference on Research and Development in Information Retrieval*, pp. 167–185, 1984.

[57] S. Raghavan and H. Garcia-Molina, "Crawling the hidden web," in *Proceedings of the Very Large Sacle DataBases*, pp. 129–138, ACM, 2001.

[58] A. Dasgupta, N. Zhang, and G. Das, "Leveraging count information in sampling hidden databases," in *Proceedings of the International Conference on Data Engineering*, pp. 329 –340, IEEE, 2009.

[59] A. Dasgupta, N. Zhang, and G. Das, "Turbo-charging hidden database samplers with overflowing queries and skew reduction," in *Proceedings of the 13th International Conference on Extending Database Technology*, pp. 51–62, ACM, 2010.

[60] N. Bruno, L. Gravano, and A. Marian, "Evaluating top-k queries over web-accessible databases," in *Proceedings of the International Conference on Data Engineering*, pp. 369–380, IEEE, 2002.

[61] A. Dasgupta, X. Jin, B. Jewell, N. Zhang, and G. Das, "Unbiased estimation of size and other aggregates over hidden web databases," in *Proceedings of the International Conference on Management of Data*, pp. 855–866, 2010.

[62] V. Verykios, A. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association rule hiding," *Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 434–447, 2004.

[63] S. R. M. Oliveira and O. R. Zaïane, "Privacy preserving frequent itemset mining," in *Proceedings of the International Conference on Privacy, Security and Data Mining*, pp. 43–54, IEEE, 2002.

[64] E. Bertino, I. N. Fovino, and L. P. Provenza, "A framework for evaluating privacy preserving data mining algorithms," *Data Mining and Knowledge Discovery*, vol. 11, no. 2, pp. 121–154, 2005.

[65] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 217–228, ACM, 2002.

[66] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 639–644, ACM, 2002.

[67] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *Transaction on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1026–1037, 2004.

[68] T. K. Chia, K. C. Sim, H. Li, and H. T. Ng, "A lattice-based approach to query-by-example spoken document retrieval," in *Proceedings of the International Conference on Research and Development in Information Retrieval*, pp. 363–370, ACM, 2008.

[69] F. R. Chung, *Spectral Graph Theory*. American Mathematical Society, 92 ed., 1997.

[70] I. Benjamini, G. Kozma, and N. Wormald, "The mixing time of the giant component of a random graph," *Random Structures & Algorithms*, vol. 45, no. 3, pp. 383–407, 2014.

[71] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1st ed., 1995.

[72] O. Goldreich, *Foundations of Cryptography: Basic Applications*. Cambridge university press, 2nd ed., 2009.

[73] B. Bringmann, A. Zimmermann, L. De Raedt, and S. Nijssen, "Dont be afraid of simpler patterns," in *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 55–66, Springer, 2006.

[74] D. Pavlov, H. Mannila, and P. Smyth, "Beyond independence: Probabilistic models for query approximation on binary transaction data," *IEEE Transaction on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1409–1421, 2003.

[75] B. Everitt, *Finite Mixture Distributions*. Encyclopedia of Statistics in Behavioral Science, 1st ed., 2005.

[76] I. V. Cadez, P. Smyth, and H. Mannila, "Probabilistic modeling of transaction data with applications to profiling, visualization, and prediction," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 37–46, 2001.

[77] V. A. Dobrushkin, *Methods in Algorithmic Analysis*. CRC press, 1st ed., 2016.

[78] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[79] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 801–809, 2011.

[80] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[81] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, ACM, 2014.

[82] F. Liu, B. Liu, C. Sun, M. Liu, and X. Wang, "Deep learning approaches for link prediction in social network services," in *Neural Information Processing*, vol. 8227, pp. 425–432, Springer Berlin, Heidelberg, 2013.

[83] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, ACM, 2016.

[84] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, ACM, 2016.

[85] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis.," in *Proceedings of the International Conference on Document Analysis and Recognition*, vol. 3, pp. 958–962, 2003.

[86] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 3642–3649, IEEE, 2012.

[87] C. Zhou, B. Cule, and B. Goethals, "Itemset based sequence classification," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 353–368, Springer, 2013.

[88] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu., and V. S. Tseng, "SPMF: A Java open source pattern mining library," *Machine Learning Research*, vol. 15, pp. 3389–3393, 2014.

[89] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining closed sequential patterns in large datasets," in *Proceedings of the International Conference SIAM Data Mining*, pp. 166–177, SIAM, 2003.

[90] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the Workshop on New Challenges for NLP Frameworks*, pp. 45–50, 2010.

[91] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1289–1296, 2008.

[92] "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 0, pp. 293–306, 1985.

[93] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proceedings of the International Conference on Very Large Data Bases*, vol. 1215, pp. 487–499, 1994.

[94] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, pp. 1–11, 2004.

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

[96] T. N. A. of Realtors, "2014 profile of international home buying activity," *National Association of Realtors*, 2015.

[97] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[98] J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of KDD Cup and Workshop*, p. 35, ACM, 2007.

[99] "The new way to search for real estate." `http://www.inman.com/2014/12/10/`, 2014.

[100] K. Gouda and M. Zaki, "Efficiently mining maximal frequent itemsets," in *Proceedings of the International Conference on Data Mining*, pp. 163–170, IEEE, 2001.

[101] "Django (version 1.5) [computer software]." `https://djangoproject.com`, 2013.

[102] "Bootstrap (version 3.3.6) [HTML, CSS, and JS framework]." `http://getbootstrap.com/`, 2011.

[103] "jQuery (version 1.x) [JS framework]." `https://api.jquery.com/`, 2012.

[104] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining*, pp. 1–20, 2010.

[105] T. N. A. of Realtors, "2015 home buyer and seller generational trends," *National Association of Realtors*, 2015.

[106] J. Krainer, C. Wei, *et al.*, "House prices and fundamental value," *Federal Reserve Bank of San Francisco Economic Letter*, 2004.

[107] C. H. Nagaraja, L. D. Brown, and L. H. Zhao, "An autoregressive approach to house price modeling," *The Annals of Applied Statistics*, vol. 5, no. 1, pp. 124–149, 2011.

[108] M.-L. Downie and G. Robson, *Automated Valuation Models: An International Perspective*. Royal Institution of Chartered Surveyors, 1st ed., 2008.

[109] P. A. Champ, K. J. Boyle, and T. C. Brown, *A Primer on Nonmarket Valuation*.

[110] Y. Fu, H. Xiong, Y. Ge, Z. Yao, Y. Zheng, and Z.-H. Zhou, "Exploiting geographic dependencies for real estate appraisal: A mutual perspective of ranking and clustering," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 1047–1056, ACM, 2014.

[111] Y. Fu, G. Liu, S. Papadimitriou, H. Xiong, Y. Ge, H. Zhu, and C. Zhu, "Real estate ranking via mixed land-use latent models," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 299–308, ACM, 2015.

[112] Y. Fu, Y. Ge, Y. Zheng, Z. Yao, Y. Liu, H. Xiong, and N. J. Yuan, "Sparse real estate ranking with online user reviews and offline moving behaviors," in *Proceedings of the International Conference on Data Mining*, pp. 120–129, IEEE, 2014.

[113] E. M. Daly, A. Botea, A. Kishimoto, and R. Marinescu, "Multi-criteria journey aware housing recommender system," in *Proceedings of the Conference on Recommender systems*, pp. 325–328, ACM, 2014.

[114] X. Yuan, J.-H. Lee, S.-J. Kim, and Y.-H. Kim, "Toward a user-oriented recommendation system for real estate websites," *Information Systems*, vol. 38, no. 2, pp. 231–243, 2013.

VITA

VITA

Md Mansurul Alam Bhuiyan
Indiana University Purdue University Indianapolis
Email: mbhuiyan@iupui.edu
Alternate Email: rabbi_buet@yahoo.com
Phone: (+1) 765-237-1961

Education:

PhD, Computer Science (Specialization in Data Mining), 2011-2016
Indiana University Purdue University Indianapolis (IUPUI),
Indianapolis, Indiana, USA

B.Sc, Computer Science and Engineering, 2004 - 2008
Bangladesh University of Engineering and Technology (BUET),
Dhaka, Bangladesh

Research Interest:

Data Mining, Big Data, Large Network Analysis, Applied Data Science

Experience:

i) *Research Assistant*, IUPUI, Indianapolis, IN, USA, 2011 - 2016
ii) *Data Science Intern*, Walmart Labs, San Bruno, CA, USA, May 2015 - August 2015
iii) *Summer Research Intern*, Qatar Computing Research Institute, Doha, Qatar, May 2014 - August 2014
iv) *Summer Research Intern*, Verisign Labs, Reston, VA, USA, May 2013 - August 2013