Learning what matters –
Sampling interesting patterns

Vladimir Dzyuba[1] and Matthijs van Leeuwen[2]

[1] Department of Computer Science, KU Leuven, Belgium
[2] LIACS, Leiden University, The Netherlands
vladimir.dzyuba@cs.kuleuven.be, m.van.leeuwen@liacs.leidenuniv.nl

Abstract. In the field of *exploratory data mining*, local structure in data can be described by *patterns* and discovered by mining algorithms. Although many solutions have been proposed to address the redundancy problems in pattern mining, most of them either provide succinct pattern sets or take the interests of the user into account—but not both. Consequently, the analyst has to invest substantial effort in identifying those patterns that are relevant to her specific interests and goals.

To address this problem, we propose a novel approach that combines pattern sampling with interactive data mining. In particular, we introduce the LETSIP algorithm, which builds upon recent advances in 1) weighted sampling in SAT and 2) learning to rank in interactive pattern mining. Specifically, it *exploits user feedback to directly learn the parameters of the sampling distribution that represents the user's interests.*

We compare the performance of the proposed algorithm to the state-of-the-art in interactive pattern mining by emulating the interests of a user. The resulting system allows efficient and interleaved learning and sampling, thus user-specific anytime data exploration. Finally, LETSIP demonstrates favourable trade-offs concerning both quality–diversity and exploitation–exploration when compared to existing methods.

1 Introduction

Imagine a data analyst who has access to a medical database containing information about patients, diagnoses, and treatments. Her goal is to identify novel connections between patient characteristics and treatment effects. For example, one treatment may be more effective than another for patients of a certain age and occupation, even though the latter is more effective at large. Here, age and occupation are latent factors that explain the difference in treatment effect.

In the field of *exploratory data mining*, such hypotheses are represented by *patterns* [1] and discovered by mining algorithms. Informally, a pattern is a statement in a formal language that concisely describes the structure of a subset of the data. Unfortunately, in any realistic database the *interesting* and/or *relevant* patterns tend to get lost among a humongous number of patterns.

The solutions that have been proposed to address this so-called *pattern explosion*, caused by enumerating *all* patterns satisfying given constraints, can be

roughly clustered into four categories: 1) *condensed representations* [9], 2) *pattern set mining* [8], 3) *pattern sampling* [5], 4) and—most recently—*interactive pattern mining* [18]. As expected, each of these categories has its own strengths and weaknesses and there is no ultimate solution as of yet.

That is, condensed representations, e.g., closed itemsets, can be lossless but usually still yield large result sets; pattern set mining and pattern sampling can provide succinct pattern sets but do not take the analyst into account; and existing interactive approaches take the user into account but do not adequately address the pattern explosion. Consequently, the analyst has to invest substantial effort in identifying those patterns that are relevant to her specific interests and goals, which often requires extensive data mining expertise.

Aims and contributions Our overarching aim is to enable analysts—such as the one described in the medical scenario above—to discover small sets of patterns from data that they consider interesting. This translates to the following three specific requirements. First, we require our approach to yield *concise and diverse result sets*, effectively avoiding the pattern explosion. Second, our method should *take the user's interests into account* and ensure that the results are relevant. Third, it should achieve this *with limited effort on behalf of the user*.

To satisfy these requirements, we propose an approach that combines pattern sampling with interactive data mining techniques. In particular, we introduce the LETSIP algorithm, for **Le**arn **t**o **S**ample **I**nteresting **P**atterns, which follows the *Mine, Interact, Learn, Repeat* framework [12]. It samples a small set of patterns, receives feedback from the user, exploits the feedback to learn new parameters for the sampling distribution, and repeats these steps. As a result, the user may utilize a compact diverse set of interesting patterns at any moment, blurring the boundaries between learning and discovery modes.

We satisfy the first requirement by using a sampling technique that samples high quality patterns with high probability. While sampling does not guarantee diversity *per se*, we demonstrate that it gives concise yet diverse results in practice. Moreover, sampling has the advantage that it is *anytime*, i.e., the result set can grow by user's request. LETSIP's sampling component is based on recent advances in sampling in SAT [11] and their extension to pattern sampling [14].

The second requirement is satisfied by learning what matters to the user, i.e., by interactively learning the distribution patterns are sampled from. This allows the user to steer the sampler towards subjectively interesting regions. We build upon recent work [12,6] that uses *preference learning* to learn to rank patterns.

Although user effort can partially be quantified by the total amount of input that needs to be given during the analysis, the third requirement also concerns the time that is needed to find the first interesting results. For this it is of particular interest to study the *trade-off between exploitation and exploration*. As mentioned, one of the benefits of interactive pattern sampling is that the boundaries between learning and discovery are blurred, meaning that the system keeps learning while it continuously aims to discover potentially interesting patterns.

We evaluate the performance of the proposed algorithm and compare it to the state-of-the-art in interactive pattern mining by emulating the interests of a user.

The results confirm that the proposed algorithm has the capacity to learn what matters based on little feedback from the user. More importantly, the LETSIP algorithm demonstrates favourable trade-offs concerning both quality–diversity and exploitation–exploration when compared to existing methods.

2 Interactive pattern mining: Problem definition

Recall the medical analyst example. We assume that after inspecting patterns, she can judge their interestingness, e.g., by comparing two patterns. Then the primary task of interactive pattern mining consists in learning a formal model of her interests. The second task involves using this model to mine novel patterns that are subjectively interesting to the user (according to the learned model).

Formally, let $\mathcal{D}$ denote a dataset, $\mathcal{L}$ a pattern language, $\mathcal{C}$ a (possibly empty) set of constraints on patterns, and $\succ$ the unknown subjective pattern preference relation of the current user over $\mathcal{L}$, i.e., $p_1 \succ p_2$ implies that the user considers pattern p_1 subjectively more interesting than pattern p_2:

Problem 1 (Learning). Given $\mathcal{D}$, $\mathcal{L}$, and $\mathcal{C}$, dynamically collect feedback U with respect to patterns in $\mathcal{L}$ and use U to learn a (subjective) pattern interestingness function $h : \mathcal{L} \to \mathbb{R}$ such that $h(p_1) > h(p_1) \Leftrightarrow p_1 \succ p_2$.

The mining task should account for the potential diversity of user's interests. For example, the analyst may (unwittingly) be interested in several unrelated treatments with disparate latent factors. An algorithm should be able to identify and mine patterns that are representative of these diverse hypotheses.

Problem 2 (Mining). Given $\mathcal{D}$, $\mathcal{L}$, $\mathcal{C}$, and h, mine a set of patterns $\mathcal{P}_h$ that maximizes a combination of interestingness h and diversity of patterns.

The interestingness of $\mathcal{P}$ can be quantified by the average quality of its members, i.e., $\sum_{p \in \mathcal{P}} h(p) \mid /|\mathcal{P}|$. Diversity measures quantify how different patterns in a set are from each other. *Joint entropy* is a common diversity measure [21].

3 Related work

In this paper, we focus on two classes of related work aimed at alleviating the pattern explosion, namely 1) pattern sampling and 2) interactive pattern mining.

Pattern sampling. First pattern samplers are based on Markov Chain Monte Carlo (MCMC) random walks over the pattern lattice [5,16,4]. Their main advantage is that they support "black box" distributions, i.e., they do not require any prior knowledge about the target distribution, a property essential for interactive exploration. However, they often converge only slowly to the desired target distribution and require the selection of the "right" proposal distributions.

Samplers that are based on alternative approaches include direct two-step samplers and XOR samplers. Two-step samplers [7], while provably accurate and efficient, only support a limited number of distributions and thus cannot be easily

extended to interactive settings. FLEXICS [14] is a recently proposed pattern sampler based on the latest advances in weighted constrained sampling in SAT [11]. It supports black-box target distributions, provides guarantees with respect to sampling accuracy and efficiency, and has been shown to be competitive with the state-of-the-art methods described above.

Interactive pattern mining. Most recent approaches to interactive pattern mining are based on learning to rank patterns. They first appeared in Xin et al. [22] and Rueping [19] and were independently extended by Boley et al. [6] and Dzyuba et al. [12]. The central idea behind these algorithms is to alternate between mining and learning. PRIIME [3] focuses on advanced feature construction for interactive mining of structured data, e.g., sequences or graphs.

To the best of our knowledge, IPM [2] is the only existing approach to interactive itemset sampling. It uses binary feedback ("likes" and "dislikes") to update weights of individual items. Itemsets are sampled proportional to the product of weights of constituent items. Thus, the model of user interests in IPM is fairly restricted; moreover, it potentially suffers from convergence issues typical for MCMC. We empirically compare LETSIP with IPM in Section 6.

4 Preliminaries

Pattern mining and sampling. We focus on *itemset mining*, i.e., pattern mining for binary data. Let $\mathcal{I} = \{1 \ldots M\}$ denote a set of items. Then, a dataset $\mathcal{D}$ is a bag of transactions over $\mathcal{I}$, where each transaction t is a subset of $\mathcal{I}$, i.e., $t \subseteq \mathcal{I}$; $\mathcal{T} = \{1 \ldots N\}$ is a set of transaction indices. The pattern language $\mathcal{L}$ also consists of sets of items, i.e., $\mathcal{L} = 2^{\mathcal{I}}$. An itemset p occurs in a transaction t, iff $p \subseteq t$. The frequency of p is the proportion of transactions in which it occurs, i.e., $freq\,(p) = |\{t \in \mathcal{D} \mid p \subseteq t\}|/N$. In labeled datasets, each transaction t has a label from $\{-, +\}$; $freq^{-,+}$ are defined accordingly.

The choice of constraints and a quality measure allows a user to express her analysis requirements. The most common constraint is *minimal frequency* $freq\,(p) \geq \theta$. In contrast to hard constraints, quality measures are used to describe soft preferences that allow to rank patterns; see Section 6 for examples.

While common mining algorithms return the top-k patterns w.r.t. a measure $\varphi : \mathcal{L} \to \mathbb{R}^{+}$, pattern sampling is a randomized procedure that 'mines' a pattern with probability proportional to its quality, i.e., $P_{\varphi}\,(p \text{ is sampled}) = \varphi\,(p)/Z_{\varphi}$, if $p \in \mathcal{L}$ satisfies $\mathcal{C}$, and 0 otherwise, where Z_{φ} is the (unknown) normalization constant. This is an instance of weighted constrained sampling.

Weighted constrained sampling. This problem has been extensively studied in the context of sampling solutions of a SAT problem. WEIGHTGEN [11] is a recent algorithm for approximate weighted sampling in SAT. The core idea consists of partitioning the solution space into a number of "cells" and sampling a solution from a random cell. Partitioning with desired properties is obtained via augmenting the SAT problem with uniformly random XOR constraints (XORs).

To sample a solution, WEIGHTGEN dynamically estimates the number of XORs required to obtain a suitable cell, generates random XORs, stores the

Algorithm 1 LᴇᴛSIP

Input: Dataset $\mathcal{D}$, minimal frequency threshold θ
Parameters: Query size k, query retention l, range A, cell sampling strategy ς
    SCD: regularisation parameter λ, iterations T; Fʟᴇxɪᴄs: error tolerance κ
                                                                              ▷ *Initialization*
 1: Ranking function $h_0 = \text{Lᴏɢɪsᴛɪᴄ}(\mathbf{0}, A)$ ▷ Zero weights lead to uniform sampling
 2: Feedback $U \leftarrow \emptyset$, $Q_0^* \leftarrow \emptyset$
                                                        ▷ *Mine, Interact, Learn, Repeat* loop
 3: **for** $t = 1, 2, \ldots$ **do**
 4:     $R = \text{TᴀᴋᴇFɪʀsᴛ}(Q_{t-1}^*, l)$ ▷ Retain top patterns from the previous iteration
 5:     Query $Q_t \leftarrow R \cup \text{SᴀᴍᴘʟᴇPᴀᴛᴛᴇʀɴs}(h_{t-1}) \times (k - |R|)$ times
 6:     $Q_t^* = \text{Oʀᴅᴇʀ}(Q_t)$, $U \leftarrow U \cup Q_t^*$ ▷ Ask user to order patterns in Q_t
 7:     $h_t \leftarrow \text{Lᴏɢɪsᴛɪᴄ}(\text{LᴇᴀʀɴWᴇɪɢʜᴛs}(U; \lambda, T), A)$
 8: **function** SᴀᴍᴘʟᴇPᴀᴛᴛᴇʀɴs(Sampling weight function $w : \mathcal{L} \to [A, \ 1]$)
 9:     $C = \text{FʟᴇxɪᴄsRᴀɴᴅᴏᴍCᴇʟʟ}(\mathcal{D}, freq\,(\cdot) \geq \theta, w; \kappa)$
10:     **if** $\varsigma = \text{Tᴏᴘ}(m)$ **then return** m highest-weighted patterns
11:     **else if** $\varsigma = \text{Rᴀɴᴅᴏᴍ}$ **then return** PᴇʀꜰᴇᴄᴛSᴀᴍᴘʟᴇ(C, w)

---

solutions of the augmented problem (i.e., a random cell), and returns a perfect weighted sample from the cell. Owing to the properties of partitioning with uniformly random XORs, WᴇɪɢʜᴛGᴇɴ provides theoretical performance guarantees regarding quality of samples and efficiency of the sampling procedure.

For implementation purposes, WᴇɪɢʜᴛGᴇɴ only requires an efficient oracle that enumerates solutions. Moreover, it treats the target sampling distribution as a black box: it requires neither a compact description thereof, nor the knowledge of the normalization constant. Both features are crucial in pattern sampling settings. Fʟᴇxɪᴄs [14], a recently proposed pattern sampler based on WᴇɪɢʜᴛGᴇɴ, has been shown to be accurate and efficient. Due to the page limit, we postpone further details to the extended version of this paper [13, Appendix A].

Preference learning. The problem of learning ranking functions is known as *object ranking*. A common solving technique involves minimizing pairwise loss, e.g., the number of discordant pairs. For example, user feedback $U = \{p_1 \succ p_3 \succ p_2, \ p_4 \succ p_2\}$ is seen as $\{(p_1 \succ p_3), (p_1 \succ p_2), (p_3 \succ p_2), (p_4 \succ p_2)\}$. Given feature representations of objects $\boldsymbol{p_i}$, object ranking is equivalent to positive-only classification of difference vectors, i.e., a ranked pair example $p_i \succ p_j$ corresponds to a classification example $(\boldsymbol{p_i} - \boldsymbol{p_j}, +)$. All pairs comprise a training dataset for a scoring classifier. Then, the predicted ranking of any set of objects can be obtained by sorting these objects by classifier score descending. For example, this formulation is adopted by SᴠᴍRᴀɴᴋ [17].

5   Algorithm

Key questions concerning instantiations of the *Mine, interact, learn, repeat* framework include 1) the feedback format, 2) learning quality measures from feedback, 3) mining with learned measures, and crucially, 4) selecting the patterns to show

to the user. As pattern sampling has been shown to be effective in mining and learning, we present LetSIP, a sampling-based instantiation of the framework which employs Flexics. The sequel describes the mining and learning components of LetSIP. Algorithm 1 shows its pseudocode.

Mining patterns by sampling. Recall that the main goal is to discover patterns that are subjectively interesting to a particular user. We use parameterised logistic functions to measure the interestingness/quality of a given pattern p:

$$\varphi_{\text{logistic}}(p; w, A) = A + \frac{1 - A}{1 + e^{-\boldsymbol{w} \cdot \boldsymbol{p}}}$$

where $\boldsymbol{p}$ is the vector of pattern features for p, $\boldsymbol{w}$ are feature weights, and A is a parameter that controls the range of the interestingness measure, i.e. $\varphi_{\text{logistic}} \in (A, 1)$. Examples of pattern features include $Length(p) = |p|/|\mathcal{I}|$, $Frequency(p) = freq(p)/|\mathcal{D}|$, $Items(i, p) = [i \in p]$; and $Transactions(t, p) = [p \subseteq t]$, where $[\cdot]$ denotes the Iverson bracket. Weights reflect feature contributions to pattern interestingness, e.g., a user might be interested in combinations of particular items or disinterested in particular transactions. The set of features would typically be chosen by the mining system designer rather than by the user herself. We empirically evaluate several feature combinations in Section 6.

Specifying feature weights manually is tedious and opaque, if at all possible. Below we present an algorithm that learns the weights based on easy-to-provide feedback with respect to patterns. This motivates our choice of logistic functions: they enable efficient learning. Furthermore, their bounded range $[A, 1]$ yields distributions that allow efficient sampling directly proportional to $\varphi_{\text{logistic}}$ with Flexics. Parameter A essentially controls the *tilt* of the distribution [14].

User interaction & learning from feedback. Following previous research [12], we use *ordered feedback*, where a user is asked to provide a total order over a (small) number of patterns according to their subjective interestingness; see Figure 1 for an example. We assume that there exists an unknown, user-specific target ranking R^*, i.e., a total order over $\mathcal{L}$. The inductive bias is that there exists $\boldsymbol{w}^*$ such that $p \succ q \Rightarrow \varphi_{\text{logistic}}(p, \boldsymbol{w}^*) > \varphi_{\text{logistic}}(q, \boldsymbol{w}^*)$. We apply the reduction of object ranking to binary classification of difference vectors (see Section 4). Following Boley et al. [6], we use Stochastic Coordinate Descent (SCD) [20] for minimizing L1-regularized logistic loss. However, unlike Boley et al., we directly use the learned functions for sampling.

SCD is an anytime convex optimization algorithm, which makes it suitable for the interactive setting. Its runtime scales linearly with the number of training pairs and the dimensionality of feature vectors. It has two parameters: 1) the number of weight updates (per iteration of LetSIP) T and 2) the regularization parameter λ. However, direct learning of $\varphi_{\text{logistic}}$ is infeasible, as it results in a non-convex loss function. We therefore use SCD to optimize the standard logistic loss, which is convex, and use the learned weights $\boldsymbol{w}$ in $\varphi_{\text{logistic}}$.

Selecting patterns to show to the user. An interactive system seeks to ensure faster learning of accurate models by targeted selection of patterns to

show to the user; this is known as *active learning* or *query selection*. Randomized methods have been successfully applied to this task [12]. Furthermore, in large pattern spaces the probability that two redundant patterns are sampled in one (small) batch is typically low. Therefore, a sampler, which produces independent samples, typically ensures diversity within batches and thus sufficient *exploration*. We directly show k patterns sampled by FLEXICS proportional to $\varphi_{\text{logistic}}$ to the user, for which she has to provide a total order as feedback.

We propose two modifications to FLEXICS, which aim at emphasising *exploitation*, i.e., biasing sampling towards higher-quality patterns. First, we employ alternative cell sampling strategies. Normally FLEXICS draws a perfect weighted random sample, once it obtains a suitable cell. We denote this strategy as $\varsigma = \text{RANDOM}$. We propose an alternative strategy $\varsigma = \text{TOP}(m)$, which picks the m highest-quality patterns from a cell (Line 10 in Algorithm 1). We hypothesize that, owing to the properties of random XOR constraints, patterns in a cell as well as in consecutive cells are expected to be sufficiently diverse and thus the modified cell sampling does not disrupt exploration.

Rigorous analysis of (unweighted) uniform sampling by Chakraborty et al. shows that re-using samples from a cell still ensures broad coverage of the solution space, i.e., diversity of samples [10]. Although as a downside, consecutive samples are not i.i.d., the effects are bounded in theory and inconsequential in practice. We use these results to take license to modify the theoretically motivated cell sampling procedure. Although we do not present a similar theoretical analysis of our modifications, we evaluate them empirically.

Second, we propose to retain the top l patterns from the previous query and only sample $k - l$ new patterns (Lines 4–5). This should help users to relate the queries to each other and possibly exploit the structure in the pattern space.

6 Experiments

The experimental evaluation focuses on 1) the accuracy of the learned user models and 2) the effectiveness of learning and sampling. Evaluating interactive algorithms is challenging, for domain experts are scarce and it is hard to gather enough experimental data to draw reliable conclusions. In order to perform extensive evaluation, we emulate users using (hidden) interest models, which the algorithm is supposed to learn from ordered feedback only.

We follow a protocol also used in previous work [12]: we assume that R^* is derived from a quality measure φ, i.e., $p \succ q \Leftrightarrow \varphi(p) > \varphi(q)$. Thus, the task is to learn to sample frequent patterns proportional to φ from (short) sample rankings. As φ, we use frequency $freq$, surprisingness $surp$, and discriminativity in labeled data as measured by χ^2, where $surp(p) = \max\{freq(p) - \prod_{i \in p} freq(\{i\}), 0\}$ and

$$\chi^2(p) = \sum_{c \in \{-,+\}} \frac{(freq(p)(freq^c(p) - |\mathcal{D}^c|))^2}{freq(p)|\mathcal{D}^c|} + \frac{(freq(p)(freq^c(p) - |\mathcal{D}^c|))^2}{(|\mathcal{D}| - freq(p))|\mathcal{D}^c|}$$

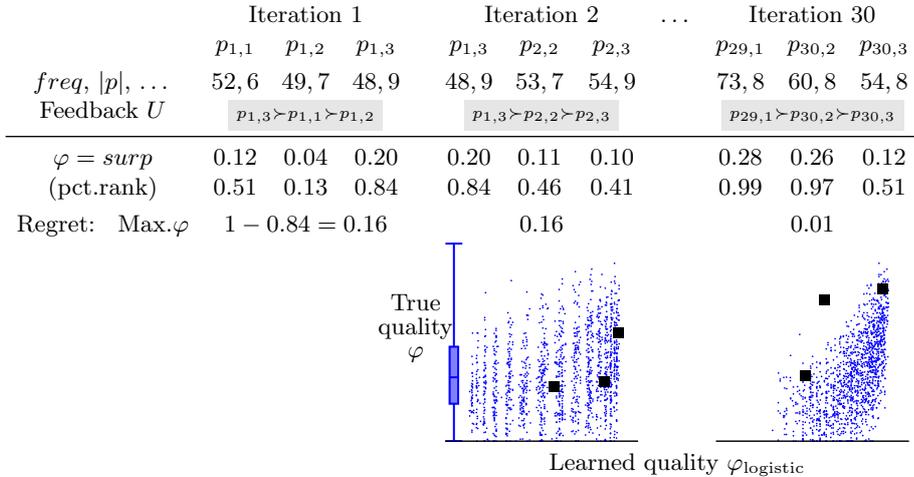|	Iteration 1			Iteration 2			...	Iteration 30		
	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,3}$	$p_{2,2}$	$p_{2,3}$		$p_{29,1}$	$p_{30,2}$	$p_{30,3}$
$freq, |p|, \ldots$	52, 6	49, 7	48, 9	48, 9	53, 7	54, 9		73, 8	60, 8	54, 8
Feedback U	$p_{1,3} \succ p_{1,1} \succ p_{1,2}$			$p_{1,3} \succ p_{2,2} \succ p_{2,3}$				$p_{29,1} \succ p_{30,2} \succ p_{30,3}$		
$\varphi = surp$	0.12	0.04	0.20	0.20	0.11	0.10		0.28	0.26	0.12
(pct.rank)	0.51	0.13	0.84	0.84	0.46	0.41		0.99	0.97	0.51
Regret: Max.φ	$1 - 0.84 = 0.16$			0.16				0.01		

Fig. 1. We emulate user feedback U using a hidden quality measure φ (here *surp*; the boxplot shows the distribution of φ in the given dataset). The rows above the bar show the properties of the sampled patterns that would be inspected by a user, e.g., *frequency* or *length*, and the emulated feedback. The scatter plots show the relation between φ and the learned model of user interests $\varphi_{\text{logistic}}$ after 1 and 29 iterations of feedback and learning. The performance of the learned model improves considerably as evidenced by higher values of φ of the sampled patterns (squares) and lower regret.

We investigate the performance of the algorithm on ten datasets[3]. For each dataset, we set the minimal support threshold such that there are approximately $140\,000$ frequent patterns. Table 1 shows dataset statistics. Each experiment involves 30 iterations (queries). We use the default values suggested by the authors of SCD and Flexics for the auxiliary parameters of LetSIP: $\lambda = 0.001$, $T = 1000$, and $\kappa = 0.9$.

We evaluate performance using *cumulative regret*, which is the difference between the ideal value of a certain

Table 1. Dataset properties.

	$|\mathcal{I}|$	$|\mathcal{D}|$	θ	Frequent patterns
anneal	93	812	660	149 331
australian	125	653	300	141 551
german	112	1000	300	161 858
heart	95	296	115	153 214
hepatitis	68	137	48	148 289
lymph	68	148	48	146 969
primary	31	336	16	162 296
soybean	50	630	28	143 519
vote	48	435	25	142 095
zoo	36	101	10	151 806

tain measure M and its observed value, summed over iterations. We use the maximal and average quality φ in a query and joint entropy H_J as performance measures. To allow comparison across datasets and measures, we use percentile ranks by φ as a non-parametric measure of ranking performance. We also divide joint entropy by k: thus, the ideal value of each measure is 1 (e.g., the highest

---

[3] Source: `https://dtai.cs.kuleuven.be/CP4IM/datasets/`

possible φ over all frequent patterns has the percentile rank of 1), and the regret is defined as $\sum 1 - M(Q_i^*)$, where $M \in \{\varphi_{avg}, \varphi_{max}, H_J\}$. We repeat each experiment ten times with different random seeds and report average regret.

A characteristic experiment in detail. Figure 1 illustrates the workings of LETSIP and the experimental setup. It uses the `lymph` dataset, the target quality measure $\varphi = surp$, features $= Items$, $k = 3$, $A = 0.1$, $l = 1$, $\varsigma = $ RANDOM.

LETSIP starts by sampling patterns uniformly. A human user would inspect the patterns (items not shown) and their properties, e.g., frequency or length, or visualizations thereof, and rank the patterns by their subjective interestingness; in these experiments, we order them according to their values of φ. The algorithm uses the feedback to update $\varphi_{\text{logistic}}$. At the next iteration, the patterns are sampled from an updated distribution. As $l = 1$, the top-ranked pattern from the previous iteration ($p_{1,3}$) is retained. After a number of iterations, the accuracy of the approximation increases considerably, while the regret decreases. On average, one iteration takes 0.5s on a desktop computer.

Evaluating components of LetSIP. We investigate the effects of the choice of features and parameter values on the performance of LETSIP, in particular query size k, query retention l, range A, and cell sampling strategy ς. We use the following feature combinations ($\|$ denotes concatenation): *Items* (I); *Items*$\|$*Length*$\|$*Frequency* (ILF); and *Items*$\|$*Length*$\|$*Frequency*$\|$*Transactions* (ILFT). Values for other parameters and aggregated results are shown in Table 2.

Increasing the query size decreases the maximal quality regret more than twofold, which indicates that the proposed learning technique is able to identify the properties of target measures from ordered lists of patterns. However, as larger queries also increase the user effort, further we use a more reasonable query size of $k = 5$. Similarly, additional features provide valuable information to the learner. Changing the range A does not affect the performance.

The choice of values for query retention l and the cell sampling strategy allows influencing the exploration-exploitation trade-off. Interestingly, retaining one highest-ranked pattern results in the lowest regret with respect to the *maximal* quality. Fully random queries ($l = 0$) do not enable sufficient exploitation, whereas higher retention ($l \geq 2$)—while ensuring higher *average* quality—prevents exploration necessary for learning accurate weights.

The cell sampling strategy is the only parameter that clearly affects joint entropy, with purely random cell sampling yielding the lowest regret. However, it is also results in the highest quality regrets, which negates the gains in diversity. Taking the best pattern according to $\varphi_{\text{logistic}}$ ensures the lowest quality regrets and joint entropy equivalent to other strategies. Based on these findings, we use the following parameters in the remaining experiments: $k = 5$, features $=$ ILFT, $A = 0.5$, $l = 1$, $\varsigma = $ TOP(1).

The largest proportion of LETSIP's runtime costs is associated with sampling (costs of weight learning are low due to a relatively low number of examples). The most important factor is the number of items $|\mathcal{I}|$: the average runtime per iteration ranges from 0.8s for `lymph` to 5.8s for `australian`, which is suitable for online data exploration.

Table 2. Effect of LETSIP's parameters on regret w.r.t. three performance measures. Results are aggregated over datasets, quality measures, and other parameters.

		Regret: avg.φ	Regret: max.φ	Regret: H_J
Query size k	5	6.35 ± 1.04	1.13 ± 0.52	$\mathbf{13.28 \pm 0.89}$
	10	$\mathbf{5.91 \pm 0.59}$	$\mathbf{0.47 \pm 0.18}$	17.44 ± 0.45
All results below are for query size of $k = 5$				
Features	I	8.17 ± 0.96	1.35 ± 0.56	13.64 ± 0.90
	ILF	6.30 ± 1.36	1.16 ± 0.59	13.15 ± 0.96
	ILFT	$\mathbf{4.60 \pm 0.78}$	$\mathbf{0.87 \pm 0.40}$	$\mathbf{13.06 \pm 0.81}$
Range A	0.5	6.43 ± 1.06	1.15 ± 0.52	$\mathbf{13.20 \pm 0.86}$
	0.1	$\mathbf{6.26 \pm 1.01}$	$\mathbf{1.11 \pm 0.51}$	13.36 ± 0.91
Query retention l	0	8.19 ± 1.21	2.53 ± 0.72	13.38 ± 0.69
	1	6.78 ± 0.99	$\mathbf{0.53 \pm 0.34}$	$\mathbf{13.06 \pm 0.72}$
	2	5.61 ± 0.94	0.61 ± 0.42	13.56 ± 1.05
	3	$\mathbf{4.80 \pm 1.00}$	0.80 ± 0.57	13.33 ± 1.22
Cell sampling ς	RANDOM	10.60 ± 0.71	1.89 ± 0.64	$\mathbf{12.15 \pm 0.59}$
	TOP(1)	$\mathbf{5.14 \pm 1.13}$	$\mathbf{0.81 \pm 0.45}$	13.70 ± 1.00
	TOP(2)	5.45 ± 1.06	0.87 ± 0.47	13.60 ± 0.98
	TOP(3)	5.95 ± 1.20	0.95 ± 0.50	13.57 ± 0.96

Comparing with alternatives. We compare LETSIP with APLE [12], another approach based on active preference learning, and IPM [2], an MCMC-based interactive sampling framework. For the former, we use query size k and feature representation identical to LETSIP, query selector MMR($\alpha = 0.3$, $\lambda = 0.7$), $C_{\text{RANKSVM}} = 0.005$, and 1000 frequent patterns sampled uniformly at random and sorted by $freq$ as the *source ranking*. To compute regret, we use the top-5 frequent patterns according to the learned ranking function.

To emulate binary feedback for IPM based on φ, we use a technique similar to the one used by the authors: we designate a number of items as "interesting" and "like" an itemset, if more than half of its items are "interesting". To select the items, we sort frequent patterns by φ descending and add items from the top-ranked patterns until 15% of all patterns are considered "liked".

As we were not able to obtain the code for IPM, we implemented its sampling component by materializing all frequent patterns and generating perfect samples according to the learned multiplicative distribution. Note that this approach favors IPM, as it eliminates the issues of MCMC convergence. We request 300 samples (the amount of training data roughly equivalent to that of LETSIP), partition them into 30 groups of 10 patterns each, and use the tail 5 patterns in each group for regret calculations. Following the authors' recommendations, we set the learning parameter to $b = 1.75$. For the sampling-based methods LETSIP and IPM, we also report the diversity regret as measured by joint entropy.

Table 3. LETSIP has considerably lower regrets than alternatives w.r.t. quality and, for samplers, diversity as quantified by joint entropy. (For $\varphi = surp$ (marked by *), IPM fails for 7 out of 10 datasets due to double overflow of multiplicative weights.)

| | Regret: avg.φ | | | Regret: joint entropy H_J | | |
	freq	χ^2	*surp*	*freq*	χ^2	*surp*
LETSIP	2.4 ± 0.5	$\mathbf{2.4 \pm 0.1}$	$\mathbf{4.5 \pm 1.4}$	$\mathbf{11.7 \pm 0.6}$	$\mathbf{11.7 \pm 0.5}$	$\mathbf{15.9 \pm 1.1}$
IPM	15.5 ± 1.8	12.8 ± 2.3	15.5 ± 1.8*	15.7 ± 1.9	15.4 ± 1.9	19.8 ± 2.1*
APLE	$\mathbf{0.0 \pm 0.0}$	4.5 ± 3.8	5.3 ± 3.9	–	–	–

Table 3 shows the results. The regret of LETSIP is substantially lower than that of either of the alternatives. The advantage over IPM is due to a more powerful learning mechanism and feature representation. IPM's multiplicative weights are biased towards longer itemsets and items seen at early iterations, which may prevent sufficient exploration, as evidenced by higher joint entropy regret. Non-sampling method APLE performs the best for $\varphi = freq$, which can be represented as a linear function of the features and learned by RANKSVM with the linear kernel. It performs substantially worse in other settings and has the highest variance, which reveals the importance of informed source rankings and the cons of pool-based active learning. These results validate the design choices made in LETSIP.

7 Conclusion

We presented LETSIP, a sampling-based instantiation of the *Mine, interact, learn, repeat* interactive pattern mining framework. The user is asked to rank small sets of patterns according to their (subjective) interestingness. The learning component uses this feedback to build a model of user interests via active preference learning. The model directly defines the sampling distribution, which assigns higher probabilities to more interesting patterns. The sampling component uses the recently proposed FLEXICS sampler, which we modify to facilitate control over the exploration-exploitation balance in active learning.

We empirically demonstrate that LETSIP satisfies the key requirements to an interactive mining system. We apply it to itemset mining, using a well-principled method to emulate a user. The results demonstrate that LETSIP learns to sample diverse sets of interesting patterns. Furthermore, it outperforms two state-of-the-art interactive methods. This confirms that it has the capacity to tackle the pattern explosion while taking user interests into account.

Directions for future work include extending LETSIP to other pattern languages, e.g., association rules, investigating the effect of noisy user feedback on the performance, and formal analysis, e.g., with multi-armed bandits [15]. A user study is necessary to evaluate the practical aspects of the proposed approach.

References

1. Aggarwal, C.C., Han, J. (eds.): Frequent Pattern Mining. Springer (2014)
2. Bhuiyan, M., Hasan, M.A.: Interactive knowledge discovery from hidden data through sampling of frequent patterns. Statistical Analysis and Data Mining: The ASA Data Science Journal 9(4), 205–229 (aug 2016)
3. Bhuiyan, M., Hasan, M.A.: PRIIME: A generic framework for interactive personalized interesting pattern discovery. In: Proc. of IEEE Big Data. pp. 606–615 (2016)
4. Boley, M., Gärtner, T., Grosskreutz, H.: Formal concept sampling for counting and threshold-free local pattern mining. In: Proceedings of SDM. pp. 177–188 (2010)
5. Boley, M., Grosskreutz, H.: Approximating the number of frequent sets in dense data. Knowledge and information systems 21(1), 65–89 (2009)
6. Boley, M., Mampaey, M., Kang, B., Tokmakov, P., Wrobel, S.: One Click Mining – interactive local pattern discovery through implicit preference and performance learning. In: Workshop Proceedings of KDD. pp. 28–36 (2013)
7. Boley, M., Moens, S., Gärtner, T.: Linear space direct pattern sampling using coupling from the past. In: Proceedings of KDD. pp. 69–77 (2012)
8. Bringmann, B., Nijssen, S., Tatti, N., Vreeken, J., Zimmermann, A.: Mining sets of patterns. Tutorial at ECML/PKDD (2010)
9. Calders, T., Rigotti, C., Boulicaut, J.F.: A survey on condensed representations for frequent sets. In: Boulicaut, J.F., De Raedt, L., Mannila, H. (eds.) Constraint-Based Mining and Inductive Databases, pp. 64–80. Springer (2006)
10. Chakraborty, S., Fremont, D., Meel, K., Seshia, S., Vardi, M.: On parallel scalable uniform SAT witness generation. In: Proceedings of TACAS. pp. 304–319 (2015)
11. Chakraborty, S., Fremont, D., Meel, K., Vardi, M.: Distribution-aware sampling and weighted model counting for SAT. In: Proc. of AAAI. pp. 1722–1730 (2014)
12. Dzyuba, V., van Leeuwen, M., Nijssen, S., De Raedt, L.: Interactive learning of pattern rankings. International Journal on Artificial Intelligence Tools 23(06) (2014)
13. Dzyuba, V., van Leeuwen, M.: Learning what matters – sampling interesting patterns (mar 2017), `http://arxiv.org/abs/1702.01975`
14. Dzyuba, V., van Leeuwen, M., De Raedt, L.: Flexible constrained sampling with guarantees for pattern mining. Data Mining and Knowledge Discovery (in press), preprint available at `https://arxiv.org/abs/1610.09263`
15. Filippi, S., Cappé, O., Garivier, A., Szepesvári, C.: Parametric bandits: The generalized linear case. In: Proceedings of NIPS. pp. 586–594 (2010)
16. Hasan, M.A., Zaki, M.: Output space sampling for graph patterns. In: Proceedings of VLDB. pp. 730–741 (2009)
17. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of KDD. pp. 133–142 (2002)
18. van Leeuwen, M.: Interactive data exploration using pattern mining. In: Holzinger, A., Jurisica, I. (eds.) Interactive Knowledge Discovery and Data Mining in Biomedical Informatics, pp. 169–182. Springer (2014)
19. Rueping, S.: Ranking interesting subgroups. In: Proc. of ICML. pp. 913–920 (2009)
20. Shalev-Shwartz, S., Tewari, A.: Stochastic methods for ℓ_1-regularized loss minimization. Journal of Machine Learning Research pp. 1865–1892
21. van Leeuwen, M., Ukkonen, A.: Discovering skylines of subgroup sets. In: Proceedings of ECML/PKDD. pp. 273–287 (2013)
22. Xin, D., Shen, X., Mei, Q., Han, J.: Discovering interesting patterns through users interactive feedback. In: Proceedings of KDD. pp. 773–778 (2006)