Interactive Pattern Mining using Discriminant Sub-patterns as Dynamic Features

Arnold Hien, Samir Loudni, Noureddine Aribi, Abdelkader Ouali, Albrecht Zimmermann

Interactive Pattern Mining using Discriminant Sub-patterns as Dynamic Features

Arnold Hien[1,2], Samir Loudni[2], Noureddine Aribi[3], Abdelkader Ouali[1], and Albrecht Zimmermann[1]

[1] Normandie Univ., UNICAEN, CNRS – UMR GREYC, France
{abdelkader.ouali,albrecht.zimmermann}@unicaen.fr
[2] TASC – DAPI, IMT-Atlantique, LS2N – CNRS, 4 rue Alfred Kastler, 44307 Nantes, France
{samir.loudni,arnold.hien}@imt-atlantique.fr
[3] Lab. LITIO, University of Oran1, 31000 Oran, Algeria
aribi.noureddine@univ-oran1.dz

Abstract. Recent years have seen a shift from a pattern mining process that has users define constraints before-hand, and sift through the results afterwards, to an interactive one. This new framework depends on exploiting user feedback to learn a quality function for patterns. Existing approaches have a weakness in that they use static pre-defined low-level features, and attempt to learn independent weights representing their importance to the user. As an alternative, we propose to work with more complex features that are derived directly from the pattern ranking imposed by the user. Those features are used to learn weights to be aggregated with low-level features and help to drive the quality function in the right direction. Experiments on UCI datasets show that using higher-complexity features leads to the selection of patterns that are better aligned with a hidden quality function while being competitively fast when compared to state-of-the-art methods.

1 Introduction

Constraint-based pattern mining is a fundamental data mining task, extracting locally interesting patterns to be either interpreted directly by domain experts, or to be used as descriptors in downstream tasks, such as classification or clustering. Since the publication of the seminal paper [1], two problems have limited the usability of this approach: 1) how to translate user preferences and background knowledge into constraints, and 2) how to deal with the large result sets that often number in the thousands or even millions of patterns. Replacing the original support-confidence framework with other quality measures [14] does not address the pattern explosion. Post-processing results via condensed representations still typically leaves many patterns, while pattern set mining [11] just pushes the problem further down the line.

In recent years, research on *interactive pattern mining* has proposed to alter the mining process itself: instead of specifying constraints once, mining a result set, and then post-processing it, interactive pattern mining performs an iterative loop [12]. This loop involves three repeating main steps: (1) pattern extraction in which a relatively small set of patterns is extracted; (2) interaction in which the user expresses his preferences w.r.t.

those patterns; (3) preference learning in which the expressed preferences are translated into a quality assessment function for mining patterns in future iterations.

The most recent proposal to dealing with the question of finding interesting patterns involves the user, via interactive pattern mining [12] often involving sampling [3], with LETSIP [4] one of the end points of this development. Other interactive methods have been proposed, APLE [5], another approach based on active preference learning to learn a linear ranking function using RANKSVM [8], and IPM [3], an MCMC-based interactive sampling framework. However, existing approaches have a short-coming: to enable preference learning, they represent patterns by independent descriptors, such as included items or covered transactions, and expect the learned function, usually a regression or multiplicative weight model, to handle relations.

In this paper, we propose a new interactive pattern mining approach that introduces more complex class of descriptors for *explainable ranking*, thereby allowing to capture the importance of item interactions. These descriptors exploit the concept of **discriminating sub-patterns**, which separate patterns that are given low rank by the user from those with high rank. By temporarily adding those descriptors, we can learn weights for them, which are then apportioned to involved items without blowing up the feature space. Results on UCI datasets show favourable trade-offs in quality–time of learning.

2 Preliminaries

Pattern Mining. Given a database $\mathcal{D}$, a language $\mathcal{L}$ defining subsets of the data and a selection predicate q that determines whether an element $\phi \in \mathcal{L}$, the task is to find the theory $\mathcal{T}h(\mathcal{L}, \mathcal{D}, q) = \{\phi \in \mathcal{L} \mid \ q(\mathcal{D}, \phi)\ is\ true\}$. A well-known pattern mining task is *frequent itemset mining* [1]. Let $\mathcal{I}$ be a set of n *items*, an *itemset* (or pattern) X is a non-empty subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L}_\mathcal{I} = 2^\mathcal{I} \backslash \emptyset$. A transactional dataset $\mathcal{D}$ is a bag (or multiset) of transactions over $\mathcal{I}$, where each *transaction* t is a subset of $\mathcal{I}$, i.e., $t \subseteq \mathcal{I}$; $\mathcal{T} = \{1, ..., m\}$ a set of m *transaction* indices. An itemset X *occurs* in a transaction t, iff $X \subseteq t$. The *cover* of X in $\mathcal{D}$ is the bag of transactions in which it occurs: $\mathcal{V}_\mathcal{D}(X) = \{t \in \mathcal{D} \mid X \subseteq t\}$. The *support* of X in $\mathcal{D}$ is the size of its cover: $sup_\mathcal{D}(X) = |\mathcal{V}_\mathcal{D}(X)|$.

Learning from Preferences. An algorithmic template of the *Mine*, *Interact*, *Learn*, *Repeat* framework is listed in Algorithm 1. The interactive process proceeds iteratively for some reasonable number of iterations T, which depends on the task at hand. Let $\Phi : \mathcal{L}_\mathcal{I} \rightarrow \mathbb{R}$ denote the true, unobserved preferences function of the user. The algorithm maintains an internal estimate φ^t of the true function, where $t \in [T]$ is the iteration index. At each iteration, it selects a query $\mathcal{X}^t$ to be posed to the user. The user's feedback is then used (possibly along with all the feedback received so far) to compute a new estimate φ^{t+1} of Φ. Key questions concerning instantiations of the Mine, interact, learn, repeat framework include 1) feature representations of patterns to be ranked and the feedback format, 2) learning user's preferences from feedback, 3) mining with learned preferences, and crucially, 4) selecting the patterns to show to the user.

a) User Interaction & Pattern Representations. User feedback w.r.t. patterns takes the form of providing a total order over a (small) set of patterns [4,12], called a query. User feedback $\{X_1 \succ X_3 \succ X_2\}$, for instance, indicates that the user prefers X_1 over

Algorithm 1: Template algorithm for interactive pattern mining

In: Dataset $\mathcal{D}$, set of patterns $\mathcal{X}$
Parameters: Query size k, number of steps T, feature pattern representations $\mathcal{F}$
Out: φ : Ranking function
begin
  $\mathcal{U} \leftarrow \emptyset$, $\varphi^0 \leftarrow$ initial function estimates
  for $t = 1, 2 \dots T$ **do**
    select a query $\mathcal{X}^t$ based on φ^{t-1} ▷ Mine
    ask query $\mathcal{X}^t$ to the user and get feedback $\widehat{\mathcal{R}}^t$ ▷ Interact
    $\mathcal{U} \leftarrow \mathcal{U} \cup \widehat{\mathcal{R}}^t$, compute φ^t based on φ^{t-1} and $\mathcal{U}$ ▷ Learn φ
  return φ;

X_3, which they prefer over X_2 in turn, and so on. Pattern representations determines how the user characterizes patterns of interest to him. Patterns are commonly represented using a vector of *static features* (also called **descriptors**) $\mathcal{F} = \langle F_1, \dots, F_n \rangle$. The description of a pattern X w.r.t. $\mathcal{F}$ is given by the vector $\mathbf{P} = \langle \mathbf{P_1}, \dots, \mathbf{P_n} \rangle$, where $\mathbf{P_i}$ is the value associated to F_i. Examples of features include numerical descriptors like $Len(X) = |X|/|\mathcal{I}|$, $Freq(X) = sup_{\mathcal{D}}(X)/|\mathcal{D}|$, in which case the corresponding $\mathbf{P_i}$ is truly in $\mathbb{R}$, or binary descriptors $Items(i, X) = [i \in X]$; and $Trans(t_i, X) = [X \subseteq t_i]$, where [.] denotes the Iverson bracket, leading to (partial) feature vectors $\in \{0,1\}^{|\mathcal{I}|}$ and $\in \{0,1\}^{|\mathcal{D}|}$, respectively.

b) Learning from Feedback. Evaluating patterns in terms of quality function is a very natural way of representing preferences. In the object preferences scenario [9], such a function is a mapping $\varphi : \mathcal{X} \rightarrow \mathbb{R}$ that assigns a score $\varphi(X)$ to each pattern X and, thereby, induces an order on $\mathcal{X}$. As in [4], we use a parametrized logistic function to measure the interestingness/quality of a given pattern X: $\varphi_{logistic}(X; w, A) = A + \frac{1-A}{1+e^{-\mathbf{w}_{\mathcal{F}} \cdot \mathbf{P}}}$, where $\mathbf{P}$ the afore-mentioned description of a pattern X, $\mathbf{w}_{\mathcal{F}}$ the weight vector associated to descriptors $\mathcal{F}$ reflecting feature contributions to pattern interestingness, and A is a parameter that controls the range of the interestingness measures, i.e. $\varphi_{logistic} \in (A, 1)$. However, setting feature weights manually is tedious, thus we present in section 3 an algorithm that learns the weights based on easy-to-provide feedback with respect to patterns. Given a user feedback $\mathcal{U} = \{X_1 \succ X_3 \succ X_2\}$ which is translated into pairwise rankings $\{(X_1 \succ X_3), (X_1 \succ X_2), \dots\}$, each ranked pair $X_i \succ X_j$ corresponds to a classification example $(\mathbf{P}_i - \mathbf{P}_j, +)$ of a training dataset. We use Stochastic Coordinate Descent (SCD) [13] for minimizing logistic loss stemming from this training dataset, and use the learned weights in $\varphi_{logistic}$.

3 DiSPaLe: Discriminating Sub-Pattern Feature Learning

We present DISPALE, an instantiation of the framework described by Algorithm 1, which exploits more complex descriptors in combination with static low-level features to learn logistic functions. These new descriptors are learned from discriminating sub-patterns. The sequel describes how discriminating sub-pattern are extracted from the user-defined pattern ranking and how they are used in the learning component of DISPALE (see Algorithm 2). Table 1b summarizes the different notations used in this paper.

Algorithm 2: DISPALE (**Di**scriminating **S**ub-**Pa**ttern feature **Le**arning)

In: Dataset $\mathcal{D}$, set of patterns $\mathcal{X}$
Parameters: Query size k, number of steps T, range A, query retention ℓ, feature pattern representations $\mathcal{F}$
Out: φ : Ranking function;
begin
  $\mathcal{U} \leftarrow \emptyset, w^0_{\mathcal{F}} \leftarrow \mathbf{0}, \mathcal{X}^0 \leftarrow \emptyset, \varphi^0 = \varphi_{logistic}(w^0_{\mathcal{F}}, A)$
  for $t = 1, 2 \ldots T$ **do**
    $\mathcal{X}^t \leftarrow \text{TOP}(\mathcal{X}^{t-1}, \ell) \cup (\text{SAMPLEPATTERNS}(\mathcal{D}, \varphi^{t-1}) \times (k - \ell))$
    $\widehat{\mathcal{R}}^t \leftarrow \text{RANK}(\mathcal{X}^t),\ disc \leftarrow \text{MINEDISCRIMINATING}(\mathcal{X}^t, \widehat{\mathcal{R}}^t),\ \mathcal{U} \leftarrow \mathcal{U} \cup \widehat{\mathcal{R}}^t$
    $\langle w^t_{\mathcal{F}}, w^t_{\mathcal{F}_{disc}} \rangle \leftarrow \text{LEARNWEIGHTS}(\mathcal{U}, \mathcal{F} \cup \mathcal{F}_{disc})$
    $w^t_{\mathcal{F}} \leftarrow \text{UPDATEWEIGHTS}(w^t_{\mathcal{F}}, w^t_{\mathcal{F}_{disc}})$
    $\varphi^t \leftarrow \varphi_{logistic}(w^t_{\mathcal{F}}, A)$
  return φ^T

3.1 Towards More Expressive and Learnable Pattern Descriptions

Features for pattern representation involve indicator variables for included items or subgraphs, or for covered transactions [3,4], or pattern length, etc. The issue is that such features are treated as if they were independent, whether in the logistic function mentioned above, or multiplicative functions [3,12]. While this allows to identify pattern components that are *globally* interesting for the user, it is impossible to learn relationships such as "the user is interested in item i_1 if item i_3 is present but item i_4 is absent". In addition, the pattern elements whose inclusion is indicated are defined before-hand, and the user feedback has no influence on them. We therefore propose to learn more expressive features in order to improve the learning of user preferences. In this work, we propose to consider *discriminating sub-patterns* that better capture (or explain) these preferences. Those features exploit ranking-correlated patterns, *i.e.,* patterns that influence the user ranking either by allowing some patterns to be well ranked or the opposite.
a) Interclass Variance. As explained above, our goal is to mine sub-patterns that discriminate between patterns that have been given a high user ranking and those that received a low one. An intuitive way of modelling this problem consists of considering the numerical ranks given to individual patterns as numerical labels and the mining setting as akin to regression. We are not aiming to build a full regression model but only to mine an individual pattern that correlates with the numerical label. For this purpose, we use the interclass variance measure as proposed by [10].

Definition 1. *Let $\mathcal{X}$ be a query, $\widehat{\mathcal{R}}$ the user ranked patterns, $\mathcal{X}_Y$ the subset of patterns X in $\mathcal{X}$ containing the sub-pattern Y, and $\overline{\mathcal{X}}_Y = \mathcal{X} - \mathcal{X}_Y$. The interclass variance of the sub-pattern y is defined by: $ICV(Y, \widehat{R}) = |\mathcal{X}_Y| \cdot (\mu(\mathcal{X}) - \mu(\mathcal{X}_Y))^2 + |\overline{\mathcal{X}}_Y| \cdot (\mu(\mathcal{X}) - \mu(\overline{\mathcal{X}}_Y))^2$, where $\mu(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \cdot \sum_{X \in \mathcal{X}} r(X)$, and $r(X)$ is the rank of X in $\widehat{\mathcal{R}}$.*

b) Extracting Discriminating Sub-patterns. To find the sub-pattern $Y \subseteq X \in \mathcal{X}$ with the greatest interclass variance ICV, we systematically search the pattern space spanned by the items involved in patterns of the user's query $\mathcal{X}$. Semantically, this is the sub-pattern whose presence in one or more patterns X has influenced their ranking. So, if $Y \subseteq X$, we can say that the ranking of X at the $r(X)^{th}$ position in $\widehat{R}$ is more likely to be explained by the presence of sub-pattern Y.

Algorithm 3: Extracting discriminating sub-patterns

```
1  Function MineDiscriminating(𝒳, R̂)
2      ICV_max ← 0, disc ← ∅
3      I_𝒳 ← {i ∈ X | X ∈ 𝒳}
4      𝒮 ← I_𝒳
5      For each i ∈ I_𝒳 do
6          If ICV(i, R̂) ≥ ICV_max then
7              ICV_max ← ICV(i, R̂), disc ← {i}
8      For each Y ∈ 𝒮 do
9          While (∃i ∈ I_𝒳 ∧ ∃X ∈ 𝒳 st. Y ∪ {i} ⊆ X ∧ i ∉ Y) do
10             If ICV(Y ∪ {i}, R̂) ≥ ICV_max then
11                 ICV_max ← ICV(Y ∪ {i}, R̂), disc ← Y ∪ {i} , 𝒮 ← 𝒮 ∪ disc
12     return disc
```

Algorithm 3 implements the function MINEDISCRIMINATING (see Algorithm 2, line 8), which learns the best discriminating pattern as a descriptor. Its accepts as input the query $\mathcal{X}$ and the ranked patterns $\widehat{R}$ by the user, and returns the sub-itemset with the highest ICV. Its starts by computing the ICV of all items of the patterns in $\mathcal{X}$ (loop 5−7). Then, it iteratively combines the items to form a larger and finer-grained discriminating sub-pattern (loop 8-11). Obviously, before combining sub-itemsets, we should ensure that the resulting sub-pattern belongs to an existing pattern $X \in \mathcal{X}$ (line 9). If such a sub-pattern exists, we update the value of ICV_{max}, we save the best discriminating sub-pattern computed so far and we update with $disc$ the set of sub-itemsets that can be extended for further improvements (lines 10-11). Finally, the best discriminating pattern is returned at line 12.

Example 1. Consider a dataset with items $\mathcal{I} = \{1, \ldots, 7\}$. Let's consider a user query $\mathcal{X} = \{X_1, X_2, X_3, X_4\}$, with $X_1 = \{5, 7\}, X_2 = \{2, 7\}, X_3 = \{1\}, X_4 = \{4\}$ and let $\widehat{R} = \{X_2 \succ X_1 \succ X_3 \succ X_4\}$. For $Y = \{2\}$, we have $\mathcal{X}_Y = \{X_2\}$, $\overline{\mathcal{X}}_Y = \{X_1, X_3, X_4\}$, $\mu(\mathcal{X}_Y) = 1$, $\mu(\overline{\mathcal{X}}_Y) = 3$ and $\mu(\mathcal{X}) = 2.5$. Applying definition 1 gives $ICV(2, \widehat{R}) = 3$. After the first loop of Algorithm 3, $ICV_{max} = 4$ and $disc = \{7\}$.

3.2 Discriminating Sub-patterns as Descriptors

Exploiting discriminating sub-patterns as a new descriptors in DISPALE brings meaningful knowledge to consider during an interactive preference learning. In fact, this sub-pattern correlated with the user's ranking emphasizes the items of interest related to his ranking. Now, we describe how these discriminating patterns can be used in order to improve the learning function $\varphi_{logistic}$ for patterns.

A direct way of exploiting discriminating sub-patterns consists of adding them as new descriptors to the initial features $\mathcal{F}$ during the iterations. However, this will increase the size of $\mathcal{F}$, introduces additional cost and most probably leads to over-fitting and generalization issues of the learning function $\varphi_{logistic}$. Instead, we propose to use the discriminating sub-pattern $disc$ extracted at each iteration as a **temporary descriptor** F_{disc} that can be added to $\mathcal{F}$ in order to learn a weight $w_{F_{disc}}$ (see Algorithm 2, line 9). We propose three types of discriminating descriptors:

- F_{disc_X}: a binary descriptor used to assess the presence/absence of $disc$ in a pattern $X \in \mathcal{X}$; its associated value $\mathbf{P_{disc_X}} = \begin{cases} 1 \text{ if } disc \subseteq X \\ 0 \text{ otherwise} \end{cases}$
- $F_{disc_\mathcal{T}}$: a numerical descriptor representing the frequency of the discriminating sub-pattern $disc$; its associated value $\mathbf{P_{disc_\mathcal{T}}} = \begin{cases} sup_\mathcal{D}(disc)/|\mathcal{D}| \text{ if } disc \subseteq X \\ 0 \text{ otherwise} \end{cases}$
- $F_{disc_\mathcal{I}}$: a numerical descriptor representing the relative size of the discriminating sub-pattern $disc$; its associated value $\mathbf{P_{disc_\mathcal{I}}} = \begin{cases} |disc|/|\mathcal{I}| \text{ if } disc \subseteq X \\ 0 \text{ otherwise} \end{cases}$

By denoting $\mathcal{F}_{disc}$ the set of discriminating descriptors added to $\mathcal{F}$, we obtain the following temporary vector of descriptors: $\langle \underbrace{F_1, \ldots, F_n}_{\mathcal{F}}, \underbrace{F_{disc_X}, F_{disc_\mathcal{T}}, F_{disc_\mathcal{I}}}_{\mathcal{F}_{disc}} \rangle$.

Given the user-defined pattern ranking $\widehat{\mathcal{R}}^t$ at iteration t on query $\mathcal{X}^t$, we learn two weight vectors : the weight vector $w^t_\mathcal{F}$ associated to $\mathcal{F}$ and the weight vector $w^t_{\mathcal{F}_{disc}}$ associated to $\mathcal{F}_{disc}$. As descriptors $\mathcal{F}_{disc}$ are added temporary, the weights learned for $w^t_{\mathcal{F}_{disc}}$ are used back to update the weights $w^t_\mathcal{F}$ in order to be exploited for the next iteration. This new learning schema can be summarized as follows (see Fig. 1 in [7]):
(i) each pattern $X \in \mathcal{X}^t$ is converted into a vector $\mathbf{P} = \langle \mathbf{P_1}, \ldots, \mathbf{P_n}, \mathbf{P_{disc_X}}, \mathbf{P_{disc_\mathcal{T}}}, \mathbf{P_{disc_\mathcal{I}}} \rangle$, where $\mathbf{P_i}$ is the value associated to a feature/descriptor $F_i \in \mathcal{F} \cup \mathcal{F}_{disc}$.
(ii) new weights $w^t_{F_i}$ are learned for each descriptor $F_i \in \mathcal{F} \cup \mathcal{F}_{disc}$. The learned weights $w^t_{\mathcal{F}_{disc}}$ are then used back to update the weights $w^t_\mathcal{F}$ using a multiplicative weight method (see Algorithm 2, line 10).
(iii) finally, a new estimate φ^t is computed using the new $w^t_\mathcal{F}$ (see Algorithm 2, line 11).

3.3 Updating the Weights of Feature Pattern Representations

Let $disc$ be the discriminating sub-pattern extracted from the query $\mathcal{X}^t$. We propose two rules to update the weight vector $w^t_\mathcal{F}$ from the weight vector $w^t_{\mathcal{F}_{disc}}$:

- for *binary features* $F_i \in \mathcal{F}$ (items and transactions):
  - $F_i \equiv items(i, disc)$, $w^t_{F_i} = f^{ag}(w^t_{F_i}, w^t_{F_{disc_X}})$
  - $F_i \equiv Trans(t_i, disc) \wedge t_i \in \mathcal{V}_\mathcal{D}(disc)$, $w^t_{F_i} = f^{ag}(w^t_{F_i}, w^t_{F_{disc_X}})$
- for *numerical features* $F_i \in \mathcal{F}$ (frequency, length, ...):
  - $F_i \equiv Freqency$: $w^t_{F_i} = f^{ag}(w^t_{F_i}, w^t_{F_{disc_\mathcal{T}}})$
  - $F_i \equiv Lenght$: $w^t_{F_i} = f^{ag}(w^t_{F_i}, w^t_{F_{disc_\mathcal{I}}})$

The Multiplicative Weights Method [2] is a simple idea which has been repeatedly discovered in fields as diverse as Machine Learning, and Optimization. The setting for this method is the following: A decision maker (DM) has a choice of n decisions, and needs to repeatedly make a decision. The method assigns initial weights to the DM, and updates these weights multiplicatively and iteratively according to the feedback of how well an expert performed. Following this idea, we propose, at each iteration, to update the feature weights $w^t_\mathcal{F}$ by multiplying them with factors which depend on the learned weights of discriminating descriptors in that iteration. Intuitively, this updating scheme tend to focus higher weight on features that better explain patterns ranked by the user in the long run, thus increasing the probability of being present in patterns of the next iterations. We propose to multiplicative updating rules:

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	θ	$\#\mathcal{P}$
Anneal	812	89	660	149 331
Chess	3 196	75	2 014	155118
German	1 000	110	350	161 858
Heart-cleveland	296	95	115	153 214
Hepatitis	137	68	35	148 289
Kr-vs-kp	3 196	73	2 014	155118
Lymph	148	68	48	146 969
Mushroom	8 124	112	813	155657
Soybean	630	50	28	143 519
Vote	435	48	25	142 095
Zoo-1	101	36	10	151 806

(a) Dataset Characteristics.

Notation	Significance
$t \in [T]$	Iteration index
$\mathcal{X}^t$	User query
$\widehat{R}^t$	User-defined feedback on $\mathcal{X}^t$
$\mathcal{F}$	Vector of feature representations of patterns
$\mathcal{F}_{disc}$	Vector of discriminating descriptors of patterns
$\mathbf{P}$	Pattern description w.r.t $(\mathcal{F} \cup \mathcal{F}_{disc})$
$disc$	Discriminating sub-pattern extracted from $\widehat{R}^t$
F_{disc_X}	Binary descriptor related to the presence/absence of $disc$ in X
$F_{disc_\mathcal{T}}$	Numerical descriptor related to the frequency of $disc$
$F_{disc_\mathcal{I}}$	Numerical descriptor related to the relative size $disc$
$w^t_\mathcal{F}$	Weight vector associated to static features $\mathcal{F}$
$w^t_{\mathcal{F}_{disc}}$	Weight vector associated to dynamic features $\mathcal{F}_{disc}$
η	Regularization parameter
$\varphi_{logistic}$	Learned logistic function

(b) Notations.

Table 1: θ represents an absolute value.

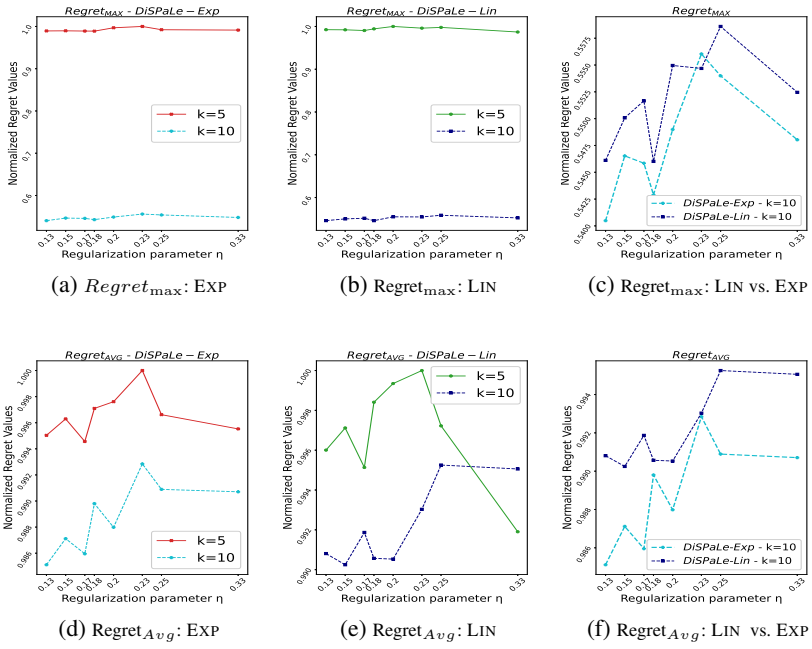
- by a *linear factor*: $f^{ag}(w^t_{F_i}, w^t_{F_{disc}}) = w^t_{F_i} \times (1 + \eta \cdot w^t_{F_{disc}})$
- by an *exponential factor*: $f^{ag}(w^t_{F_i}, w^t_{F_{disc}}) = w^t_{F_i} \times \exp^{\eta \cdot w^t_{F_{disc}}}$

where $\eta \in]0, \frac{1}{2}]$ is regularization parameter used to control the increase in weights resulting from this update. In our experiments (see Section 4), we compare both updating rules for learning weights.

Example 2. Let us consider example 1 and $disc = \{7\}$. Let us assume that $\mathcal{F}$ represents items and frequency features and $\mathcal{F}_{disc} = \langle F_{disc_X}, F_{disc_\mathcal{T}} \rangle$. Suppose that $Freq(X_1) = 0.54$, $Freq(X_3) = 0.36$ and $Freq(disc) = 0.63$. According to $\mathcal{F}$, $X_1 = \{5, 7\}$ is represented by the vector $\mathbf{P}_1 = \langle 0, 0, 0, 0, 1, 0, 1, 0.54 \rangle$, while $X_3 = \{1\}$ by $\mathbf{P}_3 = \langle 1, 0, 0, 0, 0, 0, 0, 0.36 \rangle$. Using additionally features $\mathcal{F}_{disc}$, we obtain the new vector $\mathbf{P}_1 = \langle 0, 0, 0, 0, 1, 0, 1, 0.54, \mathbf{1}, \mathbf{0.63} \rangle$ since $disc \subset X_1$. Similarly, for $X_3 = \{1\}$, $\mathbf{P}_3 = \langle 0, 0, 0, 0, 1, 0, 1, 0.36, \mathbf{0}, \mathbf{0} \rangle$. Let $t = 1$, to learn the weights $w^1_\mathcal{F}$ and $w^1_{\mathcal{F}_{disc}}$, the user's feedback is translated into pairwise rankings and distances between vectors $\mathbf{P}_i$ for each pair are calculated (see Section 2). After the learning step, we obtain $w^1_\mathcal{F} = \langle -0.33, 0.99, 0, -0.99, 0.33, 0, 1.33, 0.15 \rangle$ and $w^1_{\mathcal{F}_{disc}} = \langle 1.33, 0.84 \rangle$. Using the linear factor with $\eta = 0.2$, we update the weight $w^1_{F_7}$ associated to item 7 (since $7 \in disc$) and the weight $w^1_{F_{disc_\mathcal{T}}}$ associated to frequency as follows: $w^1_{F_7} = w^1_{F_7} \times (1 + \eta \cdot w^1_{F_{disc_X}}) = 1.68$; $w^1_{F_{disc_\mathcal{T}}} = w^1_{F_{disc_\mathcal{T}}} \times (1 + \eta \cdot w^1_{disc_\mathcal{T}}) = 0.175$. After the updating step, the resulting weight vector $w^1_\mathcal{F} = \langle -0.33, 0.99, 0, -0.99, 0.33, 0, \mathbf{1.68}, \mathbf{0.175} \rangle$.

4 Experiments

a) Evaluation Methodology and Pattern Selection. To experimentally evaluate our approach DISPALE, we emulate user feedback using a (hidden) quality measure Φ, which is not known to the learning algorithm. We follow the same protocol used in [4]: for each dataset, a set $\mathcal{P}$ of frequent patterns is mined without prior user knowledge. We assume that there exists a user ranking $\widehat{R}$ on the set $\mathcal{P}$, derived from Φ, i.e. $X \succ Y \Leftrightarrow \Phi(X) > \Phi(Y)$. Thus, the task is to learn a logistic function $\varphi_{logistic}$ to sample frequent



(a) $Regret_{\max}$: EXP (b) $\text{Regret}_{\max}$: LIN (c) $\text{Regret}_{\max}$: LIN vs. EXP

(d) Regret_{Avg}: EXP (e) Regret_{Avg}: LIN (f) Regret_{Avg}: LIN vs. EXP

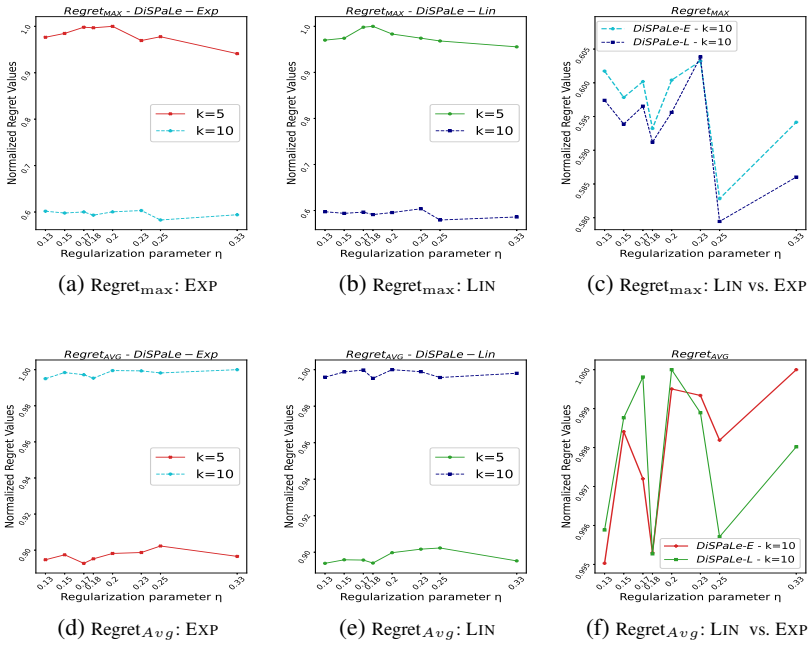
Fig. 1: Effects of the parameter η of DISPALE on $Regret_{max}$, $Regret_{Avg}$ and $\ell = 0$. Results are aggregated over data sets and the three feature combinations (I, IT, ILFT). Regret values are normalized to the range [0,1] based on the maximum regret value.

patterns which approximates Φ. We use surprisingness $surp$ as Φ, where $surp(X) = \max\{sup_{\mathcal{D}}(X) - \prod_{i=1}^{|X|} sup_{\mathcal{D}}(\{i\}), 0\}$.

To compare the performance of the different approaches, we use *cumulative regret*, which is the difference between the ideal value of a certain measure M and its observed value, summed over iterations for each dataset. At each iteration t, we evaluate the regret of ranking pattern X_i by Φ as follows: we compute the percentile rank $pct.rank(X_i)$ by Φ of each pattern $X_i \in \mathcal{X}^t$ $(1 \leq i \leq k)$ as $pct.rank(X_i) = (DI + \frac{DE}{2})/|\mathcal{P}|$ where $DI = |Y \in \mathcal{P},\ \Phi(Y) < \Phi(X_i)|$ and $DE = |Y \in \mathcal{P},\ \Phi(Y) = \Phi(X_i)|$. The percentile rank over all patterns of $\mathcal{X}^t$ measures the ability of the learnable function $\varphi_{logistic}$ to extract interesting patterns, i.e. patterns X_i for which $\Phi(X_i)$ is higher. Thus, the ideal value is 1 (e.g., the highest possible value of Φ over all frequent patterns has the percentile rank of 1). The regret is then defined as $1 - M_{(1 \leq i \leq k)}(pct.rank(X_i))$ where $M \in \{\max, Avg\}$ and $k = |\mathcal{X}^t|$. We repeat each experiment 10 times with different random seeds; the average cumulative regret is reported. We ensure that all compared methods are sampled on the same pattern bases at each iteration.

For the evaluation, we used UCI data-sets, available at the CP4IM repository[4]. For each dataset, we set the minimal support threshold such that the size of $\mathcal{P}$ is approxi-

[4] https://dtai.cs.kuleuven.be/CP4IM/datasets/



(a) $\text{Regret}_{\max}$: EXP (b) $\text{Regret}_{\max}$: LIN (c) $\text{Regret}_{\max}$: LIN vs. EXP

(d) Regret_{Avg}: EXP (e) Regret_{Avg}: LIN (f) Regret_{Avg}: LIN vs. EXP

Fig. 2: Effects of the parameter η of DISPALE on $Regret_{max}$, $Regret_{Avg}$ and $\ell = 1$.

mately $145,000$ frequent patterns. Table 1a shows the data set statistics. Each experiment involves 100 iterations. We compare DISPALE with two state-of-the-art interactive methods, LETSIP [4], an interactive sampling method to learn a logistic function, and an active preference learning to learn a linear ranking function using RANKSVM. We address the following two **research questions**: (i) What effect do DISPALE 's parameters have on the quality of learned patterns? (ii) How does DISPALE compares to LETSIP and RANKSVM?

To select the k patterns to show to the user, we use EFLEXICS [6] to draw the k weighted random samples proportional to $\varphi_{logistic}$ as in [4] (see Suppl. Mat. [7] for more details). These patterns are selected according to a TOP(m) strategy, which picks the m highest-quality patterns. The same procedure is also used in RANKSVM. Moreover, to help users to relate the queries to each other, we retain the top ℓ patterns from the previous query and only sample $(k-\ell)$ new patterns. We use the default values suggested by [4] for the parameters in EFLEXICS: $\lambda = 0.001$, $\kappa = 0.9$, $A = 0.1$ and TOP(1).

b) Parameter Settings of DISPALE. We evaluate the effects of different parameter settings on DISPALE: the query size $k \in \{5, 10\}$, the updating rule and the regularization parameter η. We use the following feature combination: $Items$ (I); $Items + Transactions$ (IT); and $Items + Length + Frequency + Transactions$ (ILFT). We consider two settings for ℓ: $\ell = 0$ and $\ell = 1$. Figure 1 shows the evolution of the

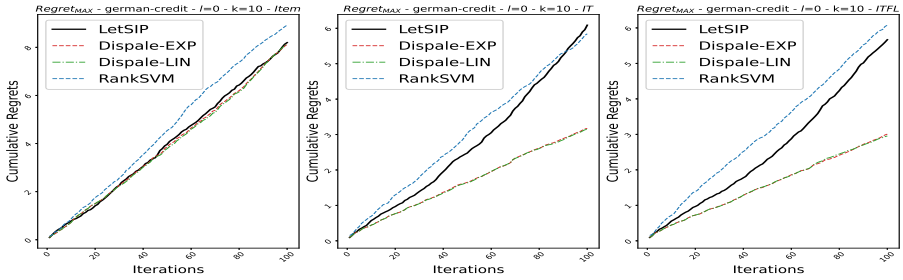
Table 2: Evaluation of the importance of pattern features and comparison of DISPALE-EXP ($\eta = 0.13$) with LETSIP and RANKSVM for $k = 10$. Results are aggregated over all datasets. (1): LETSIP, (2): DISPALE-EXP, (3): RANKSVM. Detailed values of the regret for each dataset and results of DISPALE-LIN are given in [7].

	$\ell = 0$						$\ell = 1$					
	Regret: $Regret_{\max}$			Regret: $Regret_{Avg}$			Regret: $Regret_{\max}$			Regret: $Regret_{Avg}$		
Descriptors	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
I	**112.137**	114.567	123.130	554.816	**553.050**	582.303	**10.438**	11.438	11.465	**496.918**	499.151	521.634
IT	108.446	**91.528**	101.635	543.556	**492.967**	542.595	10.761	11.465	**9.192**	483.689	**449.444**	491.014
ITLF	106.006	**88.391**	100.162	538.848	**487.537**	540.844	11.275	11.579	**9.601**	490.818	**450.202**	490.649

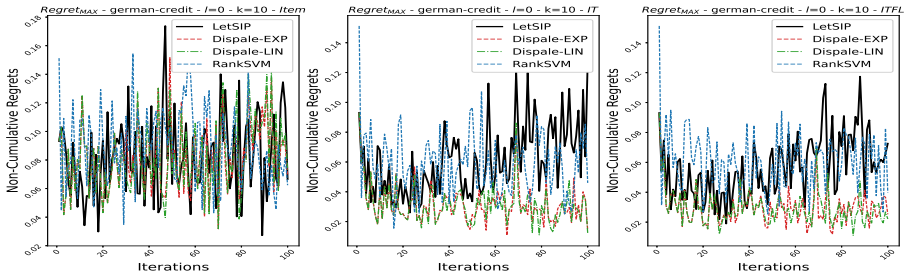
values of $Regret_{\max}$ and $Regret_{Avg}$ for different values of η and for ℓ = 0. Figures 1a and 1b show that both updating rules (LIN and EXP) ensure the lowest quality regrets with $k = 10$ w.r.t. $Regret_{\max}$. This indicates that our approach is able to identify the properties of the target ranking from ordered lists of patterns even when the query size increases. Additionally, the lowest regret values are obtained with $\eta = 0.13$. Regarding $Regret_{Avg}$ (see Figures 1d and 1e), $k = 10$ continues to be the better query size and $\eta = 0.13$ gives the lowest regret values. Finally, Figures 1c and 1f compares the regret values of DISPALE-EXP and DISPALE-LIN for $k = 10$. As we can seen, DISPALE-EXP allows to achieve the best regrets. Figure 2 shows the effect of DISPALE's parameters on regret values for $\ell = 1$. Retaining one highest-ranked pattern from the previous query w.r.t. $Regret_{\max}$ does not affect the conclusions drawn previously: $k = 10$ being the better query size. However, we can see the opposite behaviour w.r.t. $Regret_{Avg}$ (see Figures 2d and 2e): querying 5 patterns allows attaining low regret values. Interestingly, as Figure 2f shows, DISPALE-EXP outperforms DISPALE-LIN on almost all values of η. Based on these findings, we set $k = 10$ and $\eta = 0.13$ for the next experiments.

c) Evaluating the Importance of Pattern Features. Table 2 compare different combination of feature representations of patterns for two settings of query retention ℓ. As we can see, additional features provide valuable information to learn more accurate pattern rankings, particularly for DISPALE-EXP where the regrets decrease when adding the feature T. However, the importance of features depends on the pattern type and the target measure Φ [5]. For surprising pattern mining, $Length$ is the most likely to be included in the best feature set, because long patterns tend to have higher values of Surprisingness. $Items$ are important as well, because individual item frequencies are directly included in the formula of Surprisingness. $Transactions$ are important because this feature set helps capture interactions between other features, albeit indirectly.

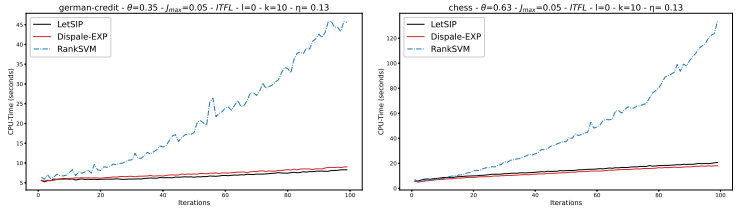
d) Comparing DISPALE with LETSIP and RANKSVM. Table 2 reports the regret values. When considering $Items$ as a feature, LETSIP performs the best. However, selecting queries uniformly at random allows DISPALE-EXP to improve slightly the $Regret_{Avg}$. Moreover, regarding the other features (IT and ITLF), DISPALE-EXP always outperforms LETSIP and RANKSVM. When retaining one highest-ranked pattern ($\ell = 1$), RANKSVM exhibits the lowest $Regret_{\max}$ values, while LETSIP and DISPALE-EXP perform very similarly. However, for $Regret_{Avg}$, DISPALE-EXP performs the best. These results indicate that the learned ranks by DISPALE-EXP in the target



(a) GERMAN-CREDIT: cumulative $\text{Regret}_{\max}$ values.



(b) GERMAN-CREDIT: non-cumulative $\text{Regret}_{\max}$ values.



(c) German-credit: runtime (d) Chess: runtime

Fig. 3: A detailed view of comparison for different pattern features, $k = 10$ and $\ell = 0$.

ranking are more accurate compared to those learned by the alternatives. This confirm the advantage of using discriminants sub-patterns as descriptors.

Figure 3 presents a detailed view of comparison on GERMAN-CREDIT dataset (other results are given in [7]). Curves show the evolution of the regret (cumulative and non cumulative) over 100 iterations of learning for different combination of features. The results confirm again the capacity of DISPALE-EXP to identify frequent patterns with lowest regrets. Figures 3c and 3d compares the performance of the three approaches in terms of CPU-times on two datasets. Overall, learning more complex descriptors from the user-defined pattern ranking does not add significantly to the runtimes of our approach: on most of the data sets considered, DISPALE-EXP and LETSIP behave very similarly, and the difference is very negligible (see Suppl. Mat [7] for other results). However, RANKSVM requires much more time to learn a linear ranking function.

5 Conclusion

In this paper, we have proposed a new approach to the state-of-the art of interactive pattern mining: instead of using static low-level features that have been pre-defined before the process starts, our approach learns more complex descriptors from the user-defined pattern ranking. These features allow to capture the importance of item interactions, and, as shown experimentally, lead to lower cumulative and individual regret than using low-level features. We have explored two multiplicative updating rules for mapping weights learned for complex features back to their component items, and find that the exponential factor gives better results on most of the data sets we worked with. We have evaluated our proposal only on itemset data so far since the majority of existing work is defined for this kind of data. But the importance of using complex dynamic features can be expected to be even higher when interactively mining complex, i.e. sequential, tree, or graph-structured data. We will explore this direction in future work.

Acknowledgements A. Hien and S. Loudni were financially support by the ANR project InvolvD (ANR-20-CE23-0023).

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th VLDB. pp. 487–499. Santiago de Chile, Chile (Sep 1994)
2. Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: a meta-algorithm and applications. Theory Comput. **8**(1), 121–164 (2012)
3. Bhuiyan, M., Hasan, M.A.: Interactive knowledge discovery from hidden data through sampling of frequent patterns. Stat. Anal. Data Min. **9**(4), 205–229 (2016)
4. Dzyuba, V., van Leeuwen, M.: Learning what matters - sampling interesting patterns. In: PAKDD 2017, Proceedings, Part I. pp. 534–546 (2017)
5. Dzyuba, V., van Leeuwen, M., Nijssen, S., Raedt, L.D.: Interactive learning of pattern rankings. International Journal on Artificial Intelligence Tools **23**(6) (2014)
6. Dzyuba, V., van Leeuwen, M., Raedt, L.D.: Flexible constrained sampling with guarantees for pattern mining. Data Min. Knowl. Discov. **31**(5), 1266–1293 (2017)
7. Hien, A., Loudni, S., Aribi, N., Ouali, A., Zimmermann, A.: Code and supplementary material. https://gitlab.com/phdhien/dispale (2023)
8. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the ACM SIGKDD KDD 2002. pp. 133–142. New York, NY, USA (2002)
9. Kamishima, T., Kazawa, H., Akaho, S.: A Survey and Empirical Comparison of Object Ranking Methods, pp. 181–201 (09 2010)
10. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium. pp. 226–236 (2000)
11. Raedt, L.D., Zimmermann, A.: Constraint-based pattern set mining. In: Proceedings of the 17th SIAM ICDM 2007, Minneapolis, Minnesota, USA. pp. 237–248. SIAM (2007)
12. Rüping, S.: Ranking interesting subgroups. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) Proceedings of ICML 2009. vol. 382, pp. 913–920 (2009)
13. Shalev-Shwartz, S., Tewari, A.: Stochastic methods for l_1-regularized loss minimization. J. Mach. Learn. Res. **12**, 1865–1892 (2011)
14. Tan, P.N., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: KDD. pp. 32–41 (2002)