Anytime Subgroup Discovery in Numerical Domains with Guarantees

Aimene Belfodil, Adnene Belfodil, Mehdi Kaytoue

Anytime Subgroup Discovery in Numerical Domains with Guarantees

Aimene Belfodil *,[1,2], Adnene Belfodil*,[1], and Mehdi Kaytoue[1,3]

[1] Univ Lyon, INSA Lyon, CNRS, LIRIS UMR 5205, F-69621, LYON, France
[2] Mobile Devices Ingénierie, 100 Avenue Stalingrad, 94800, Villejuif, France
[3] Infologic, 99 avenue de Lyon, 26500 Bourg-Lès-Valence, France
firstname.surname@insa-lyon.fr

Abstract. Subgroup discovery is the task of discovering patterns that accurately discriminate a class label from the others. Existing approaches can uncover such patterns either through an exhaustive or an approximate exploration of the pattern search space. However, an exhaustive exploration is generally unfeasible whereas approximate approaches do not provide guarantees bounding the *error* of the best pattern quality nor the exploration progression ("*How far are we of an exhaustive search*"). We design here an algorithm for mining numerical data with three key properties w.r.t. the state of the art: (i) It yields progressively interval patterns whose quality improves over time; (ii) It can be interrupted anytime and always gives a guarantee bounding the *error* on the top pattern quality and (iii) It always bounds a distance to the exhaustive exploration. After reporting experimentations showing the effectiveness of our method, we discuss its generalization to other kinds of patterns.

Keywords: Subgroup discovery, Anytime algorithms, Discretization

1 Introduction

We address the problem of discovering patterns that accurately discriminate one class label from the others in a numerical dataset. Subgroup discovery (SD) [27] is a well established pattern mining framework which strives to find out data regions uncovering such interesting patterns. When it comes to numerical attributes, a pattern is generally a conjunction of restrictions over the attributes, e.g., pattern $50 \leq age < 70 \wedge smoke\_per\_day \geq 3$ fosters lung cancer incidence. To look for such patterns (namely interval patterns), various approaches are usually implemented. Common techniques perform *a discretization* transforming the numerical attributes to categorical ones in a pre-processing phase before using the wide spectrum of existing mining techniques [2,20,22,3]. This leads, however, to a loss of information even if an exhaustive enumeration is performed on the transformed data [2]. Other approaches explore the whole search space of all restrictions either exhaustively [18,14,6] or heuristically [23,5]. While an exhaustive enumeration is generally unfeasible in large data, the various state-of-the-art

*Both authors contributed equally to this work.

algorithms that heuristically explore the search space provide no provable guarantee on how they *approximate* the top quality patterns and on how far they are from an exhaustive search. Recent techniques set up a third and elegant paradigm, that is direct sampling approaches [3,4,13]. Algorithms falling under this category are non-enumerative methods which directly sample solutions from the pattern space. They simulate a distribution which rewards high quality patterns with respect to some interestingness measure. While [3,4] propose a direct two-step sampling procedure dedicated for categorical/boolean datasets, authors in [13] devise an interesting framework which add a third step to handle the specificity of numerical data. The proposed algorithm addresses the discovery of dense neighborhood patterns by defining a new density metric. Nevertheless, it does not consider the discovery of discriminant numerical patterns in labeled numerical datasets. Direct sampling approaches abandon the completeness property and generate only approximate results. In contrast, anytime pattern mining algorithms [5,16] are enumerative methods which exhibits the anytime feature [29], a solution is always available whose quality improves gradually over time and which converges to an exhaustive search if given enough time, hence ensuring completeness. However, to the best of our knowledge, no existing anytime algorithm in SD framework, makes it possible to ensure guarantees on the patterns discriminative power and the remaining distance to an exhaustive search while taking into account the nature of numerical data.

To achieve this goal, we propose a novel anytime algorithm, `RefineAndMine`, tailored for discriminant interval patterns discovery in numerical data. It starts by mining interval patterns in a coarse discretization, followed by successive refinements yielding increasingly finer discretizations highlighting potentially new interesting patterns. Eventually, it performs an exhaustive search, if given enough time. Additionally, our method gives two provable guarantees at each refinement. The first evaluates how close is the best found pattern so far to the optimal one in the whole search space. The second measures how already found patterns are diverse and cover well all the interesting regions in the dataset.

The outline is as follows. We recall in Sec. 2 basic definitions. Next, we define formally the problem in Sec. 3. Subsequently We introduce in Sec. 4 our mining algorithm before formulating the guarantees it provides in Sec. 5. We empirically evaluate the efficiency of `RefineAndMine` in Sec. 6 and discuss its potential improvements in Sec. 7. Additional materials are available in our companion page[4]. For more details and proofs, please refer to the technical report[5].

2 Preliminaries

Input. A *labeled numerical dataset* $(\mathcal{G}, \mathcal{M})$ is given by a finite set (of objects) $\mathcal{G}$ partitioned into two subsets $\mathcal{G}^+$ and $\mathcal{G}^-$ enclosing respectively positive (target) and negative instances; and a sequence of numerical attributes $\mathcal{M} = (m_i)_{1 \leq i \leq p}$ of size $p = |\mathcal{M}|$. Each *attribute* m_i is an application $m_i : \mathcal{G} \to \mathbb{R}$ that associates

[4] `https://github.com/Adnene93/RefineAndMine`

[5] `https://goo.gl/NWtXfp`

to each object $g \in \mathcal{G}$ a value $m_i(g) \in \mathbb{R}$. We can also see $\mathcal{M}$ as a mapping $\mathcal{M} : \mathcal{G} \to \mathbb{R}^p, g \mapsto (m_i(g))_{1 \leq i \leq p}$. We denote $m_i[\mathcal{G}] = \{m_i(g) \mid g \in \mathcal{G}\}$ (More generally, for a function $f : E \to F$ and a subset $A \subseteq E$, $f[A] = \{f(e) \mid e \in A\}$). Fig. 1 (left table) presents a 2-dimensional labeled numerical dataset and its representation in the Cartesian plane (filled dots represent positive instances).

Interval patterns and their extents. When dealing with numerical domains in SD, we generally consider for intelligibility *interval patterns* [18]. An *Interval pattern* is a conjunction of restrictions over the numerical attributes; i.e. a set of conditions *attribute* $\geqslant v$ with $\geqslant \in \{=, \leq, <, \geq, >\}$. Geometrically, interval patterns are *axis-parallel hyper-rectangles*. Fig. 1 (center-left) depicts pattern (non-hatched rectangle) $c_2 = (1 \leq m_1 \leq 4) \wedge (0 \leq m_2 \leq 3) \triangleq [1, 4] \times [0, 3]$.

Interval patterns are naturally partially ordered thanks to "hyper-rectangle inclusion". We denote the *infinite partially ordered set* (*poset*) of all interval patterns by $(\mathcal{D}, \sqsubseteq)$ where $\sqsubseteq$ (same order used in [18]) denotes the dual order $\supseteq$ of hyper-rectangle inclusion. That is pattern $d_1 \sqsubseteq d_2$ *iff* d_1 encloses d_2 ($d_1 \supseteq d_2$). It is worth mentioning that $(\mathcal{D}, \sqsubseteq)$ forms a *complete lattice* [26]. For a subset $S \subseteq \mathcal{D}$, the join $\bigsqcup S$ (i.e. smallest upper bound) is given by the rectangle intersection. Dually, the meet $\bigsqcap S$ (i.e the largest lower bound) is given by the smallest hyper-rectangle enclosing all patterns in S. Note that the top (resp. bottom) pattern in $(\mathcal{D}, \sqsubseteq)$ is given by $\top = \emptyset$ (resp. $\bot = \mathbb{R}^p$). Fig. 1 (right) depicts two patterns (hatched) $e_1 = [1, 5] \times (1, 4]$ and $e_2 = [0, 4) \times [2, 6]$, their meet (non hatched) $e_1 \sqcap e_2 = [0, 5] \times (1, 6]$ and their join (black) $e_1 \sqcup e_2 = [1, 4) \times [2, 4]$.

A pattern $d \in \mathcal{D}$ is said to cover an object $g \in \mathcal{G}$ iff $\mathcal{M}(g) \in d$. To use the same order $\sqsubseteq$ to define such a relationship, we associate to each $g \in \mathcal{G}$ its corresponding pattern $\delta(g) \in \mathcal{D}$ which is the degenerated hyper-rectangle $\delta(g) = \{\mathcal{M}(g)\} = \bigtimes_{i=1}^{p} [m_i(g), m_i(g)]$. The cover relationship becomes $d \sqsubseteq \delta(g)$. The *extent* of a pattern is the set of objects supporting it. Formally, there is a function $ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\} = \{g \in \mathcal{G} \mid \mathcal{M}(g) \in d\}$ (where $\wp(\mathcal{G})$ denotes the set of all subsets of $\mathcal{G}$). Note that if $d_1 \sqsubseteq d_2$ then $ext(d_2) \subseteq ext(d_1)$. We define also the positive (resp. negative) extent as follows: $ext^+(d) = ext(d) \cap \mathcal{G}^+$ (resp. $ext^-(d) = ext(d) \cap \mathcal{G}^-$). With the mapping $\delta : \mathcal{G} \to \mathcal{D}$ and the complete lattice $(\mathcal{D}, \sqsubseteq)$, we call the triple $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ the *interval pattern structure* [18,10].

	m_1	m_2	label
g_1	1	2	+
g_2	1	3	+
g_3	2	1	+
g_4	3	5	+
g_5	2	4	−
g_6	2	5	−
g_7	3	4	−
g_8	4	4	−

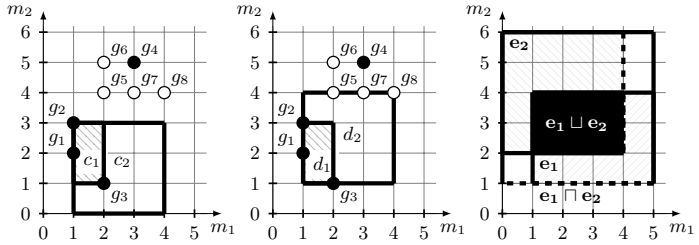

Fig. 1: (left to right) (1) a labeled numerical dataset. **(2)** closed c_1 vs non-closed c_2 interval patterns. **(3)** `cotp` d_1 vs non `cotp` d_2. **(4)** meet and join of two patterns.

Measuring the discriminative power of a pattern. In SD, a quality measure $\phi : \mathcal{D} \to \mathbb{R}$ is usually defined to evaluate at what extent a pattern *well-discriminates* the positive instances in $\mathcal{G}^+$ from those in $\mathcal{G}^-$. Two atomic measures are generally employed to quantify the quality of a pattern d: the *true positive rate* $tpr : d \to |ext^+(d)|/|\mathcal{G}^+|$ and the *false positive rate* $fpr : d \to |ext^-(d)|/|\mathcal{G}^-|$. Several measures exist in the literature [12,21]. A measure is said to be *objective* or *probability based* [12] if it depends solely on the number of co-occurrences and non co-occurrences of the pattern and the target label. In other words, those measures can be defined using only tpr, fpr and potentially other constants (e.g. $|\mathcal{G}|$). Formally, $\exists \phi^* : [0,1]^2 \to \mathbb{R}$ s.t. $\phi(d) = \phi^*(tpr(d), fpr(d))$. Objective measures depends only on the pattern *extent*. Hence, we use interchangeably $\phi(ext(d))$ and $\phi(d)$. An objective quality measure ϕ is said to be *discriminant* if its associated measure ϕ^* is *increasing* with tpr (fpr being fixed) and *decreasing* with fpr (tpr being fixed). For instance, with $\alpha^+ = |\mathcal{G}^+|/|\mathcal{G}|$ and $\alpha^- = |\mathcal{G}^-|/|\mathcal{G}|$ denoting labels prevalence, $wracc^*(tpr, fpr) = \alpha^+ \cdot \alpha^- \cdot (tpr - fpr)$ and $informedness^*(tpr, fpr) = tpr - fpr$ are discriminant measures.

Compressing the set of interesting patterns using closure. Since discriminant quality measures depend only on the extent, *closed patterns* can be leveraged to reduce the number of resulting patterns [10]. A pattern $d \in \mathcal{D}$ is said to be closed (w.r.t. pattern structure $\mathbb{P}$) if and only if it is the most restrictive pattern (i.e. the smallest hyper-rectangle) enclosing its extent. Formally, $d = int(ext(d))$ where int mapping (called *intent)* is given by: $int : \wp(\mathcal{G}) \to \mathcal{D}, A \mapsto \bigsqcap_{g \in A} \delta(g) = \bigtimes_{i=1}^{p} [\min_{g \in A} m_i(g), \max_{g \in A} m_i(g)]$. Fig. 1 (center-left) depicts the closed interval pattern (hatched rectangle) $c_1 = [1,2] \times [1,3]$ which is the closure of $c_2 = [1,4] \times [0,3]$ (non hatched rectangle). Note that since $\mathcal{G}$ is finite, the set of all closed patterns is finite and is given by $int[\wp(\mathcal{G})]$.

A more concise set of patterns using Relevance theory. Fig. 1 (center-right) depicts two interval patterns, the hatched pattern $d_1 = [1,2] \times [1,3]$ and the non-hatched one $d_2 = [1,4] \times [1,4]$. While both patterns are *closed*, d_1 has better discriminative power than d_2 since they both cover exactly the same positive instances $\{g_1, g_2, g_3\}$; yet, d_2 covers more negative instances than d_1. *Relevance theory* [11] formalizes this observation and helps us to remove some clearly uninteresting closed patterns. In a nutshell, a closed pattern $d_1 \in \mathcal{D}$ is said to be *more relevant than* a closed pattern $d_2 \in \mathcal{D}$ iff $ext^+(d_2) \subseteq ext^+(d_1)$ and $ext^-(d_1) \subseteq ext^-(d_2)$. For ϕ discriminant, if d_1 is more relevant than d_2 then $\phi(d_1) \geq \phi(d_2)$. A closed pattern d is said to be *relevant* iff there is no other closed pattern c that is more relevant than d. It follows that if a closed pattern is relevant then it is *closed on the positive* (`cotp` for short). An interval pattern is said to be `cotp` if any smaller interval pattern will at least drop one positive instance (i.e. $d = int(ext^+(d))$). interestingly, $int \circ ext^+$ is a closure operator on $(\mathcal{D}, \sqsubseteq)$. Fig. 1 (center-right) depicts a non `cotp` pattern $d_2 = [1,4] \times [1,4]$ and its closure on the positive $d_1 = int(ext^+(d_2)) = [1,2] \times [1,3]$ which is relevant. Note that not all `cotp` are *relevant*. The set of `cotp` patterns is given by $int[\wp(\mathcal{G}^+)]$. We call *relevant (resp.* `cotp`*) extent*, any set $A \subseteq \mathcal{G}$ s.t. $A = ext(d)$ with d is a *relevant (resp.* `cotp`*)* pattern. The set of relevant extents is denoted by $\mathcal{R}$.

3 Problem Statement

Correct enumeration of relevant extents. First, consider the (simpler) problem of enumerating all relevant extents in $\mathcal{R}$. For a (relevant extents) enumeration algorithm, three properties need generally to hold. An algorithm which output is the set of solutions $\mathcal{S}$ is said to be (1) *complete* if $\mathcal{S} \supseteq \mathcal{R}$, (2) *sound* if $\mathcal{S} \subseteq \mathcal{R}$ and (3) *non redundant* if each solution in $\mathcal{S}$ is outputted *only once*. It is said to be *correct* if the three properties hold. Guyet et al. [15] proposed a *correct algorithm* that enumerate relevant extents induced by the *interval pattern structure* in two steps: (1) Start by a *DFS complete* and *non redundant* enumeration of all `cotp` patterns (extents) using `MinIntChange` algorithm [18]; (2) Post-process the found `cotp` patterns by removing non relevant ones using [11] characterization (this step adds the *soundness* property to the algorithm).

Problem Statement. Given a discriminant objective quality measure ϕ, we want to design an *anytime enumeration algorithm* such that: (1) given enough time, outputs all relevant extents in $\mathcal{R}$, (2) when interrupted, provides a guarantee bounding the difference of quality between the top-quality found extent and the top possible quality w.r.t. ϕ; and (3) outputs a second guarantee ensuring that the resulting patterns are diverse.

Formally, let $\mathcal{S}_i$ be the set of outputted solutions by the anytime algorithm at some step (or instant) i (at $i+1$ we have $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$). We want that (1) when i is big enough, $\mathcal{S}_i \supseteq \mathcal{R}$ (only *completeness* is required). For (2) and (3), we define two metrics[6] to compare the results in $\mathcal{S}_i$ with the ones in $\mathcal{R}$. The first metric, called *accuracy* (eq. 1), evaluates the difference between top pattern quality ϕ in $\mathcal{S}_i$ and $\mathcal{R}$ while the second metric, called $specificity$ (eq. 2), evaluates how diverse and complete are patterns in $\mathcal{S}_i$.

$$accuracy_\phi(S_i, \mathcal{R}) = \sup_{A \in \mathcal{R}} \phi(A) - \sup_{B \in \mathcal{S}_i} \phi(B) \tag{1}$$

$$specificity(S_i, \mathcal{R}) = \sup_{A \in \mathcal{R}} \inf_{B \in \mathcal{S}_i} (|A \,\Delta\, B|/|\mathcal{G}|) \tag{2}$$

The idea behind $specificity$ is that each extent A in $\mathcal{R}$ is "approximated" by the most similar extent in $\mathcal{S}_i$; that is the set $B \in \mathcal{S}_i$ minimizing the *metric distance* $A, B \mapsto |A \,\Delta\, B|/|\mathcal{G}|$ in $\wp(\mathcal{G})$. The $specificity$[7] is then the highest possible distance (pessimistic). Note that $specificity(\mathcal{S}_i, \mathcal{R}) = 0$ *is equivalent to* $\mathcal{S}_i \supseteq \mathcal{R}$. Clearly, the lower these two metrics are, the closer we get to the desired output $\mathcal{R}$. While $accuracy_\phi$ and $specificity$ can be *evaluated* when a complete exploration of $\mathcal{R}$ is possible, our aim is to *bound* the two aforementioned measures independently from $\mathcal{R}$ providing a *guarantee*. In other words, the anytime algorithm need to output additionally to S_i, the two following measures: (2) $\overline{accuracy_\phi}(\mathcal{S}_i)$ and (3) $\overline{specificity}(\mathcal{S}_i)$ s.t. $accuracy_\phi(\mathcal{S}_i, \mathcal{R}) \leq \overline{accuracy_\phi}(\mathcal{S}_i)$ and $specificity(\mathcal{S}_i, \mathcal{R}) \leq \overline{specificity}(\mathcal{S}_i)$. These two bounds need to decrease overtime providing better information on $\mathcal{R}$ through $\mathcal{S}_i$.

[6] The metrics names fall under the taxonomy of [29] for anytime algorithms.

[7] The $specificity$ is actually a directed Hausdorff distance [17] from $\mathcal{R}$ to $\mathcal{S}_i$.

4 Anytime Interval Pattern Mining

Discretizations and pattern space. Our algorithm relies on the enumeration of a chain of discretization from the coarsest to the finest. A *discretization* of $\mathbb{R}$ is any *partition* of $\mathbb{R}$ using intervals. In particular, let $C = \{c_i\}_{1 \leq i \leq |C|} \subseteq \mathbb{R}$ be a finite set with $c_i < c_{i+1}$ for $i \in \{1, ..., |C| - 1\}$. Element of C are called *cut points* or *cuts*. We associate to C a *finite discretization* denoted by $dr(C)$ and given by $dr(C) = \{(-\infty, c_1)\} \cup \{[c_i, c_{i+1}) \mid i \in \{1, ..., |C| - 1]\}\} \cup \{[c_{|C|}, +\infty)\}$.

Generally speaking, let $p \in \mathbb{N}^*$ and let $C = (C_k)_{1 \leq k \leq p} \in \wp(\mathbb{R})^p$ representing *sets of cut points* associated to each dimension k (i.e. $C_k \subseteq \mathbb{R}$ finite $\forall k \in \{1, ..., p\}$). The partition $dr(C)$ of $\mathbb{R}^p$ is given by: $dr(C) = \prod_{k=1}^{p} dr(C_k)$. Fig. 2 depicts two discretizations. Discretizations are ordered using the natural order between partitions[8]. Moreover, cut-points sets are ordered by $\leq$ as follows: $C^1 \leq C^2 \equiv (\forall k \in \{1, ..., p\})\, C_k^1 \subseteq C_k^2$ with $C^i = (C_k^i)_{1 \leq k \leq p}$. Clearly, if $C^1 \leq C^2$ then discretization $dr(C^1)$ *is coarser than* $dr(C^2)$.

Let $C = (C_k)_{1 \leq k \leq p}$ be the cut-points. Using the elementary hyper-rectangles (i.e. cells) in the discretization $dr(C)$, one can build a (finite) subset of descriptions $\mathcal{D}_C \subseteq \mathcal{D}$ which is the set of all possible descriptions (hyper-rectangles) that can be built using these cells. Formally: $\mathcal{D}_C = \{\bigsqcap S \mid S \subseteq dr(C)\}$. Note that $\top = \emptyset \in \mathcal{D}_C$ since $\bigsqcap \emptyset = \bigsqcup \mathcal{D} = \top$ by definition. Proposition 1 states that $(\mathcal{D}_C, \sqsubseteq)$ is a complete sub-lattice of $(\mathcal{D}, \sqsubseteq)$.

Proposition 1. *$(\mathcal{D}_C, \sqsubseteq)$ is a finite (complete) sub-lattice of $(\mathcal{D}, \sqsubseteq)$ that is: $\forall d_1, d_2 \in \mathcal{D}_C : d_1 \sqcup d_2 \in \mathcal{D}_C$ and $d_1 \sqcap d_2 \in \mathcal{D}_C$. Moreover, if $C^1 \leq C^2$ are two cut-points sets, then $(\mathcal{D}_{C^1}, \sqsubseteq)$ is a (complete) sub-lattice of $(\mathcal{D}_{C^2}, \sqsubseteq)$.*

Finest discretization for a complete enumeration of relevant extents. There exist cut points $C \subseteq \wp(\mathbb{R})^p$ such that the space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents (i.e. $ext[\mathcal{D}_C] \supseteq \mathcal{R}$). For instance, if we consider $C = (m_k[\mathcal{G}])_{1 \leq k \leq p}$, the description space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents. However, is there coarser discretization that holds all the relevant extents? The answer is affirmative. One can show that the only interesting cuts are those separating between positive and negative instances (called boundary cut-points by [9]). We call such cuts, *relevant cuts*. They are denoted by $C^{rel} = (C_k^{rel})_{1 \leq k \leq p}$ and we have $ext[\mathcal{D}_{C^{rel}}] \supseteq \mathcal{R}$. Formally, for each dimension k, a value $c \in m_k[\mathcal{G}]$ is a *relevant cut in* C_k^{rel} for attribute m_k iff: $(c \in m_k[\mathcal{G}^+]$ and $prev(c, m_k[\mathcal{G}]) \in m_k[\mathcal{G}^-])$ or $(c \in m_k[\mathcal{G}^-]$ and $prev(c, m_k[\mathcal{G}]) \in m_k[\mathcal{G}^+])$ where $next(c, A) = \inf\{a \in A \mid c < a\}$ (resp. $prev(c, A) = \sup\{a \in A \mid a < c\}$) is the following (resp. preceding) element of c in A. Finding *relevant cuts* C_k^{rel} is of the same complexity of sorting $m_k[\mathcal{G}]$ [9]. In the dataset depicted in Fig. 1, relevant cuts are given by $C^{rel} = (\{2, 3, 4, 5\}, \{4, 5\})$. Discretization $dr(C_2^{rel})$ is depicted in Fig. 2 (center).

[8] Let E be a set, a partition P_2 of E *is finer than* a partition P_1 (or P_1 *is coarser than* P_2) and we denote $P_1 \leq P_2$ if any subset in P_1 is a subset of a subset in P_2.

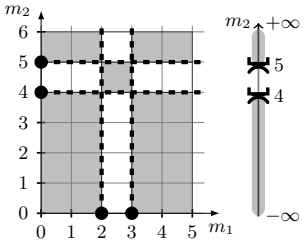

Fig. 2: (left) Discretization $dr((C_1, C_2))$ in $\mathbb{R}^2$ with $C_1 = \{2, 3\}$ and $C_2 = \{4, 5\}$ and **(right)** discretization $dr((C_2))$ in $\mathbb{R}$. Adding a cut point in any C_k will create finer discretization.

Anytime enumeration of relevant extents. We design an *anytime* and *interruptible* algorithm dubbed `RefineAndMine`. This method, presented in Algorithm 1, relies on the enumeration of a chain of discretizations on the data space, from the coarsest to the finest. It begins by searching relevant cuts in pre-processing phase (line 2). Then, it builds a *coarse discretization* (line 3) containing a small set of relevant cut-points. Once the initial discretization built, `cotp` patterns are mined thanks to `MinIntChange` Algorithm (line 4) [18]. Then as long as the algorithm is not interrupted (or within the computational budget), we add new cut-points (line 6) building finer discretizations. For each added cut-point (line 8), only new interval patterns are searched for (mined descriptions d are new but their extents $ext(d)$ are not necessarily new) . That is `cotp` patterns which left or right bound is *cut* on the considered attribute *attr* (i.e. $d.I_{attr} \in \{[cut, a), [cut, +\infty), [a, cut), (-\infty, cut) \mid a \in C^{cur}_{attr}\}$ with $d.I_{attr}$ is the $attr^{th}$ interval of d). This can be done by a slight modification of `MinIntChange` method. `RefineAndMine` terminates when the set of relevant cuts is exhausted (i.e. $C^{cur} = C^{rel}$) ensuring a *complete* enumeration of relevant extents $\mathcal{R}$.

The initial discretization (Line 3) can be done by various strategies (see [28]). A simple, yet efficient, choice is the equal frequency discretization with a fixed number of cuts. Other strategies can be used, e.g. [9]. Adding new cut-points (Line 6) can also be done in various ways. One strategy is to add a random relevant cut on a random attribute to build the next discretization. Section 5.3 proposes another more elaborated strategy that heuristically guide `RefineAndMine` to rapidly find good quality patterns (observed experimentally).

Algorithm 1: `RefineAndMine`

Input: $(\mathcal{G}, \mathcal{M})$ a numerical datasets with $\{\mathcal{G}^+, \mathcal{G}^-\}$ partition of $\mathcal{G}$

1 **procedure** `RefineAndMine`()
2 Compute relevant cuts C^{rel}
3 Build an initial set of cut-points $C^{cur} \leq C^{rel}$
4 Mine `cotp` patterns in $\mathcal{D}_{C^{cur}}$ (and their extents) using `MinIntChange`
5 **while** $C^{cur} \neq C^{rel}$ *and within* ***computational budget*** **do**
6 Choose the next relevant cut $(attr, cut)$ with $cut \in C^{rel}_{attr} \backslash C^{cur}_{attr}$
7 Add the relevant cut *cut* to C^{cur}
8 Mine **new** `cotp` **patterns** (and their extents) in $\mathcal{D}_{C^{cur}}$

5 Anytime Interval Pattern Mining with Guarantees

Algorithm `RefineAndMine` starts by mining patterns in a coarse discretization. It continues by mining more patterns in increasingly finer discretizations until the search space is totally explored (final complete lattice being $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$). According to Proposition 1, the description spaces built on discretizations are complete sub-lattices of the total description space. A similar idea involves performing successive enumeration of growing pattern languages (projections) [6]. In our case, it is a successive enumeration of growing complete sub-lattices. For the sake of generality, in the following of this section $(\mathcal{D}, \sqsubseteq)$ denotes a *complete lattice*, and for all $i \in \mathbb{N}^*$, $(\mathcal{D}_i, \sqsubseteq)$ denotes *complete sub-lattices* of $(\mathcal{D}, \sqsubseteq)$ such that $\mathcal{D}_i \subseteq \mathcal{D}_{i+1} \subseteq \mathcal{D}$. For instance, in `RefineAndMine`, the total complete lattice is $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ while the $(\mathcal{D}_i, \sqsubseteq)$ are $(\mathcal{D}_{C^{cur}}, \sqsubseteq)$ at each step. Following Sec. 3 notation, the outputted set $\mathcal{S}_i$ at a step i contains the set of all ***cotp*** *extents* associated to $\mathcal{D}_i$. Before giving the formulas of $\overline{accuracy_\phi}(\mathcal{S}_i)$ and $\overline{specificity}(\mathcal{S}_i)$, we give some necessary definitions and underlying properties. At the end of this section, we show how `RefineAndMine` can be adapted to efficiently compute these two bounds for the case of interval patterns.

Similarly to the interval pattern structure [18], we define in the general case a *pattern structure* $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ on the *complete lattice* $(\mathcal{D}, \sqsubseteq)$ where $\mathcal{G}$ is a non empty finite set (partitioned into $\{\mathcal{G}^+, \mathcal{G}^-\}$) and $\delta : \mathcal{G} \to \mathcal{D}$ is a mapping associating to each object its description (recall that in interval pattern structure, δ is the degenerated hyper-rectangle representing a single point). The extent *ext* and intent *int* operators are then respectively given by $ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\}$ and $int : \wp(\mathcal{G}) \to \wp(\mathcal{G}), A \mapsto \bigsqcap_{g \in A} \delta(g)$ with $\bigsqcap$ represents the meet operator in $(\mathcal{D}, \sqsubseteq)$ [10].

5.1 Approximating descriptions in a complete sub-lattice

Upper and lower approximations of a pattern. We start by approximating each pattern in $\mathcal{D}$ using two patterns in $\mathcal{D}_i$. Consider for instance Fig. 3 where $\mathcal{D}$ is the space of interval patterns in $\mathbb{R}^2$ while $\mathcal{D}_C$ is the space containing only rectangles that can be built over discretization $dr(C)$ with $C = (\{1, 4, 6, 8\}, \{1, 3, 5, 6\})$. Since the hatched rectangle $d = [3, 7] \times [2, 5.5] \in \mathcal{D}$ does not belong to $\mathcal{D}_C$, two descriptions in $\mathcal{D}_C$ can be used to encapsulate it. The first one, depicted by a gray rectangle, is called the *upper approximation* of d. It is given by the smallest rectangle in $\mathcal{D}_C$ enclosing d. Dually, the second approximation represented as a black rectangle and coined *lower approximation* of d, is given by the greatest rectangle in $\mathcal{D}_C$ enclosed by d. This two denominations comes from Rough Set Theory [25] where lower and upper approximations form together a *rough set* and try to capture the undefined rectangle $d \in \mathcal{D} \backslash \mathcal{D}_C$. Definition 1 formalizes these two approximations in the general case.

Definition 1. *The upper approximation mapping* $\overline{\psi_i}$ *and lower approximation mapping* $\underline{\psi_i}$ *are the mappings defined as follows:*

$$\overline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcup \{c \in \mathcal{D}_i \mid c \sqsubseteq d\} \qquad \underline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcap \{c \in \mathcal{D}_i \mid d \sqsubseteq c\}$$

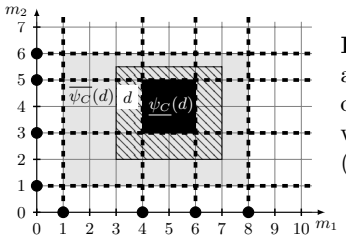

Fig. 3: Description $d = [3, 7] \times [2, 5.5]$ in $\mathcal{D}$ (hatched) and $C = (\{1, 4, 6, 8\}, \{1, 3, 5, 6\})$. Upper approximation of d in $\mathcal{D}_C$ is $\overline{\psi_C}(d) = [1, 8) \times [1, 6)$ (gray rectangle) while lower approximation of d is $\underline{\psi_C}(d) = [4, 6) \times [3, 5)$ (black rectangle).

The existence of these two mappings is ensured by the fact that $(\mathcal{D}_i, \sqsubseteq)$ is a complete sublattice of $(\mathcal{D}, \sqsubseteq)$. *Theorem 4.1* in [8] provides more properties for the two aforementioned mappings. Proposition 2 restates an important property.

Proposition 2. $\forall d \in \mathcal{D} : \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d)$. *The term lower and upper-approximation here are reversed to fit the fact that in term of* extent *we have* $\forall d \in \mathcal{D}$: $ext(\underline{\psi_i}(d)) \subseteq ext(d) \subseteq ext(\overline{\psi_i}(d))$.

A projected pattern structure. Now that we have the upper-approximation mapping $\overline{\psi_i}$, one can associate a new pattern structure $\mathbb{P}_i = (\mathcal{G}, (\mathcal{D}_i, \sqsubseteq), \overline{\psi_i} \circ \delta)$[9] to the pattern space $(\mathcal{D}_i, \sqsubseteq)$. It is worth mentioning, that while extent ext_i mapping associated to $\mathbb{P}_i$ is equal to ext, the intent int_i of $\mathbb{P}_i$ is given by $int_i : \wp(\mathcal{G}) \to \mathcal{D}_i, A \mapsto \overline{\psi_i}(int(A))$. Note that, the set of `cotp` patterns associated to $\mathbb{P}_i$ are given by $int_i[\wp(\mathcal{G}^+)] = \overline{\psi_i}[int[\wp(\mathcal{G}^+)]]$. That is, the upper approximation of a `cotp` pattern in $\mathbb{P}$ is a `cotp` pattern in $\mathbb{P}_i$.

Encapsulating patterns using their upper-approximations. We want to encapsulate any description by knowing only its upper-approximation. Formally, we want some function $f : \mathcal{D}_i \to \mathcal{D}_i$ such that $(\forall d \in \mathcal{D})\, \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq f(\overline{\psi_i}(d))$. Proposition 3 define such a function f (called *core*) and states that the *core* is the tightest (w.r.t. $\sqsubseteq$) possible function f.

Proposition 3. *The function* $core_i$ *defined by:*

$$core_i : \mathcal{D}_i \to \mathcal{D}_i, c \mapsto core(c) = \underline{\psi_i}\left(\bigsqcup \left\{ d \in \mathcal{D} \mid \overline{\psi_i}(d) = c \right\}\right)$$

verifies the following property: $\forall d \in \mathcal{D} : \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d) \sqsubseteq core_i(\overline{\psi_i}(d))$. *Moreover, for* $f : \mathcal{D}_i \to \mathcal{D}_i$, $(\forall d \in \mathcal{D})\, d \sqsubseteq f(\overline{\psi_i}(d)) \Leftrightarrow (\forall c \in \mathcal{D}_i)\, core_i(c) \sqsubseteq f(c)$.

Note that, while the *core* operator definition depends clearly on the complete lattice $(\mathcal{D}, \sqsubseteq)$, its computation should be done independently from $(\mathcal{D}, \sqsubseteq)$.

We show here how to compute the *core* in `RefineAndMine`. In each step and for cut-points $C = (C_k) \subseteq \wp(\mathbb{R})^p$, the finite lattice $(\mathcal{D}_C, \sqsubseteq)$ is a sub-lattice of the finest finite lattice $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ (since $C \leq C^{rel}$). Thereby, the *core* is computed according to this latter as follows: Let $d \in \mathcal{D}_C$ with $d.I_k = [a_k, b_k)$ for all

[9] $\mathbb{P}_i$ is said to be a projected pattern structure of $\mathbb{P}$ by the projection $\overline{\psi_i}$ [7].

$k \in \{1, ..., p\}$. The left (resp. right) bound of $core_C(d).I_k$ for any k is equal to $next(a_k, C_k)$ (resp. $prev(b_k, C_k)$) if $next(a_k, C_k^{rel}) \notin C_k$ (resp. $prev(b_k, C_k^{rel}) \notin C_k$). Otherwise, it is equal to a_k (resp. b_k). Consider the step $\mathcal{C} = (\{2, 3\}, \{4, 5\})$ in `RefineAndMine` (its associated discretization is depicted in Fig. 2 (left)) and recall that the relevant cuts set is $\mathcal{C}^{rel} = (\{2, 3, 4, 5\}, \{4, 5\})$. The core of the bottom pattern $\bot = \mathbb{R}^2$ at this step is $core_{C^{cur}}(\bot) = (-\infty, 3) \times \mathbb{R}$. Indeed, there is three descriptions in $\mathcal{D}_{C^{rel}}$ which upper approximation is $\bot$, namely $\bot$, $c_1 = (-\infty, 4) \times \mathbb{R}$ and $c_2 = (-\infty, 5) \times \mathbb{R}$. Their lower approximations are respectively $\bot$, $(-\infty, 3) \times \mathbb{R}$ and $(-\infty, 3) \times \mathbb{R}$. The join (intersection) of these three descriptions is then $core_{C^{cur}}(\bot) = (-\infty, 3) \times (-\infty, +\infty)$. Note that particularly for interval patterns, the *core* has *monotonicity*, that is $(\forall c, d \in \mathcal{D}_C)\, c \sqsubseteq d \Rightarrow core_C(c) \sqsubseteq core_C(d)$.

5.2 Bounding accuracy and specificity metrics

At the i^{th} step, the outputted extents $\mathcal{S}_i$ contains the set of `cotp` extents in $\mathbb{P}_i$. Formally, $int_i[S_i] \supseteq int_i[\wp(\mathcal{G}^+)]$. Theorem 1 and Theorem 2 gives respectively the *bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$.

Theorem 1. *Let $\phi : \mathcal{D} \to \mathbb{R}$ be a discriminant objective quality measure. The accuracy metric is bounded by:*

$$\overline{accuracy_\phi}(\mathcal{S}_i) = \sup_{c \in int_i[\mathcal{S}_i]} \left[\phi^* \left(tpr\big(c\big), fpr\big(core_i(c)\big)\right) - \phi^* \left(tpr(c), fpr(c)\right)\right]$$

Moreover $\overline{accuracy_\phi}(\mathcal{S}_{i+1}) \leq \overline{accuracy_\phi}(\mathcal{S}_i)$.

Theorem 2. *The specificity metric is bounded by:*

$$\overline{specificity}(\mathcal{S}_i) = \sup_{c \in int_i[\mathcal{S}_i]} \left(\left(|ext(c)| - |ext(core_i^+(c))|\right)/(2 \cdot |\mathcal{G}|)\right)$$

where $core_i^+(c) = int_i(ext^+(core_i(c)))$, that is $core_i^+(c)$ is the closure on the positive of $core_i(c)$ in $\mathbb{P}_i$. Moreover $\overline{specificity}(\mathcal{S}_{i+1}) \leq \overline{specificity}(\mathcal{S}_i)$.

5.3 Computing and updating bounds in `RefineAndMine`

We show below how the different steps of the method `RefineAndMine` (see Algorithm 1) should be updated in order to compute the two bounds $\overline{accuracy}$ and $\overline{specificity}$. For the sake of brevity, we explain here a naive approach to provide an overview of the algorithm. Note that here, *core* (resp. $core^+$) refers to $core_{C^{cur}}$ (resp. $core^+_{C^{cur}}$).

Compute the initial bounds (line 4). As `MinIntChange` enumerates all `cotp` patterns $d \in \mathcal{D}_{C^{cur}}$, `RefineAndMine` stores in a key-value structure (i.e. map) called `BoundPerPosExt` the following entries:

$$ext^+(d) : \left(\phi(d), \phi^* \left(tpr\big(d\big), fpr\big(core(d)\big)\right), \left(|ext(d)| - |ext(core^+(d))|\right)/(2 \cdot |\mathcal{G}|)\right)$$

The *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ are then computed at the end by a single pass on the entries of `BoundPerPosExt` using Theorems 1 and 2.

Update the bounds after adding a new cut-point (line 8). In order to compute the new *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ which decrease according to theorems 1 and 2, one need to add/update some entries in the structure `BoundPerPosExt`. For that, only two types of patterns should be looked for:

1. The new `cotp` patterns mined by `RefineAndMine`, that is those which left or right bound on attribute *attr* is the added value *cut*. Visiting these patterns will add potentially new entries in `BoundPerPosExt` or update ancient ones.
2. The old `cotp` which core changes (i.e. becomes less restrictive) in the new discretization. One can show that these patterns are those which left bound is $prev(cut, C^{cur}_{attr})$ or right bound is $next(cut, C^{cur}_{attr})$ on attribute *attr*. Visiting these patterns will only update ancient entries of `BoundPerPosExt` by potentially decreasing both second and third value.

Adding a new cut-point (line 7). We have implemented for now a strategy which aims to decrease the $\overline{accuracy_\phi}$. For that, we search in `BoundPerPosExt` for the description d having the maximal value $\phi^*\big(tpr\big(d\big), fpr\big(core(d)\big)\big)$. In order to decrease $\overline{accuracy_\phi}$, we increase the size of $core(d)$ (to potentially increase $fpr\big(core(d)\big)$). This is equivalent to choose a cut-point in the border region $C^{rel}_{attr} \backslash C^{cur}_{attr}$ for some attribute *attr* such that $cut \in d.I_{attr} \backslash core(d).I_{attr}$. Consider that we are in the step where the current discretization C^{cur} is the one depicted in Fig. 2. Imagine that the bottom pattern $\bot = \mathbb{R}^2$ is the one associated to the maximal value $\phi^*\big(tpr\big(\bot\big), fpr\big(core(\bot)\big)\big)$. The new cut-point should be chosen in $\{4, 5\}$ for $attr = 1$ (recall that $core(\bot) = (-\infty, 3) \times (-\infty, +\infty)$). Note that if for such description there is no remaining relevant cut in its *border regions* for all $attr \in \{1, ..., p\}$ then $core(d) = d$ ensuring that d is the *top pattern*.

6 Empirical Study

In this section we report quantitative experiments over the implemented algorithms. For reproducibility purpose, the source code is made available in our companion page[10] which also provide a wider set of experiments. Experiments were carried out on a variety of datasets (Tab. 1) involving ordinal or continuous numerical attributes from the UCI repository.

Dataset	num	rows	intervals	class	α	Dataset	num	rows	intervals	class	α
ABALONE_02_M	2	4177	56×10^6	M	0.37	GLASS_02_1	2	214	161×10^6	1	0.33
ABALONE_03_M	3	4177	74×10^9	M	0.37	GLASS_04_1	4	214	5×10^{15}	1	0.33
CREDITA_02_+	2	666	1×10^9	+	0.45	HABERMAN_03_2	3	306	47×10^6	2	0.26
CREDITA_04_+	4	666	3×10^{15}	+							

Table 1: Benchmark datasets and their characteristics: number of numerical attributes, number of rows, number of all possible intervals, the considered class and its prevalence

[10] Companion page: `https://github.com/Adnene93/RefineAndMine`

First, we study the effectiveness of `RefineAndMine` in terms of the speed of convergence to the optimal solution, as well as regarding the evolution over time of the accuracy of the provided bounding quality's guarantee. To this end, we report in Fig. 4, the behavior of `RefineAndMine` (i.e. quality and bounding guarantee) according to the execution time to evaluate the time/quality trade-off of the devised approach. $\overline{accuracy}$ as presented in Theorem 1 is the difference between the quality and its bounding measure. The experiments were conducted by running both `RefineAndMine` and the exhaustive enumeration algorithm (`MinIntChange` performed considering $\mathcal{D}_{C^{rel}}$) on the benchmark datasets using *informedness* measure. The exhaustive algorithm execution time enables the estimation of the computational overhead incurred by `RefineAndMine`. We interrupt a method if its execution time exceeds two hours. Note that, in the experiments, we choose to disable the computation of specificity since the latter is only optional and does not affect the effectiveness of the algorithm. This in contrast to the quality bound computation which is essential as it guides `RefineAndMine` in the cut-points selection strategy. The experiments give evidence of the effectiveness of `RefineAndMine` both in terms of finding the optimal solution as well as in providing stringent bound on the top quality pattern in a prompt manner. Two important milestones achieved by `RefineAndMine` during its execution are highlighted in Fig. 4. The first one, illustrated by the *green dotted line*, points out the required time to find the best pattern. The second milestone (*purple line*) is reached when the quality's and the bound's curves meet, this ensures that the best quality was already found by `RefineAndMine`. Interestingly, we observe that for most configurations the second milestone is attained by `RefineAndMine` promptly and well before the exhaustive method termination time. This is explained by the fact that the adopted cut points selection strategy aims to decrease as early as possible the $\overline{accuracy}$ metric. Finally, `RefineAndMine` requires in average 2 times of the requested execution time (*red dotted line*) by the exhaustive algorithm. This overhead is mostly incurred by the quality guarantee computation.

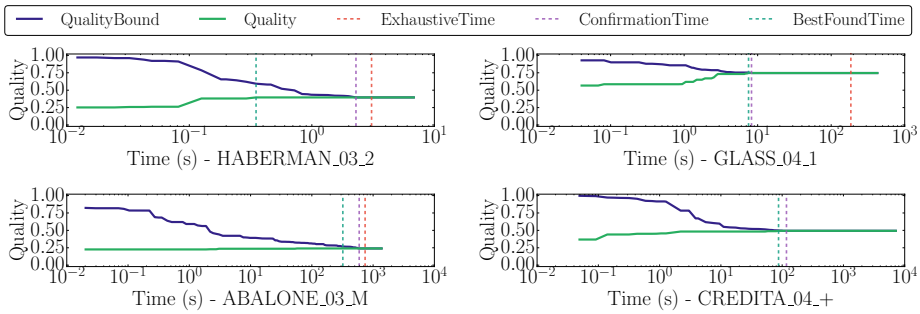

Fig. 4: Evolution over time of top pattern quality and its bounding guarantee provided by `RefineAndMine`. Execution time is reported in log scale. The last figure reports that the exhaustive enumeration algorithm was not able to finish within 2 hours

We illustrate in Fig. 5 the behavior of `RefineAndMine` in terms of finding diverse set of high quality patterns covering different parts of the dataset. To evaluate how quickly the devised approach finds a diverse patterns set, we run the exhaustive approach over the benchmark datasets to constitute a top-k diverse patterns set heuristically as following: the patterns extracted by the exhaustive search algorithm are sorted according to the quality measure and the best pattern is kept in the returned top-k list. Next, the complete patterns list are iterated over, and the top-k list is augmented by a pattern if and only if its similarity with all the patterns of the current content of the top-k list is lower than a given threshold (a Jaccard index between extents). This process is interrupted if the desired number of patterns of the top-k list is reached or no remaining dissimilar pattern is available. Similar post-processing techniques were used by [20,5]. Once this ground truth top-k list is constituted over some benchmark dataset, we run `RefineAndMine` and measure the *specificity quantity* of the obtained results set Sol with the top-k list. *specificity* metric is rewritten in eq. 3 to accommodate the desired evaluation objective of these experiments. Still, it remains upper-bounded by the general formula of $\overline{specificity}$ given in Theorem 2. This in order to evaluate at what extent the visited patterns by `RefineAndMine` well-cover the ground-truth patterns which are scattered over different parts of some input dataset. We report in Fig. 5 both *specificity* and its bounding guarantee $\overline{specificity}$, as well as, a diversity metric defined in eq. 4. Such a metric was defined in [5] to evaluate the ability of an approximate algorithm to retrieve a given ground-truth (i.e. diversified top-k discriminant patterns set). This diversity metric relies on a similarity rather than a distance (as in specificity), and is equal to 1 when all patterns of the top-k list are fully discovered.

$$specificity(\text{top-k}, Sol) = \sup_{d \in \text{top-k}} \inf_{c \in Sol} (|ext(d) \,\Delta\, ext(c)|/|\mathcal{G}|) \tag{3}$$

$$diversity(\text{top-k}, Sol) = \underset{d \in \text{top-k}}{avg} \sup_{c \in Sol} (Jaccard(ext(d), ext(c))) \tag{4}$$

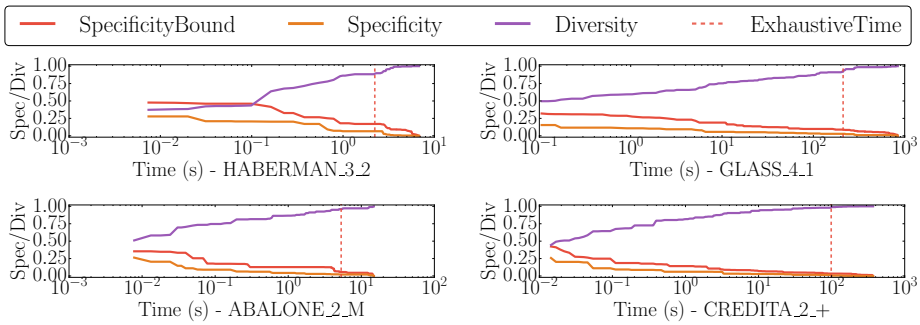

Fig. 5: Efficiency of `RefineAndMine` in terms of retrieving a diverse patterns set. Execution time is reported in log scale. The ground-truth for each benchmark dataset corresponds to the obtained Top10 diversified patterns set with a similarity threshold of 0.25 and a minimum *tpr* of 15% .

In most configurations, we notice that `RefineAndMine` is able to uncover approximately 80% (given by *diversity*) of the ground truth's patterns in less than 20% of the time required by the exhaustive search algorithm. For instance, in ABALONE_02_M, we observe that after 2 seconds (12% of the required time for the exhaustive algorithm), the patterns outputted by `RefineAndMine` approximate 92% of the ground truth. Moreover, we observe that the specificity and $\overline{specificity}$ decrease quickly with time, guaranteeing a high level of diversity.

For a comparative study, we choose to compare `RefineAndMine` with the closest approach following the same paradigm (anytime) in the literature, that is the recent `MCTS4DM` technique [5]. `MCTS4DM` is depicted by the authors as an algorithm which enables the anytime discovery of a diverse patterns set of high quality. While `MCTS4DM` ensures interruptibility and an exhaustive exploration if given enough time and memory budget, it does not ensures any theoretical guarantees on the distance from optimality and on the diversity. We report in Fig. 6 a comparative evaluation between the two techniques. To realize this study, we investigate the ability of the two methods in retrieving the ground truth patterns, this by evaluating the quality of their respective diversified top-k lists against the ground truth using the diversity metric (eq. 4). We observe that `RefineAndMine` outperforms `MCTS4DM` both in terms of finding the best pattern, and of uncovering diverse patterns set of high qualities. This is partially due to the fact that our method is specifically tailored for mining discriminant patterns in numerical data, in contrast to `MCTS4DM` which is agnostic of the interestingness measure and the description language. Note that, to enable a fair comparison of the two approaches, we report the full time spent by the methods including the overhead induced by the post-computation of the diversified top-k patterns set.

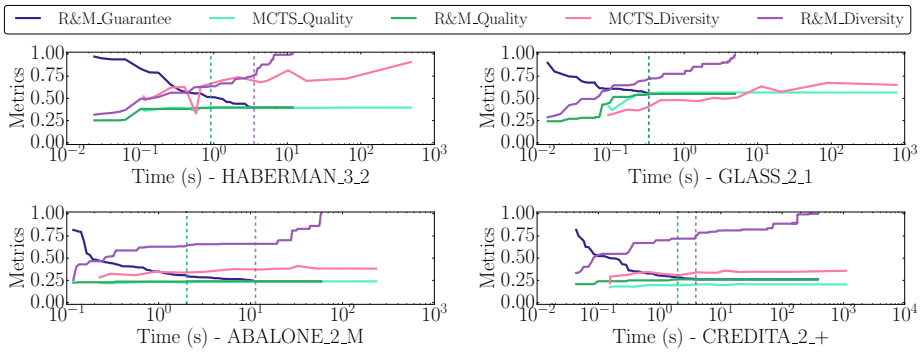

Fig. 6: Comparative experiments between `RefineAndMine` ($R\&M$) and `MCTS4DM`. Execution time is reported in log scale. The ground-truth for each benchmark dataset corresponds to the obtained Top10 diversified patterns set with a similarity threshold of 0.25 and no minimum support size threshold.

7 Discussions and Conclusion

We introduced a novel anytime pattern mining technique for uncovering discriminant patterns in numerical data. We took a close look to discriminant interestingness measures to focus on hyper-rectangles in the dataset fostering the presence of some class. By leveraging the properties of the quality measures, we defined a guarantee on the accuracy of `RefineAndMine` in approximating the optimal solution which improves over time. We also presented a guarantee on the specificity of `RefineAndMine` –which is agnostic of the quality measure– ensuring its diversity and completeness. Empirical evaluation gives evidence of the effectiveness both in terms of finding the optimal solution (w.r.t. the quality measure ϕ) and revealing local optimas located in different parts of the data.

This work paves the way for many improvements. `RefineAndMine` can be initialized with more sophisticated discretization techniques [19,9]. We have to investigate additional cut-points selection strategies. While we considered here discriminant pattern mining, the enumeration process (i.e. *successive refinement of discretizations*) can be tailored to various other quality measures in subgroup discovery. For example, the accuracy bound guarantee definition can be extended to handle several other traditional measures such as Mutual Information, χ^2 and *Gini split* by exploiting their (quasi)-convexity properties w.r.t. *tpr* and fpr variables [24,1]. Other improvements include the adaptation of `RefineAndMine` for high-dimensional datasets and its generalization for handling additional types of attributes (categorical, itemsets, etc.). The latter is facilitated by the generic notions from Section 5 and the recent works of Buzmakov et al. [6].

Aknowledgement. This work has been partially supported by the project *ContentCheck* **ANR-15-CE23-0025** funded by the French National Research Agency, the Association Nationale Recherche Technologie (**ANRt**) French program and the **APRC Conf Pap - CNRS** project. The authors would like to thank the reviewers for their valuable remarks. They also warmly thank Loïc Cerf, Marc Plantevit and Anes Bendimerad for interesting discussions.

References

1. Abudawood, T., Flach, P.A.: Evaluation measures for multi-class subgroup discovery. In: ECML PKDD. pp. 35–50. Springer (2009)
2. Atzmueller, M., Puppe, F.: Sd-map–a fast algorithm for exhaustive subgroup discovery. In: PKDD. pp. 6–17. Springer (2006)
3. Boley, M., Lucchese, C., Paurat, D., Gärtner, T.: Direct local pattern sampling by efficient two-step random procedures. In: KDD. pp. 582–590 (2011)
4. Boley, M., Moens, S., Gärtner, T.: Linear space direct pattern sampling using coupling from the past. In: KDD. pp. 69–77 (2012)
5. Bosc, G., Boulicaut, J., Raïssi, C., Kaytoue, M.: Anytime discovery of a diverse set of patterns with monte carlo tree search. DMKD **32**(3), 604–650 (2018)
6. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Fast generation of best interval patterns for nonmonotonic constraints. In: ECML PKDD. pp. 157–172. Springer (2015)

7. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Revisiting pattern structure projections. In: Formal Concept Analysis. pp. 200–215. Springer (2015)
8. Denecke, K., Wismath, S.L.: Galois connections and complete sublattices. In: Galois Connections and Applications, pp. 211–229. Springer (2004)
9. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI. pp. 1022–1029 (1993)
10. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: ICCS 2001. LNCS (2120). 129–142.
11. Garriga, G.C., Kralj, P., Lavrac, N.: Closed sets for labeled data. Journal of Machine Learning Research **9**, 559–580 (2008)
12. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. **38**(3), 9 (2006)
13. Giacometti, A., Soulet, A.: Dense neighborhood pattern sampling in numerical data. In: SIAM. pp. 756–764 (2018)
14. Grosskreutz, H., Rüping, S.: On subgroup discovery in numerical domains. Data Min. Knowl. Discov. **19**(2), 210–226 (2009)
15. Guyet, T., Quiniou, R., Masson, V.: Mining relevant interval rules. CoRR **abs/1709.03267** (2017), `http://arxiv.org/abs/1709.03267`
16. Hu, Q., Imielinski, T.: Alpine: Progressive itemset mining with definite guarantees. In: SIAM. pp. 63–71 (2017)
17. Huttenlocher, D.P., Klanderman, G.A., Rucklidge, W.: Comparing images using the hausdorff distance. IEEE Trans. Pattern Anal. Mach. Intell. **15**(9), 850–863 (1993)
18. Kaytoue, M., Kuznetsov, S.O., Napoli, A.: Revisiting Numerical Pattern Mining with Formal Concept Analysis. In: IJCAI. pp. 1342–1347 (2011)
19. Kurgan, L., Cios, K.J.: Discretization algorithm that uses class-attribute interdependence maximization. In: IC-AI. pp. 980–987 (2001)
20. van Leeuwen, M., Knobbe, A.J.: Diverse subgroup set discovery. Data Min. Knowl. Discov. **25**(2), 208–242 (2012)
21. Lenca, P., Meyer, P., Vaillant, B., Lallich, S.: On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. European Journal of Operational Research **184**(2), 610–626 (2008)
22. Lucas, T., Silva, T.C.P.B., Vimieiro, R., Ludermir, T.B.: A new evolutionary algorithm for mining top-k discriminative patterns in high dimensional data. Appl. Soft Comput. **59**, 487–499 (2017)
23. Mampaey, M., Nijssen, S., Feelders, A., Knobbe, A.J.: Efficient algorithms for finding richer subgroup descriptions in numeric and nominal data. In: ICDM. pp. 499–508 (2012)
24. Morishita, S., Sese, J.: Traversing itemset lattice with statistical metric pruning. In: ACM SIGMOD-SIGACT-SIGART. pp. 226–236 (2000)
25. Pawlak, Z.: Rough sets. International Journal of Parallel Programming **11**(5), 341–356 (1982)
26. Roman, S.: Lattices and Ordered Sets. Springer New York (2008)
27. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: PKDD. pp. 78–87 (1997)
28. Yang, Y., Webb, G.I., Wu, X.: Discretization methods. In: Data Mining and Knowledge Discovery Handbook, 2nd ed., pp. 101–116. Springer (2010)
29. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine **17**(3), 73–83 (1996)