N°d'ordre NNT : 2020LYSEI080

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA de Lyon

Ecole Doctorale N° 512
Mathématiques et Informatique (InfoMaths)

Spécialité/ discipline de doctorat :
Informatique

Soutenue publiquement le 29/09/2020, par :
Romain Mathonat

Rule Discovery in Labeled Sequential Data: Application to Game Analytics

---

Devant le jury composé de :

Atzmüller, Martin	Professeur, Osnabrueck University	Rapporteur
Termier, Alexandre	Professeur, Université Rennes 1	Rapporteur
Amer-Yahia, Sihem	Directrice de recherche, CNRS	Examinatrice
Forestier, Germain	Professeur, Université de Haute-Alsace	Examinateur
Laurent, Anne	Professeure, Université de Montpellier	Examinatrice
Raïssi, Chedy	Chargé de recherche, INRIA et UBISOFT	Examinateur
Boulicaut, Jean-François	Professeur, INSA-LYON	Directeur de thèse
Kaytoue, Mehdi	Maître de conférences HDR, INSA-LYON et INFOLOGIC R&D	Co-directeur de thèse

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	**M. Stéphane DANIELE** Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	**M. Gérard SCORLETTI** École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES	**M. Philippe NORMAND** UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Curien - 3ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tel : +33(0)4 72 44 82 32 **sylvie.ricard-blum@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 **infomaths@univ-lyon1.fr**	**M. Hamamache KHEDDOUCI** Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tel : 04.72.44.83.69 **hamamache.kheddouci@univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **ed.materiaux@insa-lyon.fr**	**M. Jean-Yves BUFFIÈRE** INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **mega@insa-lyon.fr**	**M. Jocelyn BONJOUR** INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **veronique.cervantes@univ-lyon2.fr**	**M. Christian MONTES** Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Je remercie Alexandre Termier, Professeur à l'Université de Rennes et Martin Atzmüller, Professeur à l'université d'Osnabrueck, d'avoir accepté le rôle de rapporteur de ma thèse, ainsi que leur travail de lecture approfondi et intéressé.

Je remercie également Sihem Amer-Yahia, directrice de recherche CNRS, Germain Forestier, Professeur à l'Université de Haute-Alsace, Anne Laurent, Professeur à l'Université de Montpellier, et Chedy Raïssi, chargé de recherche à l'INRIA et directeur du pôle Science des Données chez Ubisoft, pour leur participation au jury de thèse, ainsi que l'intérêt porté à ce travail de trois années.

Je remercie vivement l'entreprise Atos qui m'a accueillie durant ces trois années afin d'appliquer le fruit de mes recherches à des cas d'études industriels concrets, et en particulier Christophe Boulard pour sa bienveillance.

Concernant mon encadrement scientifique je remercie tout d'abord Jean-François Boulicaut. Premièrement, pour avoir accepté de travailler avec un jeune un peu taquin et obstiné. Deuxièmement, pour m'avoir fait confiance et partagé tes expériences afin de me guider dans cette expérience si particulière qu'est la thèse. Enfin, et de manière plus générale, merci pour ta bienveillance, ta franchise, ta sensibilité et ton soutien. Je remercie ensuite Mehdi Kaytoue, pour ton encadrement scientifique pointu qui m'a aidé à me sortir de situations délicates. Merci de m'avoir apporté une autre vision qui m'a permis d'acquérir une certaine prise de recul vis-à-vis de mon travail et de la science en général. Merci aussi pour ta confiance, ta considération à mon égard, et la possibilité que tu m'as accordée de pouvoir arbitrer mes propres choix. Enfin merci de m'avoir proposé une nouvelle aventure, où nous allons continuer de travailler ensemble.

Je remercie tous les doctorant(e)s et docteur(e)s que j'ai rencontrés, avec qui les échanges ont très souvent été enrichissants. Je remercie en particulier Mohamed et Tarek qui très tôt dans la thèse m'ont permis de prendre du recul vis-à-vis de celle-ci. Je salue aussi Adnane, Aimene, Alexandre, Anes, Corentin, Marie et tous les autres.

J'embrasse mes amis proches pour leur soutien et leur amitié : Nicolas, Pauline, Yoan, Aurélie, Guillaume, rencontrés à l'INSA, Morgan, Alexis, Lucas, Pierre, rencontrés bien avant, parmi tant d'autres.

Bien sûr je remercie toute ma famille, qui m'a toujours supporté, et soutenu, en particulier mes parents, qui m'ont apporté beaucoup d'amour. Merci du fond du coeur.

La thèse a été une expérience très riche, qui aura probablement eu une influence positive sur la personne que je suis devenu, mais elle est aujourd'hui officiellement terminée. Cependant il y a un évènement important qui restera et a eu lieu durant cette thèse, c'est la rencontre avec une personne formidable, qui m'a soutenu et supporté au jour le jour ces dernières années. Cette personne c'est Diana, ma chère Diana, je te le dis avec mon plus bel accent russe : я тебя люблю.

Enfin je tiens à remercier ma merveilleuse grand-mère Marguerite, qui m'a encouragé à me lancer dans cette aventure. Mon seul regret aura été que tu ne sois plus là aujourd'hui pour que nous fêtions cet évènement ensemble.

Au fond, on ne sait que lorsqu'on sait peu. Avec le savoir croît le doute.
Johann Wolfgang von Goethe

Contents

Chapter 1

Introduction

1.1 Context

This thesis has been completed in collaboration with Atos, an IT services and consulting company. One of the specificity of the work with such a company in terms of research projects is that interesting use cases often originate externally, coming from Atos' clients. Nevertheless, Atos aims at capitalizing on data science methods, knowledge and results, in order to potentially apply the developed solutions to other clients. This exhibits the requirement for genericity of methods we were going to propose. To do so, we first had to prospect the needs of different industrial clients, in order to identify similar needs to work on. Two important points have been isolated.

First, it was clear that there is a need for a better understanding of existing systems. Indeed, whether it is for a door opening system on trains, or the cloud environment maintenance of a company, clients want to understand the cause of different events. Particularly, they want to isolate causes of failures. Second, they also want to be able to predict those failures, but without using black box models: they want their tools to provide *interpretable* predictions. Indeed, domain experts already predict some failures, but they want their expertise to be boosted, instead of being replaced. They want to validate proposed predictions, with justification, with a human in the loop, whether it is due to the lack of trust in "AI" systems, or legal constraints, or simply to improve system reliability.

A promising approach to tackle these two problems is the use of pattern mining [54]. Roughly speaking, it consists in exploring and extracting interesting patterns, i.e., descriptions, from data. The language for descriptions is, by design, readable and interpretable by analysts and data owners, which makes this approach appealing. Moreover, using relevant *discriminative quality measures* to assess patterns quality will help us find patterns both explaining and predicting a given class.

In this industrial context, the main data types we encountered are *sequences* and *time series*: systems evolve over time, and at some point an event fires. The prediction of failures in a cloud environment offers an illustrative example of such an industrial use case. Data generated by such a system are sequences of events, and can be labeled with failures, i.e., presence or absence of a breakdown. Applying *classification techniques* helps answering to the question: "*will* a failure event occur?" (e.g., [131]), while applying *sequential event prediction* helps determining "*what* is the next event to occur?" (e.g., [81]). Nevertheless, another need is to

explain, or at least, to provide hypotheses on the *why*. Given data sequences, labeled with classes, we aim at automatically finding discriminative patterns for these classes. Considering this cloud environment example, the goal would be to compute patterns "that tend to occur with breakdowns". Such patterns provide valuable hypotheses for a better understanding of the target system. Once validated by domain experts, the patterns can then be used to support maintenance planning tasks.

In this Thesis, due to confidentiality requirements, we are not going to focus on Atos clients use cases. Note also that our work on urban farming (growing plants in controlled environment) with a client of Atos is an exception, and resulted in a publication [91] that we are not going to detail here. In what follows, we will present algorithms to tackle generic problems. However, in order to assess the relevancy of proposed approaches, we will introduce a difficult game analytics use case based on the game Rocket League[1]. This choice has been motivated by my interest in this use case, the fact that I have an expertise in this game, and that the DM2L research team of LIRIS lab in which I am working already has an expertise in game analytics domain [23], [29], [30]. We will show that we can collect and process data in time series format, and that we can use them to extract interesting patterns characteristic of players behaviour in-game. Those patterns can then be used to classify those behaviours in *real-time*, which is of interest to different actors: players, analysts, game editor and e-sport teams.

1.2 Supervised rule discovery

The general problem we address in this Thesis is to find interpretable rules extracted from a dataset, composed of conjunctions of restrictions on different variables, in order to extract knowledge for a variable of interest. This can be illustrated by the following simplistic example. Suppose we are looking for lung cancer risk factors, i.e., the rules such as:

$$age \geq 72 \wedge smoke = True \rightarrow LungCancer = True$$

Being able to extract such hypotheses from data is of great value to present to experts for further validation. Moreover, as we will explain in more details in the following, we are interested in finding *the set of the best non-redundant rules*. Informally, two rules are non-redundant if the sets of data they appear in are disjoint enough. Our motivation is twofold. First, extracting redundant rules often comes with extracting too much of them, each being a small variation of another. This confuses the end user and decreases her trust in the method. Second, it helps reducing discovery of false hypotheses.

Indeed, by construction, keeping only non-redundant rules will keep only those of highest quality, removing those which are similar, i.e., covering a similar set of data. As a fictional example, let us assume that we find rules $smoke = True \rightarrow LungCancer = True$ and $Sport = False \rightarrow LungCancer = True$. In general, people practicing sport do not smoke (or smoke less than no-sport practitioners). If the (hypothetical) ground truth is that it is *smoking* that causes lung cancer, the rule $Sport = False \rightarrow LungCancer = True$ is less interesting, and could lead to a false hypothesis. Indeed, practicing sport would not decrease lung cancer chances per se, but is negatively co-occurring with smoking, leading to the discovery of this rule which is not really isolating the true reason. Under hypotheses that data are of correct quality, and that rule quality can be assessed by a good measure, i.e., it will foster the rule $smoke = True \rightarrow LungCancer = True$, the non-redundancy will help removing this kind of false hypotheses.

[1] `https://www.rocketleague.com/`

Table 1.1: Results of success of treatment A and treatment B, discerning cases on calculi size

	Treatment A	Treatment B
Calculi ≤2cm	81/87 = **93%**	234/270 = 87%
Calculi > 2cm	192/263 = **73%**	55/80 = 69%
Total	273/350 = 78%	289/350 = **83%**

Another important component of knowledge discovery is the place of the domain expert. Some methods are purely based on data, and one could think of it as a replacement for the expert. However, in our work, we consider proposed methods not as a replacement, but as a tool to boost expert knowledge. This can be illustrated through the example of the Simpson paradox [118], on a case of renal calculi treatment [31], with the analysis inspired by [83]. The aim of this study was to evaluate the impact of different treatments on renal calculi. For simplification, let us consider only two among four experimented treatments, that we are going to call "Treatment A" for open surgery and "Treatment B" for percutaneous nephrolithotomy. If we do not consider expert knowledge, data tell us that Treatment A was successful in 78% of cases, and Treatment B in 83% of cases. Those results, of course, would make us recommend Treatment B, which gave better results. However, incorporating domain expert knowledge in the analysis would have helped us avoiding an important bias. Indeed, the experimental studies were performed on different groups, introducing a bias: depending on the size of calculi, methods were applied in different proportions. Detailed results are given in Table. 1.1.

As we can see, if we consider a more precise data description, i.e., considering the case of calculi of size greater or less than two centimeters, we can see that Treatment A is the best for the first case, with 93% of success. However, it is also the best for the second case. We then have two local descriptive rules showing that A is the best treatment, whereas globally, the B treatment seems to give better results. This paradox can be explained by the following: treatment A was more often applied in the case of calculi measuring more than two centimeters. Chances of success were clearly less in this case, whatever the treatment was, leading to a worse metric for treatment A: it is easy to make a treatment look better than it really is by "cherry picking" cases to which to apply, selecting the easier ones.

Note that when considering very specific subsets of data, one can imagine always finding very local rules that would say the contrary of the reality of the underlying phenomena, whether due to statistical randomness, or incorrectness with the method used to collect data. This problem leads to the question of isolating causality. This is not the subject of this Thesis, but interested reader can refer to the work of Judea Pearl on causality [102].

Due to this very complex problematic, methods proposed in this Thesis are not claiming to directly isolate causes, but to propose hypotheses extracted from data, that would require further studies with expert knowledge in order to validate them.

1.3 Contributions

1.3.1 Bandit model and Monte Carlo tree search for supervised rule discovery in sequences of itemsets

Labeled sequential data are ubiquitous. This makes supervised rule discovery applicable to many application domains, for instance, text or video data analysis [98], industrial process supervision

[133], biomedical genomics sequential data analysis [116], web usage mining [105], video game analytics [23], etc.

The problem of finding interesting rules in sequences of itemsets has attracted few attention. In particular, methods are often focusing on sequences of items, or are dependent on the quality measure used to assess rule qualities. Existing methods also often focus either on exploration of search space, i.e., sampling methods, or purely on exploitation, i.e., hill-climbing-like methods. We propose two algorithms, namely `SeqScout` based on a multi-armed bandit model, and `MCTSExtent`, based on the Monte Carlo Tree Search [25]. They present several advantages: they are independent of the used quality measure, they use a trade-off between exploration and exploitation, they are quite simple and perform well. Note that their adaptations to our problems explore, contrary to most of existing methods, the search space of patterns extent, i.e., of objects covered by rules, in a bottom-up way, whereas traditional approaches explore the search space of patterns in a top-down way, from the most general description towards specific data objects. Another important property is that all generated and explored patterns cover at least one dataset object. This is important when dealing with sequences of itemsets, as the size of the search space becomes quickly gigantic, and that it contains a large majority of uninteresting patterns that do not cover any object.

1.3.2 Monte Carlo tree search for supervised rule discovery in high dimensional numerical data

Let us consider the video game data analytics domain. The video game industry generated 134.9 billions of dollars in 2018, a number that keeps growing each year [2], making it an impactful and interesting domain to work on. Game activities usually provide sequences of time-tagged data points, i.e., *time series*. One interesting challenge is then to design efficient and interpretable prediction models on time series for a variety of tasks (e.g., predicting player behaviour for game customization or ad placement, user profiling for adaptive skill mastery training, etc.). We proposed to address the problem of supervised rule discovery in time series as the challenging problem of *supervised rule discovery in high-dimensional numerical data*. Indeed, we can transform the original time series into potentially high-dimensional numerical data, and then take advantage of the numerical pattern interpretability. We then proposed an algorithm called `MonteCloPi`, inspired from `MCTSExtent`, which explores only *closed on the positives* patterns (COTP) [52] [16] reducing the search space size. It consists in a Monte Carlo Tree Search on the search space of patterns extents in a bottom-up way, with a property of *completeness* on closed on the positives if given enough time and if dataset times series have the same length. Moreover, we showed that integrating such an approach to classify time series presents an advantage of being able to classify them in real time, contrary to kNN DTW [94], even with time series of varying lengths.

1.3.3 Application to player behaviour detection in game analytics

In the context of online games, or e-sport, understanding player behaviour is important for several actors. First, for game creators, collecting data from the use of their product and analyzing them helps to better understand the players, and then how to better act to improve the game. Secondly, it is interesting for players and e-sport team in order to understand players

[2] `https://en.wikipedia.org/wiki/Video_game_industry`

weaknesses and strengths, to optimize their training, or to better anticipate opponents strategies and play-styles in future matches. Finally, as e-sport matches have commented games with public, it is propitious to improve game analysis to deliver a better commentary and to better explain to the public what is happening. Indeed, when watching professionals playing, whether in traditional sport or e-sport, it can be difficult for an inexperienced spectator to understand all rules, strategies, mindsets and plays of players. Commentators and analysts are here to entertain, but also to explain the different degrees of depths in matches to public, so helping them in this task is of interest. In particular, here, we focused on the game Rocket League.

Basically, Rocket League is a game where players control a car on a footbal field. Those controls are very precise, and various: turn left, right, jump, double jump, boost, rotate in the air, etc. Interestingly, matches replays can be stored and de-compiled, so that each match's information can be extracted: players positions, velocity, accelerations, etc, at each game timestamp. Furthermore, we can collect player inputs to enrich those data. The problem we proposed to tackle is the automatic discovery and classification of players figures, called "skillshots". Indeed, similarly to traditional sport, players can perform particular and characteristic figures to take advantage on their opponent, like the nutmeg in football, for example. One of the challenges of this problem is that data contain a lot of noise: as controls of the car are very precise, two instances of the same skillshot will always be different, even for the same player. In addition to this noise in player sequences of inputs, contextual information in matches replay also vary: the ball position and speed will change, as the one of the player etc. Moreover, it is important to note that skillshots are created by the community, so they evolve through time, and they can be very complex to understand and perform.

We proposed a method to deal with this problem, using supervised rule discovery for sequences. We implemented the full workflow of proposed methodology, from skillshots definition to skillshot classification, generating, pre-processing and formatting data by our means, making the resulting dataset open to the community. We showed that proposed workflow gave very good results, and proposed ways of improvement to bring this kind of method to game analytics state-of-the-art.

Note that there is a parallel between detecting those skillshots and detecting failures in a cloud environment. Indeed, scoring a goal constitutes a generic event, which we then classify to know what "type of goal" it is, just like a occurring failure constitutes a generic event, which we could classify to know the type of failure we are dealing with.

1.4 Structure of the thesis

This Thesis is organized as follows. Chapter 2 provides the background of pattern mining and formally defines the supervised rule discovery task. Chapter 3 introduces the multi-arms bandit problem with one of its solutions, the Upper Confidence Bound (UCB) strategy, and its evolution, the Monte Carlo Tree Search (MCTS). Those are the methods from game theory that we are going to exploit in our algorithms to tackle our supervised rule discovery tasks. Chapter 4 presents our first contribution, tackling the problem of supervised rule discovery for sequences of itemsets. We propose to adapt the UCB strategy and the MCTS to this problem, exploring the search space of extents of rules in a bottom-up way, and present an empirical study on several datasets to assess the validity of proposed methods. This contribution has been partially published in the proceedings of the French conference *Extraction et Gestion de Connaissances EGC'19* [87], a more mature approach has been published in the proceedings of the 6th IEEE

International Conference on Data Science and Advanced Analytics DSAA 2019 [89], and an extended version is currently under review for the journal Knowledge and Information Systems KAIS. Chapter 5 proposes a solution to address the problem of supervised rule discovery for high dimensional numerical data and time series. We also used an MCTS approach on the search space of pattern extent in a bottom-up way, but we take advantage of the notion of closed on the positive patterns here. We proceed to several batches of experiments to assess the validity of this method. A paper describing this contribution is currently under review for a major data mining conference. Chapter 6 presents our use case on the video game Rocket League, with the goal of classifying users behaviour in real time using their controller inputs as well as contextual matches information contained in replay files. We present a complete workflow that we implemented, from the data collection task to the prediction, using expert knowledge and supervised rule discovery. This application has been published in the proceeding of the 2020 IEEE Conference on Games COG 2020 [88]. Finally, Chapter 7 concludes the Thesis and gives several perspectives for further research.

1.5 List of publications

Peer-reviewed international conferences with proceedings:

- **Romain Mathonat**, Jean-François Boulicaut, Mehdi Kaytoue: *A behavioral pattern mining approach to model players skills in Rocket League.* In 2020 IEEE Conference on Games, COG 2020, Kindai University, Osaka, Japan, August 2020, 8 pages
- Alexandre Millot, **Romain Mathonat**, Rémy Cazabet, Jean-François Boulicaut: *Actionable Subgroup Discovery and Urban Farm Optimization.* In Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings: 339-351
- **Romain Mathonat**, Diana Nurbakova, Jean-François Boulicaut, Mehdi Kaytoue: *SeqScout: Using a Bandit Model to Discover Interesting Subgroups in Labeled Sequences.* In 2019 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2019, Washington, DC, USA, October 5-8, 2019: 81-90

Peer-reviewed international workshops:

- **Romain Mathonat**, Jean-Francois Boulicaut and Mehdi Kaytoue: *A Behavioral Pattern Mining Approach to Model Player Skills in Rocket League.* In the Proceeding of the 7th Workshop on Machine Learning and Data Mining for Sports Analytics co-located with 2020 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2020), Ghent, Belgium, September 14-18, 2020.

Peer-reviewed French national conferences:

- **Romain Mathonat**, Jean-François Boulicaut, Mehdi Kaytoue: *Découverte de sous-groupes à partir de données séquentielles par échantillonnage et optimisation locale.* In Extraction et Gestion des connaissances, EGC 2019, Metz, France, January 21-25, 2019: 153-164

The following works are currently undergoing the **reviewing process:**

International journals:

- **Romain Mathonat**, Diana Nurbakova, Jean-François Boulicaut, Mehdi Kaytoue: *Anytime mining of sequential discriminative patterns in labeled sequences*, In Knowledge and Information Systems Journal. Submited revision under evaluation, 2020.

Peer-reviewed international conferences with proceedings:

- **Romain Mathonat**, Diana Nurbakova, Jean-François Boulicaut, Mehdi Kaytoue: *Anytime subgroup discovery in high dimensional numerical data*, In SIAM International Conference on Data Mining, SDM 2021

Chapter 2

Supervised Rule Discovery

This chapter aims at introducing supervised rule discovery as well as main definitions we are going to use in this thesis. In Section 2.1, we introduce the pattern mining task with a pioneer work of data mining, that influenced the field and inspired further problem formalizations and resolutions. In Section 2.2 we present definitions and we formalize the generic problem of this thesis. Then, in Section 2.3, we present general strategies used to solve instances of the problem presented in the previous section. Finally, in Section 2.4 we instantiate this generic problem to sequences of itemsets, and present existing works on it.

2.1 Introducing pattern mining task

In this section we introduce the pattern mining task, and illustrate it with the example of frequent itemset mining, introduced by Agrawal et al. in [1]. Here, we refer to this pioneering work of the domain in order to facilitate the understanding of the problems tackled in this thesis.

2.1.1 A simple formalisation

The general pattern mining task can be formalized as follows [86]:

$$Th(\mathcal{S}, \mathcal{D}, q) = \{p \in \mathcal{S} \mid q(p, \mathcal{D}) \text{ is } true\}$$

where $\mathcal{D}$ is a database of objects, $\mathcal{S}$ is the search space of a pattern description language, and q is a selection predicate. S is the set of all possible patterns that can cover database objects. The selection predicate is a boolean function indicating if a pattern p satisfies a constraint (or a combination of constraints).

Definition 1 (Covering relation) *For each pattern mining task, there is a relation between any pattern p from the pattern description language $\mathcal{S}$, and objects $o \in \mathcal{D}$. This covering relation is of the form $f : \mathcal{D} \times \mathcal{S} \to \{false, true\}$. We will denote it $covers(p, o)$ with $p \in \mathcal{S}$ and $o \in \mathcal{D}$.*

Definition 2 (Extent and support) *The extent of a pattern is the set of database objects it covers:*

$$ext(p, \mathcal{D}) = \{o \in \mathcal{D} \mid covers(p, o) \text{ is } true\}$$

The support of a pattern is the cardinality of its extent:

$$supp(p, \mathcal{D}) = |ext(p, \mathcal{D})|$$

The support is then the number of objects covered by p.

Often, the search space $\mathcal{S}$ is structured. The two following definitions are inspired by Belfodil in [14]. Interested reader will find there much more details about search spaces formalization, with an order theory point of view.

Definition 3 (Binary relation) *For each pattern mining task, there is a binary relation $\mathcal{R} \subseteq \mathcal{S}^2$ that is reflexive, transitive and anti-symmetric. We will denote it $p \ \mathcal{R} \ p'$ with $p, p' \in \mathcal{S}^2$, meaning p is in relation with p', in this order.*

Definition 4 (Pattern ordering and search space) *In this thesis, a search space of patterns $\mathcal{S}$, coupled to a binary relation $\mathcal{R}$, is called a partially ordered set, i.e., not every pair of $\mathcal{S}$ are comparable.*

This binary relation will naturally order the search space for each pattern mining task. By convention, we will represent the search space by putting the pattern $\top$ for which the relation $\top R\ p$ holds for any other pattern p, i.e., the most general pattern, on the top (level 0).

Each level $\mathcal{S}_t \subset \mathcal{S}$ is defined as follows:

$$\mathcal{S}_t = \{p_t \in \mathcal{S} \mid \exists\ p_{t-1} \in \mathcal{S}_{t-1}\ s.t.\ p_{t-1}\ \mathcal{R}\ p_t \wedge \nexists p' \in \mathcal{S}\ s.t.\ p_{t-1}\ \mathcal{R}\ p', p'\ \mathcal{R}\ p_t\}$$

2.1.2 `Apriori` and extracting association rules

Frequent itemset mining can then be described as a pattern mining task. Let $\mathcal{I}$ be *a set of items*. Each subset $X \subseteq \mathcal{I}$ is called an *itemset*. **A database object** o is an itemset. In this case, each **pattern** $p \in S$ is also an itemset. The **covering relation** is $covers(p, o) = p \subseteq o$. The predicate selection q is true i.f.f $supp(p, \mathcal{D}) \geq minSupp$, with $minSupp \in \mathbb{R}^+$, defined by the user. Note that the search space is composed of all possible subsets of $\mathcal{I}$, it is then of size $2^{|\mathcal{I}|}$. Its **binary relation** is the inclusion relation $\subseteq$. Note that in the case of itemsets the covering and binary relations are the same.

To tackle this problem, Agrawal et Al. proposed the `Apriori` algorithm in [1]. The search space can be represented as in Fig. 2.1. The more general pattern, i.e., the pattern covering all database objects, is represented at the top of the figure. Each transition between a level to its direct down neighbor is done by a direct pattern *refinement*. Informally, a refinement is a minimal operation making the pattern more specific, covering less instances. In the case of itemsets, it means creating a new pattern p' from p by adding an item $i \in \mathcal{I}\ s.t\ i \notin p$. We then have $p \subset p'$.

The key idea of this algorithm is to exploit a property that prunes search space to reduce its size: if an itemset p is frequent, all of its generalisation p', where $p' \subseteq p$, are frequent. Conversely, if an itemset p is infrequent, all of its specialisation p'', where $p \subseteq p''$, are infrequent. Going from the most general pattern $\top$, in a Breadth First Search way, i.e., level by level, using this property for pruning the search space, the `Apriori` explores the search space in a top-down way.

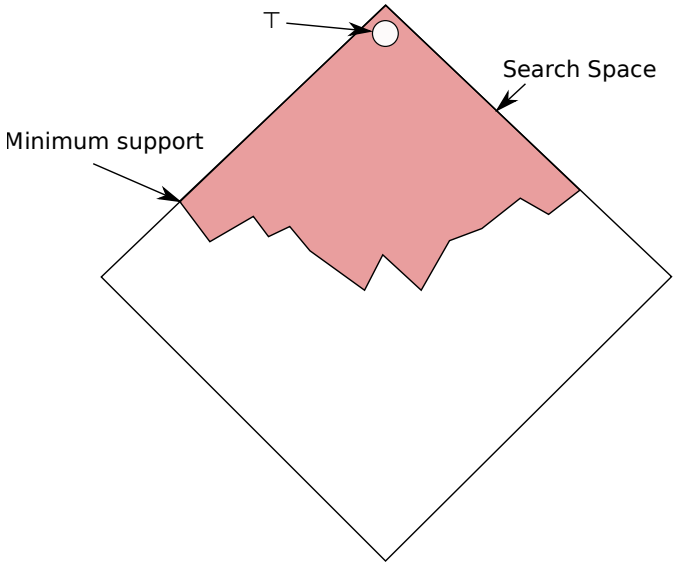


Figure 2.1: Illustration of `Apriori`.

VeilType	GillSpacing	Bruises	Poisonous
partial	crowded	no	no
partial	close	no	yes
partial	crowded	no	no
universal	close	no	yes
partial	close	yes	no
universal	distant	no	yes

Table 2.1: Mushrooms toy dataset

Once frequent itemsets have been extracted, for each pattern p, each possible rule of the form $body \rightarrow head$ is generated, where $body$ and $head$ are itemsets $body \subseteq p$ and $head \subseteq p \setminus body$. Each rule quality is assessed with its precision (also called confidence, or accuracy, or p($head|body$) [76]), defined by:

$$\mathcal{P}(body \rightarrow head) = \frac{supp(body, \mathcal{D}_{head})}{supp(body, \mathcal{D})}$$

with $\mathcal{D}_{head}$ being the subset of objects of $\mathcal{D}$ containing the itemset $head$,

Table. 2.1 is a toy dataset inspired by the mushroom dataset from the UCI repository [3]. It represents observations made by experts over different observed mushrooms.

Here each line corresponds to a database object. Such a dataset can easily be transformed to a boolean matrix, indicating for each database object what items appear in it, in order to be exploited by Apriori. Transformed dataset is given in Table. 2.2. Note that here the first row of the table corresponds in fact to $\mathcal{I}$.

Association rules mined with Apriori, with a minimum support threshold of 10% and a minimum confidence of 0.95 for rules, are:

$$GillSpacing : crowded \rightarrow VeilType : partial$$

[3] https://archive.ics.uci.edu/ml/datasets/Mushroom

VeilType: partial	VeilType: universal	GillSpacing: crowded	GillSpacing: close	GillSpacing: distant	Bruises	Poisonous
1	0	1	0	0	0	0
1	0	0	1	0	0	1
1	0	1	0	0	0	0
0	1	0	1	0	0	1
1	0	0	1	0	1	0
0	1	0	0	1	0	1

Table 2.2: Boolean Matrix transformation of 2.1

$$GillSpacing : distant \rightarrow VeilType : universal$$

$$GillSpacing : distant \rightarrow Poisonous$$

$$Bruises \rightarrow VeilType : Partial$$

$$VeilType : universal \rightarrow Poisonous$$

$$Bruises \rightarrow GillSpacing : close$$

$$Bruises, VeilType : partial \rightarrow GillSpacing : close$$

$$Bruises, GillSpacing : close \rightarrow VeilType : partial$$

$$GillSpacing : close, VeilType : universal \rightarrow Poisonous$$

$$Poisonous, VeilType : partial \rightarrow GillSpacing : close$$

$$GillSpacing : distant, Poisonous \rightarrow VeilType : universal$$

$$GillSpacing : distant, VeilType : universal \rightarrow Poisonous$$

Summarizing, the idea of `Apriori` is to extract interpretable rules in two steps. First, frequent itemsets are mined, exhibiting groups of dataset objects being frequent enough, i.e., there is a minimum support threshold that needs to be satisfied for extracted patterns. Secondly, association rules in the form $body \rightarrow head$ are extracted from those frequent itemsets: extracted rules have then more chances to cover a sufficient proportion of data.

Resulting rules can be of interest, but this method presents however several drawbacks. It is not anytime, i.e., the algorithm needs to finish to give results. Also, interesting rules can also exist below the user-specified $minSupp$. It can generate a large amount of itemsets, in which each association rule needs to be evaluated, which can be computationally expensive and give a too large set of rules. Here with such a small dataset with a very high confidence constraint we already have twelve rules. In other words, this approach consists in filtering the search space, then exploring this filtered set exhaustively to find interesting rules, using the precision measure. However such an approach does not scale with large $\mathcal{I}$, even more if applied to more complex pattern description languages having bigger search space. Finally, user may prefer to enumerate rules having a variable of interest in its right part: in that case we would be more interested in mushrooms characteristics that are discriminative of its poisonous property, in order to better identify them.

In order to reduce the size of the search space of frequent itemsets, it has been shown that exploring only a subset of patterns was sufficient: the closed patterns, introduced by Pasquier et Al. in [101].

Definition 5 (Closed pattern) *A pattern $p \in \mathcal{S}$ is closed iff $\nexists p'$ s.t. $p \mathcal{R} p'$, $ext(p) = ext(p')$.*

A closed pattern is then a most specific pattern possible for a given extent. This concept has been historically used in many other pattern description languages. It is said to compress the set of frequent patterns because it prunes patterns having the same extent as closed ones. Example of algorithms exploiting closed patterns are `CHARM` [136], `CBO` [74] or `Dryade` [124].

2.2 Supervised rule discovery: problem definition

The general problem we want to tackle in this thesis is to find a set of *interesting* rules. Several formalisms have been proposed to address this problem: subgroup discovery [130], emerging patterns [38] or contrast sets [13]. It has been shown that those approaches are different faces of the same problem in [97], under the term of *supervised descriptive rules discovery*. This is the terminology we are going to use in this thesis, with one simple variation: we consider the discovery of any rule, not only descriptive, as the predictive or descriptive value of a rule can be controlled by the chosen quality measure. Moreover, it has been shown in the very exhaustive and interesting work of Fürnkranz, Gamberger and Lavrac [49] that supervised descriptive rules discovery and standard rule learning are the two faces of the same coin.

To better formalize the problem, we need to introduce some definitions.

Definition 6 (Rule) *A rule is composed of a pattern $p \in \mathcal{S}$, with $\mathcal{S}$ being the search space of all possible patterns of the pattern description language, and a target class c. It is denoted $p \rightarrow c$.*

Definition 7 (Quality measure) *Let $\mathcal{S}$ be the set of all possible patterns in a dataset, and $\mathcal{C}$ the set of all possible target classes of a dataset $\mathcal{D}$. A quality measure*

$$\varphi : \mathcal{S} \rightarrow \mathbb{R}$$

maps every pattern $p \in \mathcal{S}$ to a real number to reflect its interestingness (inspired from [2]).

For instance, the quality measure *Precision* is a quality measure evaluating the proportion of positive elements covered by the rule, i.e., its *discriminating* power. Rougly speaking, a **discriminating rule**, and so, interesting, means that, in the dataset mined, if the pattern appears in an object, then there are chances that the class of the object is the one of the rule. As an example taken from [75], in Fig. 2.4, we have rules extracted from a dataset where objects represent medical conditions of people, and their class represent their diseases. A rule like $RespiratoryIllness = Yes \text{ and } Smoker = Yes \text{ and } Age \geq 50 \rightarrow LungCancer$ is easily interpretable: if the pattern appears, then the class of the object, i.e., the disease of the person in this case, is probably known.

The precision measure then fosters discriminating rules, without considering the pattern support, i.e., at which point it covers data. This can lead to the discovery of non-interesting rules, as a rule covering only one positive element would have a maximal precision, but would not be interesting for the end user, as it is too specific, i.e., true in a very specific setting. There is a trade-off to find between rule discriminating power, and generality. One of the most popular quality measure well balancing this trade-off is the Weighted Average Accuracy [76].

Definition 8 (Frequency) *The frequency of a pattern p is $freq(p, \mathcal{D}) = supp(p, \mathcal{D})/|\mathcal{D}|$.*

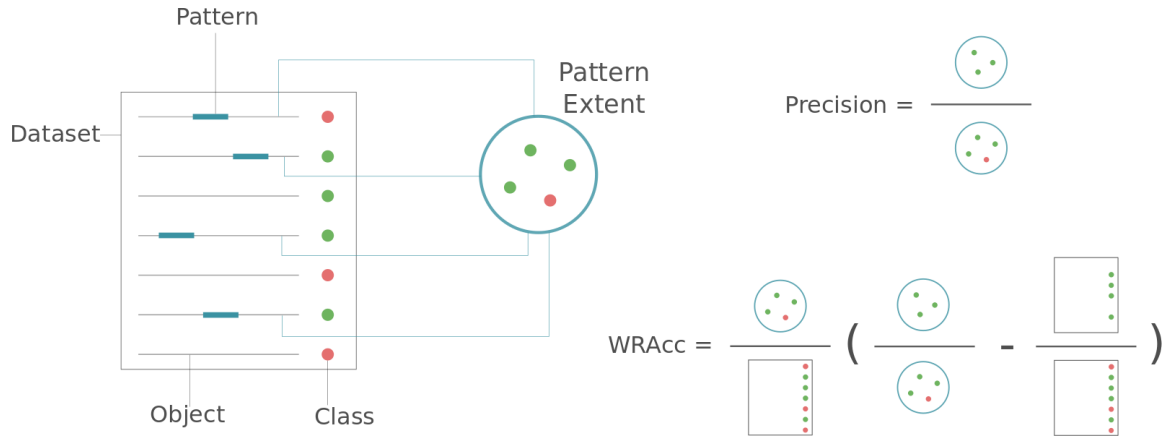


Figure 2.2: Illustration of supervised rule discovery

Definition 9 (Weighted Relative Accuracy) *The Weighted Relative Accuracy, or WRAcc, is a quality measure commonly used for supervised rules discovery. It compares the proportion of positive elements in the extent of the pattern to the proportion of positive elements in the whole database. Let $c \in C$ be a class value and p be a pattern,*

$$WRAcc(p \to c) = freq(p, \mathcal{D}) \times \left(\frac{supp(p, \mathcal{D}_c)}{supp(p, \mathcal{D})} - \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \right).$$

WRAcc is a weighted difference between the precisions $\mathcal{P}(p \to c)$ and $\mathcal{P}(\langle\rangle \to c)$. The weight is defined as $freq(p, \mathcal{D})$ to avoid the extraction of infrequent rules. $WRAcc$ value ranges in [-0.25, 0.25] in the case of a perfect balanced data set, i.e., containing 50% of positive elements. A visual explanation is given in Fig. 2.2.

In the case of labeled dataset, i.e., having a class for each object, closed on the positive patterns were introduced in order to offer a condensed representation of patterns covering positive objects [52, 16, 15]. In the same way as closed patterns pruned the search space for frequent patterns, closed on the positive prunes the search space for *discriminative* patterns, i.e., interesting patterns in the sense of characterising a target label.

Definition 10 (Positive object/element) *An object of the database labeled with the target class is called a positive element.*

Definition 11 (Closed on the positive pattern) *Let $ext^+(p)$ be the set of objects labeled with the class + for a pattern p. p is said to be a closed on the positive iff $\nexists p'$ s.t. $p \mathcal{R} p'$, $ext^+(p) = ext^+(p')$.*

Once rules have been extracted, it is interesting to give to the end user not only the best extracted rule, but the set of best found rules. Indeed, whether the goal is a descriptive or a predictive analysis, having only one rule would probably cover only a subset of the dataset. This would lead either to a partial understanding of the underlying domain where data are generated from in the case of a descriptive analysis, or an over simplified model, i.e., having a high bias, in the case of predictive analysis. In order to deal with this problem, we can return a set of top-k non θ-redundant rules. The notion of non-redundancy is used to give to the end user rules that

cover different subsets of objects, i.e., they have a different meaning. This is important to not decrease the trust of the user in the system, but also to keep the rule set interpretable. Note also that contrary to frequent itemsets mining, where the number of mined patterns can be gigantic, here we focus only on a subset of k rules, in order to not overwhelm the user.

Definition 12 (Non θ-redundant rules) *A set of rules $\mathcal{P} \subseteq \mathcal{S}$, is non θ-redundant if given $\theta \in [0;1]$ and $\forall p1, p2 \in \mathcal{S}$, where $p_1 \neq p_2$, we have: $sim(p_1, p_2) \leq \theta$, where sim is a similarity function. In the following we will use the Jaccard index as a similarity measure as in [79]:*

$$sim(p_1, p_2, \mathcal{D}) = \frac{|ext(p_1, \mathcal{D}) \cap ext(p_2, \mathcal{D})|}{|ext(p_1, \mathcal{D}) \cup ext(p_2, \mathcal{D})|}.$$

Now we can formalize the general problem we want to tackle in this thesis:

Problem 1 *For a database $\mathcal{D}$, an integer k, a real number θ, a similarity measure sim, a quality measure φ and a target class $c \in \mathcal{C}$,* ***the non redundant rule discovery task*** *consists in computing the set $\mathcal{S}_p$ of the best non θ-redundant rules of size $|\mathcal{S}_p| = k$, mined w.r.t the quality measure φ.*

This problem is a pattern mining task that can be formalized as proposed in [86]. The predicate q can be defined as:

$$q(p, \mathcal{D}) = \mathbb{1}(p \in \{ \underset{\mathcal{P} \subset \mathcal{S}, |\mathcal{P}|=k}{\operatorname{argmax}} \sum_{p \in \mathcal{P}} \varphi(p) \mid \forall p_1, p_2 \in \mathcal{P}^2, sim(p_1, p_2) \leq \theta \})$$

This problem is illustrated in in Fig. 2.2. Here only two classes are displayed. For the multi-class setting, we will use a one-versus-rest approach, i.e., we will consider object as positives if they have the target class, and negative if not.

Now for each instance of our general problem, we will need to redefine four elements:

- The dataset object definition
- The pattern definition
- The covering relation between a pattern and database objects
- The binary relation between patterns of the search space

Rules extraction attracted attention since decades, and is of particular interest nowadays. Indeed, with the apparition of deep learning methods [55], the need for interpretability has been exacerbated. Rules extraction algorithms gives interpretable rules. Note that, somehow counter intuitively, simple rules are not always really preferred by users, and on some domains, longer and more complex rules are more convincing, as showed recently by Fürnkranz et al. [50].

Note that the particular case of black box explanation models like Lime [110], Shap [85] or Anchors [111] are also based on rule discovery: to explain a prediction, the database object used for this prediction is perturbed with small variations, each one being given to the black box model. This results in a list of predictions, creating a new dataset that is mined to extract interpretable rules giving insights on why the model gives those predictions.

2.3 Search space exploration strategies

2.3.1 Enumeration-based methods

Many *enumeration techniques* enable to mine patterns from different pattern description language like, for example, boolean, numerical, sequential or graph data [54]. The main idea of such methods is to visit each candidate pattern only once while pruning large parts of the search space. Indeed, we know how to exploit formal properties (e.g., monotonicity) of many user-defined constraints. Their quality measure is thus computed either during or after the discovery step.

Adopting the general strategy of `Apriori`, the algorithm `Apriori-C` [68] and `Apriori-SD` [69] have been proposed. The main idea is to first launch `APriori`, and then to post process resulting rules with different strategies.

Years after the `Apriori` [1] algorithm has been presented, the `FP-Growth` algorithm has been proposed [58]. The idea is still to explore the search space in a top-down Breadth First Search way with smart pruning of the search space, based on the fact that the more we go down on the search space, the lesser the support. However the new idea here is to build a particular data structure during the search space exploration, called the FP-Tree. By reordering transactions according to frequencies of items, the FP-Tree can represent prefixes of the least frequent parts of the transaction (conditional pattern database). By recursively constructing new FP-Trees on conditional pattern databases, the algorithm makes a complete exploration of the search space, that can be limited with a minimal support constraint. The advantages over `Apriori` is that the database representation is smaller at each step, speeding up the process, and that it removes costly candidate generation from `Apriori`.

This method has been adapted to supervised rule discovery through the Subgroup Discovery framework with `SD-Map`[5] and `SD-Map*` [3]. They generalize the approach to numerical and categorical attributes, i.e., all database objects are vectors of elements that can take either a numerical or a categorical value, requiring a discretisation on the first one.

Zaki et al. [134] proposed a vertical representation of the database in order to increase the efficiency of the computation.

Cremilleux and Boulicaut proposed to use δ-strong rules to characterise classes [34]. Simply saying, a δ-strong rule is a rule which we tolerate to cover dataset objects labeled with an incorrect class δ times, i.e., we tolerate $body \rightarrow class$ to not hold for δ cases. They proposed to mine this subset, and showed that it reduced greatly the number of proposed rules.

Recently, the `OSMIND` algorithm has been proposed [90], adapted from [70] which were specific to the case of frequent patterns in numerical data. The strategy here consists in completely exploring the search space with in a depth-first-search (or Branch and Bound) way, with smart pruning strategies based on optimistic estimates and branch reordering. This method is also of interest because it does not relies on numeric discretisation, and explores a subset of the search space, i.e., reducing its size, containing however all interesting patterns, the closed patterns [51]. The approach proposed by Belfodil et al. named `FSSD` [15] also tackles the problem of rule set discovery, using a branch and bound algorithm, with the specificity of focusing on the quality measure of the rule set instead of each individual rules. The general strategy used by those methods are represented in Fig. 2.3.

The idea of using closed patterns to reduce the search space size has also attracted interest, for example with the algorithm `RMiner` [119]. Note that this algorithm propose to tackle the problem

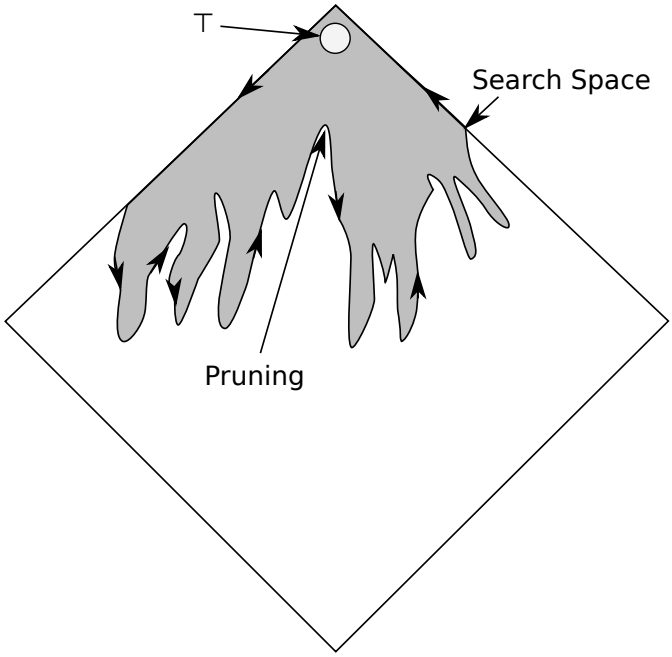


Figure 2.3: Illustration of Depth First Search-like algorithms.

of mining patterns in multi-relational data. It is based on the proposition of an exhaustive divide and conquer method proposed by Boley et al. [19].

2.3.2 Extracting rules from predictive global modeling

Several works have been done with the idea of optimizing the predictive power of a global model in first intention. The goal is to first predict globally, i.e., for each possible possible dataset object, then to extract interpretable rules.

One of the probably most popular approach using this idea is the decision tree [24]. One of the drawbacks of decision trees is that created rules have the constraint of not being overlapping, i.e., to be non-redundant at all ($\theta = 0$). This results in bigger rules set than necessary, being harder to interpret. This has been shown and called the replicated subtree problem by Pagalo and Haussler in [100]: identical subtrees can appear in the decision tree, due to this non-overlapping constraint, leading to difficulties of interpreting resulting rules.

Another well-known rule based models are decision lists, whose decision tree rules are a subset of [112], where if-else-then rules are extracted. One issue of this approach, as showed in [75], is that rules from decision lists can be difficult to interpret. Indeed, their if-else-then nature can create particularly complex rules when observing a small subset of data, because they are depending on previous extracted rules. This is illustrated in Fig. 2.4. Each rule of a decision list is depending on previous ones, making it more difficult to interpret.

In [64], authors proposed to fit a logistic regression in order to classify, with the particularity of selecting rules in their pattern description language (n-grams) as features. Then, once this logistic regression global model is trained, features having the biggest weights can be extracted to directly give interpretable rules, i.e., they predict the target class.

If Respiratory-Illness=Yes **and** Smoker=Yes **and** Age≥ 50 **then** Lung Cancer

If Risk-LungCancer=Yes **and** Blood-Pressure≥ 0.3 **then** Lung Cancer

If Risk-Depression=Yes **and** Past-Depression=Yes **then** Depression

If BMI≥ 0.3 **and** Insurance=None **and** Blood-Pressure≥ 0.2 **then** Depression

If Smoker=Yes **and** BMI≥ 0.2 **and** Age≥ 60 **then** Diabetes

If Risk-Diabetes=Yes **and** BMI≥ 0.4 **and** Prob-Infections≥ 0.2 **then** Diabetes

If Doctor-Visits ≥ 0.4 **and** Childhood-Obesity=Yes **then** Diabetes

If Respiratory-Illness=Yes **and** Smoker=Yes **and** Age≥ 50 **then** Lung Cancer

Else if Risk-Depression=Yes **then** Depression

Else if BMI ≥ 0.2 **and** Age≥ 60 **then** Diabetes

Else if Headaches=Yes **and** Dizziness=Yes, **then** Depression

Else if Doctor-Visits≥ 0.3 **then** Diabetes

Else if Disposition-Tiredness=Yes **then** Depression

Else Diabetes

Figure 2.4: Example of interpretable decision set (left) vs example decision list (right) (from [75])

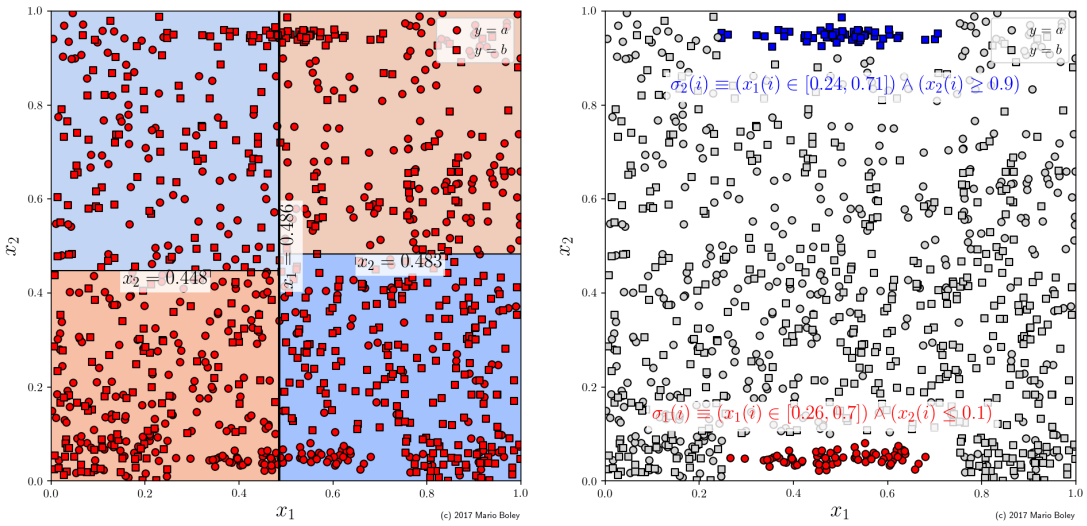


Figure 2.5: Decision Tree global decision (left) vs example of local rule (right) (from [18])

Note that in fact, rules extracted from those derived methods, i.e., the goal is first to fit a global model to predict, then to extract interpretable rules, constitutes a subset of all possible rules $\mathcal{S}$. This type of method can lead to several issues. Rules can, for example, be difficult to interpret, as they may trade precision for coverage. Another drawback of those approaches can be illustrated with decision (or regression) trees. Their predictions form rules by conjunctions of if-else statement taken by the path leading to a leaf. However, as showed in [18], this can prevent to find interesting rules that have high accuracy on a data subset, which can be of high value to better understand the underlying domain. For example, on Fig. 2.5, a decision tree of two level deep has been fitted to a dataset having two numerical attributes, and one categorical label to predict. As the decision tree needs to model globally, it can miss interesting local pattern that would be interesting, whether it is to better understand the underlying domain, or to better predict.

More recently, Interpretable Decision Sets have been presented [75]. The idea here is to find interpretable and accurate rule sets. They present the problem as an optimization one, with several target functions to optimize: model length, accuracy, rules overlapping, dataset covering. In fact, this function is a quality measure to optimize, that assesses the quality of obtained rules set, which are extracted thanks to a smart and theoritically guaranteed search space heuristic exploration. The problem addressed can then be seen as an instance of the problem 1 formulated in this thesis.

Being able to predict with high confidence on different subsets of data, and saying "I don't know" on others can be powerful [41]. This idea is also supported by Fürnkranz [48]: rule sets can be viewed as global models, but rules constituting it are locals. In particular, Knobbe et al. proposed the framework LeGo [72] to formalize a general and flexible method using a set of local rules as features to build predictive global models.

2.3.3 Heuristic Methods

An interesting trend to support supervised rules discovery is to avoid exhaustive search and to provide high quality patterns available anytime during the search, ideally with some guarantees on their quality. Examples of such guarantees are the distance to the best solution pattern [16] or the guarantee that the best solution can be found given a sufficient budget [22]. Let us discuss some of the heuristic approaches that have been proposed so far.

`Beam search` is a widely used heuristic algorithm. It traverses the search space level-wise from the most general to the most specific pattern and it restricts each level to a subset of non-redundant patterns of high quality [42]. The greedy and *exploitation-only* nature of `beam search` is its major drawback. For example, a pattern whose parents are bad could be ignored by this algorithm. However, it allows to quickly discover some interesting patterns. Removing redundancy of patterns at each step of `beam search` has shown interesting results with `DSSD` by Van Leeuwen et Al. [79]. A visual representation of beam-search is given in Fig. 2.7. Note that beam-search is implemented in free software such as CORTANA [4], VIKAMINE [4] or PYSUBGROUP [80], among other algorithms.

The `CN2-SD` algorithm [77] uses multiple `beam search` instances to select best rules w.r.t to an adapted Weighted Relative Accuracy measure [76] to build a classifier. The idea is to bias database objects with weights whose values depends if they have already been covered by other rules. Each new created rule corresponds to a new beam search launched. The `RIPPER` [33], `FOIL` [106] or `CPAR` [33] algorithms aim at creating classifiers using local rules. They used greedy heuristics to discover rules, selecting sequentially the best attributes w.r.t a quality measure to add to a rule body, with different stopping criterion.

Boley *et al.* proposed a two-step sampling approach giving the guarantee to sample patterns proportionally to different measures, namely: frequency, squared frequency, area, or discriminativity [20]. However this method only works on these measures. To consider another measure, Moens and Boley had to design a new method [92]. Dzyuba et al. [43] also proposed a sampling algorithm combined to an exhaustive one, agnostic of the quality measure to sample interesting pattern sets.

A visual representation of this kind of sampling methods is given in 2.6

Another set of techniques are those based on genetic algorithms [8], for example `SSDP+` [84], `MESDIF` [17], `NMEEF` [28] or `SDIGA` [35]. Genetic algorithms are meta-algorithms generaly used to solve optimization problems. To do so, they mimic the evolution process by creating individuals, i.e., rules in the case of rule discovery, whose quality is assessed thanks to a fitness function, i.e., a quality measure. Once the number of individuals is set, creating a population, the quality of each one is assessed. Best individuals are then selected, and combined together (crossover) to generate new individuals. To prevent the method from staying in a bad local optima, the notion of mutation is used, where some elements of individual are randomly changed, with a new

[4] http://datamining.liacs.nl/cortana.html

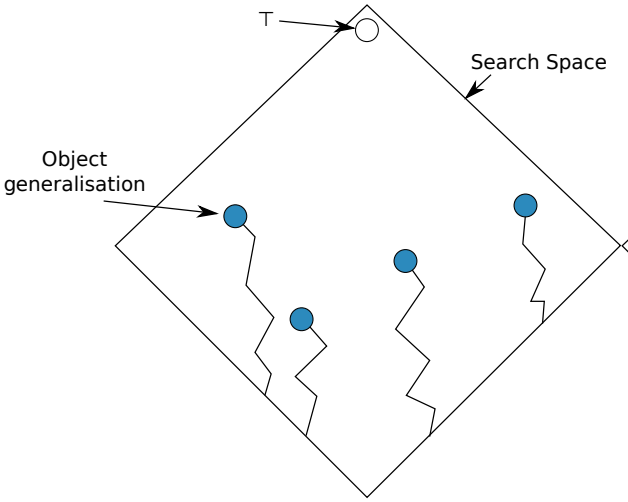


Figure 2.6: Illustration of sampling algorithms.

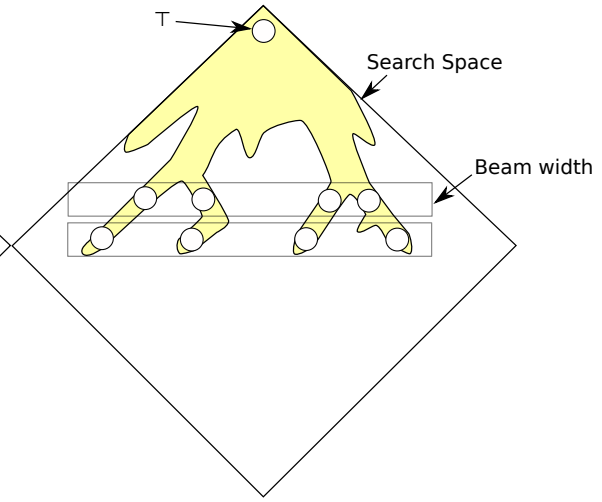


Figure 2.7: Illustration of beam-search algorithms.

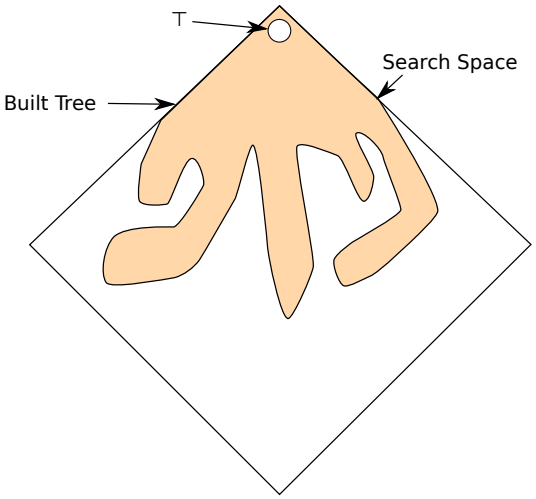


Figure 2.8: Illustration of MCTS4DM algorithm.

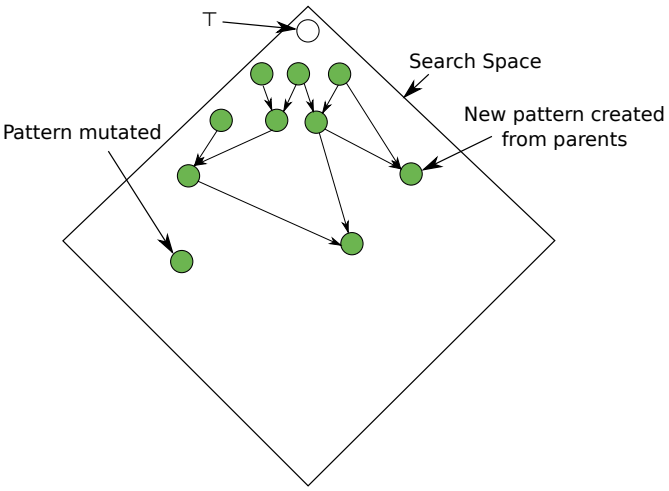


Figure 2.9: Illustration of genetic algorithms.

parameter controlling this probability. A visual representation of genetic algorithms is given in Fig. 2.9.

Genetic algorithms, however, presents several drawbacks. First they do not give any sort of guarantees on obtained patterns. Second the quality of resulting rules can vary a lot on two different algorithm launch, depending on the area exploited, which is partially random due to the mutation factor. Finally, the setting of parameters can be cumbersome: the user needs to set the number of generations, the population size and the mutation factor.

Belfodil et al. proposed a method called `RefineAndMine` [16], that extracts interesting patterns in numerical data, with the property of anytime, i.e., a results is available at any time, results improve time, and search converges to an exhaustive one if given enough time. Moreover, they provide interesting theoretical guarantees on mined patterns.

Bosc *et al.* used Monte Carlo Tree Search to support subgroup discovery, i.e., supervised rules discovery, from labeled categorical and numerical data [22]. They proposed an approach based on sampling, where each draw improves the knowledge about the search space. Such

a drawn object guides the search to achieve an exploitation/exploration trade-off. A visual representation of this algorithm (`MCTS4DM`) is given in Fig. 2.8.

2.4 Supervised Rule Discovery for sequences

Many works have been done on sequences, and are often influenced from methods presented in the previous section. One of the biggest difference of sequential pattern mining with itemset mining is that the search space can be order of magnitude bigger. Indeed, as shown in [104] and [107], the formula for counting the number of possible sequences is:

$$w_k = \sum_{i=0}^{k-1} w_i \binom{|\mathcal{I}|}{k-i}$$

$$SearchSpaceSize = \sum_{k=1}^{k_{max}} w_k$$

with k_{max} being the length maximal of a sequence, $w_0 = 1$ and $w_1 = n$ and $|\mathcal{I}|$ being the set of possible items. As an example, if we consider the well-known UCI dataset **promoters** [40], with $|\mathcal{I}| = 4$ and $k = 57$, the size of the search space is approximately 10^{41}.

In the same manner as `Apriori` for frequent itemsets, sequential pattern mining historically first dealt with the problem of finding frequent subsequences.

2.4.1 Sequential pattern mining

Sequences can be seen as generalisation of itemset: they are ordered list of itemsets. If itemsets of sequences all contain only one item, they are called "sequences of items".

The problem of sequential pattern mining can be explained in a visual way. A sequence of itemsets can be represented as showed in Fig. 2.10. Each vertical segment corresponds to an itemset, and each square of colour represents an item within this itemset. For example this sequence of itemset could be written: $\langle\{greySquare\}, \{greySquare, brownSquare\}, \{greySquare\}, \{greySquare, blueSquare\}...\rangle$. A dataset of sequences of itemset is then composed of different sequences like represented in Fig. 2.10 and Fig. 2.12. The goal of sequential pattern mining is then to extract patterns, i.e., subsequences, appearing in dataset, whose quality is assessed with quality measures, like frequency for frequent sequential pattern mining. A pattern p appearing on the first sequence is represented in Fig. 2.11 and for the second sequence in Fig. 2.13. Visually, we can see that a pattern, i.e., a subsequence, covers a sequence, i.e., an object, if its itemsets are included in itemsets of the sequence, in the same order and with a potential gap between them.

More formally, let $\mathcal{I}$ be *a set of items*. Each subset $X \subseteq \mathcal{I}$ is an *itemset*.

Definition 13 (Sequence of itemsets) *A sequence $o = \langle X_1...X_n\rangle$ is an ordered list of $n > 0$ itemsets. The size of a sequence o is denoted n, and $l = \sum_{i=1}^{n} |X_i|$ is its length. A database $\mathcal{D}$ is a set of $|\mathcal{D}|$ sequences (see Table 2.3 for an example).*

Definition 14 (Subsequence and covering function) *A sequence $p = \langle X_1...X_{n_p}\rangle$ is a subsequence of a sequence $o = \langle X'_1...X'_{n_o}\rangle$, denoted $p \sqsubseteq o$, i.e., o is covered by p, iff there exists*

Figure 2.10: Sequence of itemsets 1

Figure 2.11: A pattern p appearing in sequence 1

Figure 2.12: Sequence of itemsets 2

Figure 2.13: A pattern p appearing in sequence 2

$1 \leq j_1 < ... < j_{n_p} \leq n_o$ *such that* $X_1 \subseteq X'_{j_1} ... X_{n_p} \subseteq X'_{j_{n_p}}$. *In Table 2.3,* $\langle \{a\}\{b,c\} \rangle$ *is a subsequence of* o_1 *and* o_2. *In other words,* $ext(\langle \{a\}\{b,c\} \rangle) = \{o_1, o_2\}$. *In this pattern mining task, patterns are subsequences, and* $\mathcal{S}$ *is the search space of all possible subsequences.*

Note that $\sqsubseteq$ is also the binary relation for this pattern mining task, and is the same as the covering relation.

The first problem of sequential pattern mining has first been to find frequent subsequences, in a similar way as the pattern mining task for itemsets was first to find frequent ones. This resulted in methods exploring the search space in a similar way to frequent itemsets mining methods.

One of the first impactful work was the proposition of the `GSP` algorithm [120], which can be seen as an adaptation of the `Apriori` algorithm, i.e., a Breadth First Search strategy with pruning of the search space. With a Depth First Search strategy, the `Spade` algorithm [135] was proposed, and has been adapted for sequence classification based on frequent patterns [137].

Table 2.3: An example database $\mathcal{D}$.

id	$s \in \mathcal{D}$	c
o_1	$\langle \{a\}\{a,b,c\}\{a,c\}\{d\}\{c,f\} \rangle$	$+$
o_2	$\langle \{a,d\}\{c\}\{b,c\}\{a,e\} \rangle$	$+$
o_3	$\langle \{e,f\}\{a,b\}\{d,f\}\{c\}\{b\} \rangle$	$-$
o_4	$\langle \{e\}\{g\}\{a,b,f\}\{c\}\{c\} \rangle$	$-$

The idea of `Spade` was to represent the database in a vertical way (id-list), and to use the notion of equivalence classes to reduce the search space size, leading to better performances compared to `GSP`.

The `PrefixSpan` algorithm also uses a DFS approach, and is inspired by `FP-Growth` [58]. The idea is then to decompose sequences by prefixes, and iteratively and recursively finding frequent ones and creating their projected database, reducing the cost of support computing.

The algorithm `SPAM` [7] pushes further the idea of `Spade` with a bitset representation of database, to decrease support computing cost.

Extending `PrefixSpan`, algorithms `CloSpan` [132] and `BIDE` [128] have been proposed. They reduce the size of the search space exploring only closed patterns.

Main issues of sequential pattern mining using only frequency as a quality measure to optimize are the same as frequent itemsets mining: the number of candidate can be too large and there is redundancy between them. Note also that a problem with frequent sequential pattern used in the context of supervised sequential rules discovery is that a pattern which is frequent for a class does not mean it is discriminative of this class. For example, consider sequences of words, i.e., phrases, labeled by the book they are extracted from. The pattern "there is" can be frequent in a particular book, so it would be detected by algorithms of this section. However this pattern may not be interesting for the user, as it could also be frequent in other books: it is not *discriminative*. Extracted rules can then be of poor values, depending on the user desire.

2.4.2 Extracting interesting rules from sequential data

Several works have been done to tackle the problem of supervised rules discovery for sequences.

Morchen et al. [47] proposed an adaptation of `BIDE` to the problem of finding classification rules, i.e., subsequences, in order to build a classification system. However this method is quality-measure dependent.

He et al. in [60] and [59] used a top-down approach on sequences of items, using contrast sets [13] denomination. They propose a new way of removing redundant patterns, based on a property of the growth rate measure, which let them prune search space by removing subsequences from a pattern if this property is over a user-specified threshold. This method is then focused on sequences of items, is measure dependant, and requires several user parameters.

Ji et Al. [66] also proposed an interesting approach based on search space pruning for finding the best discriminative subsequences under a gap constraint. However, their work is also specific to only one quality measure. It requires to tune several parameters, including a minimum support, and it has been designed for processing sequences of items only, and not sequences of itemsets.

Gsponer *et al.* adapted the strategy of [64] on sequences of items, where a linear model is trained on a set of features extracted from sequences of the dataset [56]. Those features correspond to the search space of all possible subsequences. By minimizing a loss function, they can directly look at weights of their model to determine the most predictive subsequences. However, this approach is only applied on sequences of items, with numeric classes, and it can not choose a quality measure to optimize. For example, using this algorithm to find patterns corresponding to more general predictive rule covering many instances of the dataset with a lesser proportion of positive elements is not possible.

In a similar way as [20] for itemsets, Diop et al. proposed an approach which guarantees that

the probability of sampling a sequential pattern is proportional to its frequency [37]. However this method focuses on the frequency measure only.

Egho et al. have proposed the measure agnostic method `misère` [44]. Given a time budget, their idea is to generate random sequential patterns covering at least one object while keeping a pool of the best patterns obtained so far. It provides a result anytime, empirically improving over time, but there is no use of previous sampling: this is an *exploration-only* strategy. Notice however that it lacks a convergence guarantee. `Sqn2Vec` [95] proposed a new approach to represent sequences of items with embedding. An embedding represents symbols belonging to the corresponding sequence, and sequential patterns. It automatically finds the most discriminative ones thanks to a relatively simple neural network architecture, in order to then use this embedding, i.e., feature vector, to classify, or cluster sequences.

There are other problems, with other pattern mining language that propose interesting solutions. For example, Guyet et al. [57] proposed `NegSpan`, to discover negative sequential patterns, adding a specificity to the pattern description language for sequences of itemsets: an item can be expressly desired to not appear in sequences. The problem of finding maximal sequential pattern, i.e., pattern that are not sub-pattern of any other patterns in the dataset (then removing redundancy), has attract some interest, for example in [46].

To summarize, most of existing approaches present several drawbacks (or way of improvement) to tackle the problem of finding interesting rules in sequential data. The first issue is that some methods focus only on sequences of items. Second, methods searching for frequent rules among sequences having a class of interest can lead to non-discriminating (and non-interesting) patterns. The lack of theoretical guarantees can be problematic, as is the the number of parameters to set. Finally, many works are quality-measure dependent. As a lot of different ones exist, chosen depending on the type of rules the user may want, we want to propose methods agnostic of the quality measure.

2.5 Conclusion

The problematic of finding interesting rules in data, whether for descriptive or predictive analysis, is a subject of interest since decades. One of the main strengths of those approaches it that they give interpretable rules that can be used by domain experts: the goal is then not to replace them, but to boost their expertise with sophisticated tools. In particular, state of the art on supervised rule discovery for sequences has still open challenges: better dealing with the potentially huge search space, reducing the number of parameters to set or being able to be agnostic from the quality measure, for example.

Chapter 3

Bandit Models and Monte Carlo Tree Search

The multi-armed bandit model is well known in the Game Theory literature [26]. Several methods with guarantees have been proposed in order to solve problems that can be modelled with multi-arms bandit. Roughly speaking, they help choosing the best possibility in a sequential decision problem in order to maximize a reward function. This model is presented in Section. 3.1. As an evolution of those methods, the Monte Carlo Tree Search (MCTS) has been proposed. It consists in successive samplings of the search space, following an exploration-exploitation trade-off. The MCTS framework is presented in Section. 3.2.

3.1 Multi-armed Bandit Model

3.1.1 Problem settings

Bandit problems are sequential decision problems. At each iteration, one needs to select the best choice among k possible. This is modeled as a multi-armed bandit representing multiple slot machines in casinos, each one having its own reward distribution. The aim is to maximize the cumulative reward. The more an arm is played, the more information about its reward distribution we get. However, to what extent is it needed to exploit a promising arm (exploitation), instead of trying others that could be more interesting in the long term (exploration)? The formulation is then as follows: having a number N of plays, what is the best strategy, i.e., how to choose arms sequentially, to maximize the reward ? In fact, the problem is often presented as its dual: instead of maximizing the reward, one can minimize her regret, which is defined, after N plays as:

$$R_n = \mu^* n - \sum_{j=1}^{K} \mu_j \mathbb{E}[T_j(n)] \tag{3.1}$$

where μ^* is the best possible expected value of an arm among k possible, μ_j is the expected value of the j^{th} arm, and $\mathbb{E}[T_j(n)]$ is the expected number of play of arm j in the first n plays. Informally, this corresponds to the loss one can have if she does not play the best arm at each iteration.

3.1.2 Exploitation-exploration tradeoff

Auer *et al.* proposed a strategy called *UCB1* [6], for Upper Confidence Bound. It has the mathematical guarantee of having an expected logarithmic growth of regret uniformly over n. The idea is to give each arm a score, and to choose the one that maximizes it:

$$UCB1(j) = \bar{x_j} + \sqrt{\frac{2ln(n)}{n_j}}, \tag{3.2}$$

where $\bar{x_j}$ is the empirical mean of the j^{th} arm, n_j is the number of plays of the j^{th} arm and n is the total number of plays. The first term encourages the exploitation of arms with good reward, while the second encourages the exploration of less played arms by giving less credit to the ones that have been frequently played.

An example of the UCB1 strategy is given in Fig. 3.1. Each slot machine is represented with its (yet unknown) reward probability distribution on top. Yellow points represent a play of a slot machine, and the yellow vertical line represents the mean reward of the corresponding slot machine (or arm). The $n = 3$ on top left corner means that 3 plays have been performed, and $n_j = 1$ next to each slot denotes the number of time the arm has been played. For $N = 3$ we compute the UCB1 for each slot machine, and select the best one, which is the first here. We then play this arm, giving a new reward and updating its mean reward. This then decreases the value of the UCB1 of the machine 1 for two reasons: the mean reward decreased, and we played this machine more than the other. This results in selecting the third machine to play, giving a new reward, updating its mean reward and then its UCB1 value, etc.

The more we play arms, the more precise is the estimated value of the true expected value of each arm. Considering the second part of the UCB1 formula, we also foster on slot machine having less play, in case the randomness gave us a false mean estimation.

3.2 Monte Carlo Tree Search

3.2.1 Game Theory

Monte Carlo Tree Search has first been proposed as a strategy to explore search spaces, composed of different states, to choose the best action to perform for a decision problem. In game theory, this decision problem consists in finding the next action to perform, in order to win the game in the long run. A game can be described with a set of possible states, i.e., game configurations, a set of terminal configurations (where game finishes), a number of players, a set of actions that can be applied to states to make them progress, a state transition function, a reward function, assessing the quality of the configuration for a given player, and a function indicating the next player that needs to play for a given state.

MCTS can be seen as a bandit model approach, with memory. Roughly, the idea is to simulate, from an initial state, numerous final states, exploring the search space with an exploration exploitation trade-off. Those successive simulations give information about the search space, and they weight each created nodes with a value expressing to which point it leads to a "good" state, the notion of good depending on the problem and the reward function.

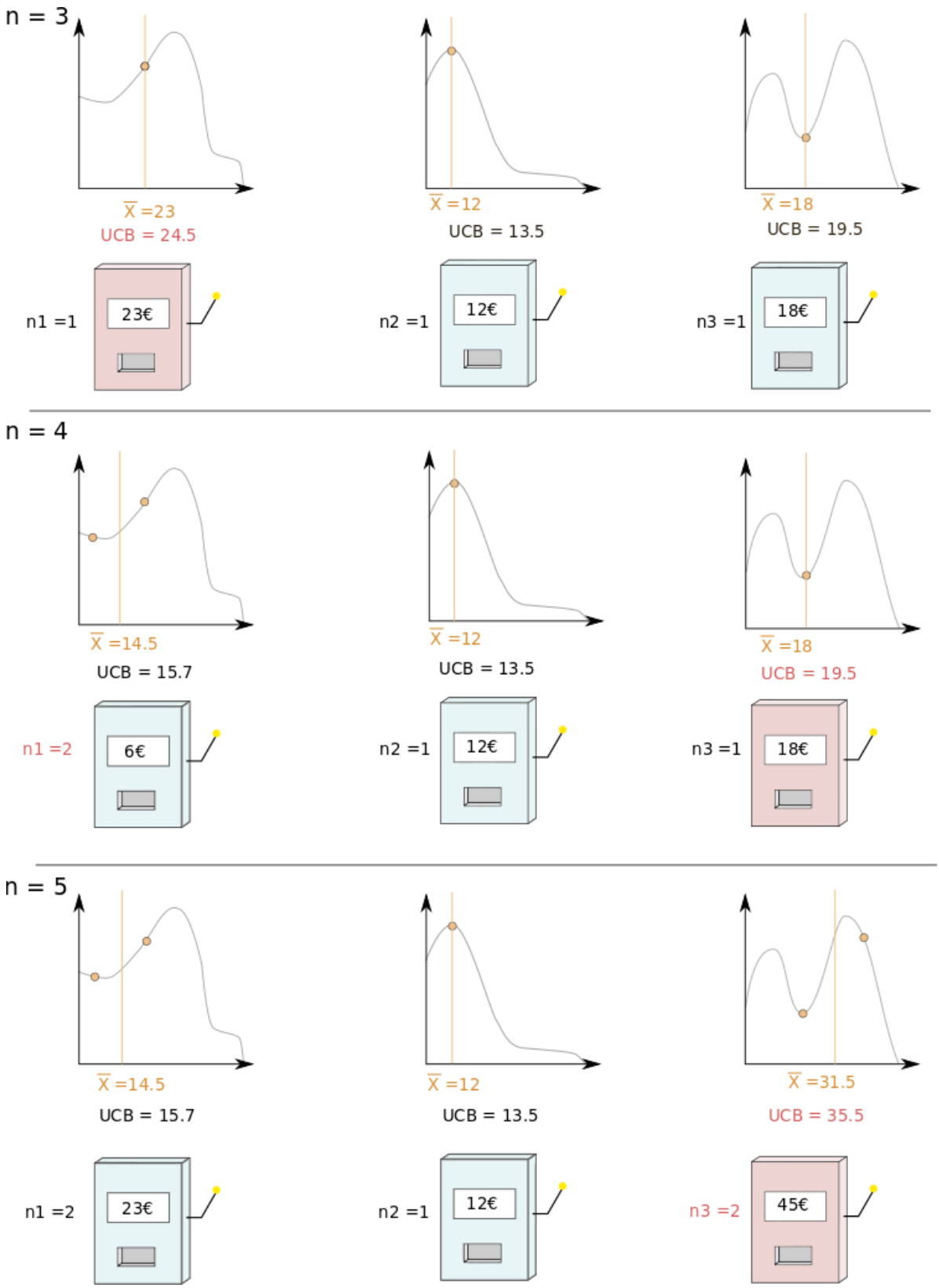


Figure 3.1: Illustration of the UCB strategy.

3.2.2 Method

The Monte Carlo Tree Search general approach is described in Fig . 3.2, and the pseudo code of one of its most popular instance, the Upper Confidence bound for Trees (`UCT`), is given in Algorithm 1. In Fig. 3.2, the top node corresponds to the initial state s_0, and each node corresponds to another state. Edges correspond to transitions between states. Each node, i.e., state, has a reward in $[0, 1]$, assessing its relevance for the given problem. A Monte Carlo Tree Search is composed of four steps being repeated iteratively.

The SELECT step First, the SELECT step (line 12-20 in Algorithm. 1) consists in exploring the already built tree and selecting the best node, i.e., state, that is not already fully expanded. A node is said to be fully expanded if all possible states accessible by performing one transition, i.e., its children node, have been expanded. The exploration exploitation dilemma is important here: do we try to exploit search space areas where we know there are interesting states, as we have sample them in previous ROLLOUT iterations, or do we try to explore the search space in area we do not have enough information yet ? Kocsis and Szepesvari first proposed in [73] to use the UCB1 from [6] to tackle this issue. The adapted formula of UCB1 for Monte Carlo Tree Search is:

$$UCT(s, s') = Q(s') + 2C_p\sqrt{\frac{2ln(N(s))}{N(s')}}, \tag{3.3}$$

where s' is a child of s, $N(s)$ being the number of time the node s has been selected, $Q(s')$ is the reward of the node s', and C_p is a strictly positive constant. The value of C_p can be tuned to foster more the exploration or the exploitation. Starting from the initial node, the UCT is computed for each child, choosing the one having the best, recursively, until a non-fully expanded node is reached.

The EXPAND step Once a node has been selected, it is expanded. This means that a random child of the selected node is created, i.e., a random possible transition from the state corresponding to selected node is chosen, and added to the built tree.

The ROLLOUT step The ROLLOUT step corresponds to a simulation that will give information to the following question: does the state of the expanded node leads to other interesting states ? This information is crucial to guide the search in next iterations of the MCTS. This step needs to be as computationally inexpensive as possible in order to maximise the number of MCTS iterations. Indeed, more iterations given to the tree means more information about the search space, so more probabilities of isolating its interesting areas. This is similar to the bandit problem, where the number of pulled arms needs to be order of magnitude greater than the number of arms to give a relevant mean estimation of underlying distributions. In order to perform a ROLLOUT, random transitions are successively performed from the expanded node, until reaching a terminal node. Note that the notion of terminal node depends on the underlying search space the MCTS is launched on. For example, on a search space of game configuration between two players, a terminal node corresponds to a configuration where one player wins. The reward of this terminal node is then computed.

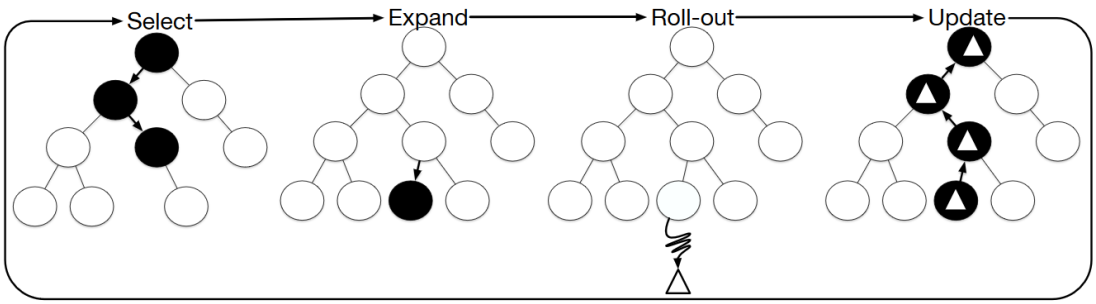


Figure 3.2: Steps of Monte Carlo Tree Search, inspired by [25]

The UPDATE step Once the reward has been computed, the information it gives is back-propagated through all nodes that have been visited during the current iteration, i.e., expanded node, selected node and all its chosen parents. Each of this node sees its quality $Q(s)$ updated with the new reward (its mean is re-computed). This way, if the reward is good, those nodes mean reward quality will increase leading to better chances of selecting them in next iterations.

This four steps are then performed successively, until no time budget is left or all search space has been explored. Then, the children having the best mean reward $Q(s)$ is selected: it is the one that tends to lead to best states for the initial problem.

Monte Carlo Tree search presents several characteristics making it interesting to use in our search space exploration setting. First, it is **aheuristic**, meaning that it can be applied directly on a search space that can be ordered as a tree to explore without requiring domain knowledge. However, domain-specific knowledge can be incorporated, for example as it has been done with the game of Go [39]. Second, it is **anytime**, meaning that the algorithm can be stopped at any moment, giving results, and that those results improve over time. Tree growing is also **asymmetric**, i.e., the tree tends to grow more to promising regions due to the UCT formula. This is a reason why the algorithm gives better results quicker than a Breadth First Search, for example: it can grow toward interesting area without having to enumerate all previous nodes. Finally, MCTS has an **exhaustive search guarantee** if given enough time. Contrary to purely random sampling strategies, MCTS will stop if all search space has been explored, before consuming all time budget. In this case, the algorithm becomes deterministic.

We present an illustration of some Monte Carlo Tree Search iterations on a case of a two-player game Tic-Tac-Toe on Fig. 3.3. It is the turn of the green player, and she faces the problem of choosing the next action to play. To do so, an MCTS is launched from the initial game configuration in (a). On (b), (c), (d) and (e), full cycles of MCTS are performed. As there are four possibilities of play from the initial state, there are four children nodes. In those four MCTS iterations, the selected node is directly the initial one, as it is not fully-expanded. Once the initial node is selected, is it expanded, creating a new configuration where green player places a circle on the board. In order to estimate the quality Q of this game configuration, the ROLLOUT step is launched: random successive plays are performed, until reaching a terminal state where one player has won. This terminal state gives information about whether or not the expanded node is an interesting configuration for the current player or not. The information is then back-propagated in the UPDATE step. For example, in (f), the third child of the initial node has been selected a second time ($N = 2$). The ROLLOUT led to a configuration where green player wins each time, the quality of the node is then $Q = 1$. Once the game search

Algorithm 1 UCT

```
1: function MCTS(budget)
2:     create root node s_0 for current state
3:     while computational budget do
4:         s_sel ← Select(s_0)
5:         s_exp ← Expand(s_sel)
6:         Δ ← Rollout(s_exp)
7:         Update(s_exp, Δ)
8:     end while
9:     return the child s of s_0 with the highest Q(S)
10: end function
11:
12: function SELECT(s)
13:     while s is non-terminal do
14:         if s is not fully-expanded then
15:             return s
16:         else
17:             s ← BestChild(s)
18:         end if
19:     end while
20: end function
21:
22: function EXPAND(s)
23:     randomly choose s_exp from non expanded children of s_sel
24:     add new child s_exp to s_sel
25:     return s_exp
26: end function
27:
28: function ROLLOUT(s)
29:     Δ ← 0
30:     while s is non-terminal do
31:         choose randomly a child s' of s
32:         s ← s'
33:     end while
34:     return the reward of the terminal state s.
35: end function
36:
37: function UPDATE(s, Δ)
38:     while s ≠ s_0 do
39:         Q(s) ← (N(s)*Q(s)+Δ) / (N(s)+1)
40:         N(s) ← N(s) + 1
41:         s ← parent of s
42:     end while
43: end function
44:
45: function BESTCHILD(s)
46:     return argmax_{s' in children of s} UCT(s, s')
47: end function
48:
```

space has been fully explored, or time budget limit is reached, the algorithms stops. Qualities of children of the initial node are then compared, and the action corresponding to the one having the biggest is selected: it is the configuration that statistically leads to better configurations for green player.

3.2.3 Applications

MCTS have been widely used in game theory [25], and particularly with the game of go [78]. One of the reason of the success of MCTS methods applied to the game of go is that it has a high branching factor, i.e., a big search space to explore, compared to chess where the use of the Minimax algorithm couple to good heuristics outperforms human expert decades ago [27]. More recently, MCTS methods received a lot of attention, in particular with the winning of AlphaGo against top world players [117].

Concerning data mining and machine learning approaches, MCTS have been applied to the feature selection problem with the algorithm FUSE [53]. The first application of MCTS to pattern mining and rules discovery has been proposed by Bosc et al. in [22]. They showed that this paradigm gave good results, particularly considering the diversity of obtained patterns.

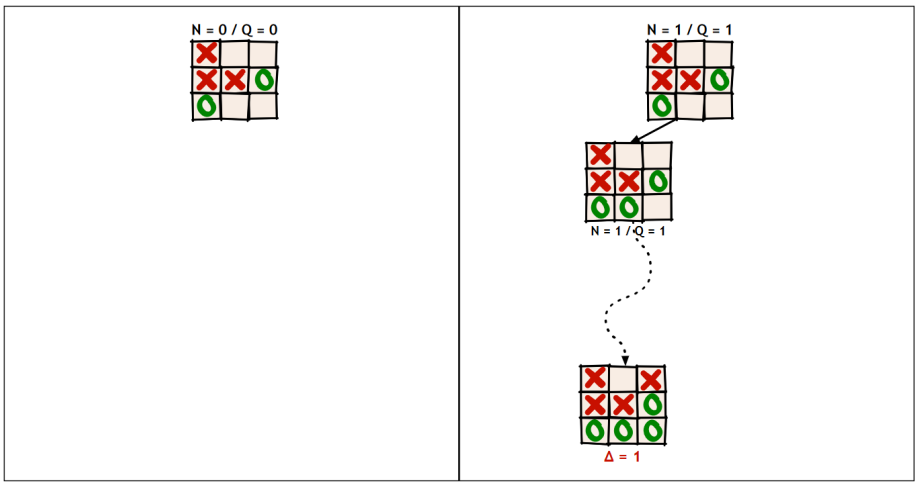
Nunes et al. proposed to adapt the framework of MCTS to Decision Trees Learning [99]. They enumerate the search space of possible Decision Tree, and launch an adapted UCT to this problem, showing that resulting Decision Trees have overall better predictive power.

Note that in those cases, MCTS have some particularities compared to MCTS used in game theory. First, every node of the search space has a quality that can be assessed, in contrast to the original definition of UCT, where non-terminal nodes take the mean quality of simulations run from them. Second, there is only one player, so the problem can be modeled as a single player game. Finally, the goal is not to weight children nodes of the initial state to select the best among them in a next action, but to explore the search space and to find the best possible patterns.

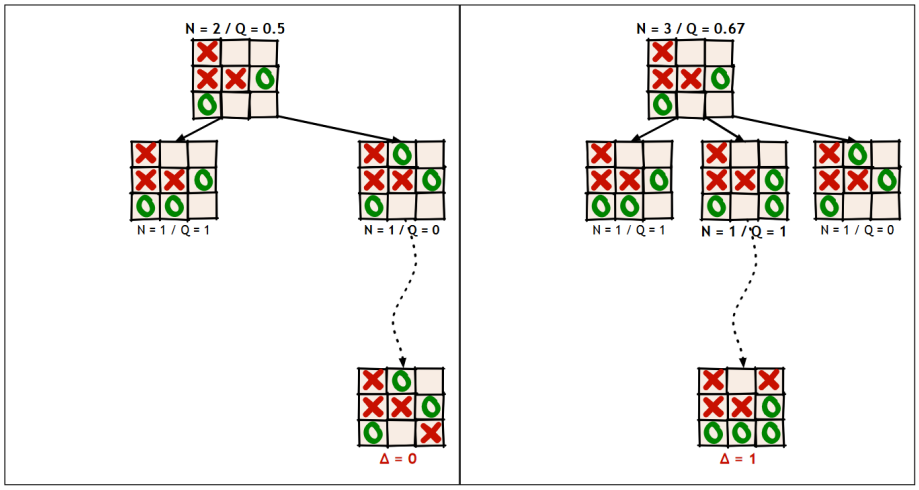
In this thesis we will use the MCTS framework to control the search for supervised rules discovery. As shown by Bosc et al. in [22], using MCTS to explore the search space efficiently requires tuning of its different steps. Moreover, they showed that this approach works greatly on classical transactional data, but when dealing with bigger search spaces, like sequences of itemsets, time series or high dimension numerical data, this method is not as efficient. To deal with this issue, we propose different algorithms using bandit models or MCTS approaches, but in contrast to `MCTS4DM`, we propose to explore the search space of *extents* in a *bottom-up way*, restricting the search space by exploring elements having a non-null support to reduce its size without pruning interesting patterns. We will also define new policies for EXPAND and ROLLOUT steps of MCTS to deal with data types we are interested in. In particular, we pay attention to have ROLLOUT steps as quick as possible, as this is the key of the performance of MCTS to perform well by taking advantage of the UCT formula.

3.3 Conclusion

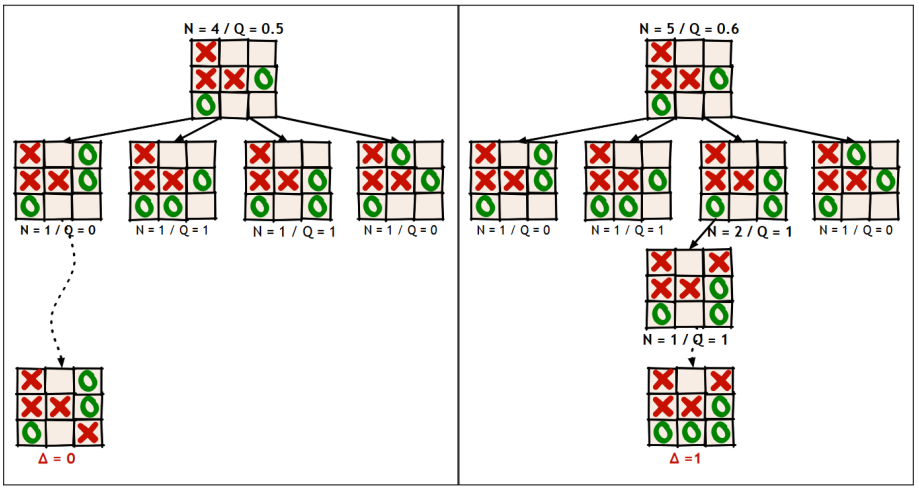
Methods to solve the multi-armed bandit problem like UCB1, and Monte Carlo Tree Search strategies like UCT come from decision and game theory, i.e., they were originally proposed to answer decision problems. However, they have some properties that make them interesting to



(a) Initialisation. (b) Iteration #1.



(c) Iteration #2. (d) Iteration #3.



(e) Iteration #4. (f) Iteration #5.

Figure 3.3: 5 iterations of MCTS to the Tic-Tac-Toe game (from [21])

adapt to other problems, as it has already been showed. In particular, using them to explore rules search spaces using an exploration-exploitation trade-off can be particularly relevant: rules qualities assessed by the chosen quality measure becomes the reward function, and the goal now it not to select the next best decision, but to extract interesting patterns from the search space. In the next chapter, we will take interest in adapting those methods to the problem of supervised rules discovery for sequences of itemsets.

Chapter 4

Mining interesting rules from sequences of itemsets

Considering an exploitation/exploration trade-off for pattern mining has already been proposed with itemsets and numerical by means of Monte Carlo Tree Search [22]. When dealing with a huge search space, using sampling guided by such a trade-off can give good results. However, contrary to [22], we consider here the search space of extents, and not the one of possible patterns, on sequence of itemsets. Exploring the search space of extents guarantees to find patterns with non null support while exploring the search space of possible patterns leads to the discovery of many having a null support. This is a crucial issue when dealing with sequences of itemsets. This chapter is organized as follows: we formally define the problem in Section 4.1. We then describe our solution algorithms `SeqScout` in Section 4.2 and `MCTSExtent` in Section 4.3. Section 4.4 presents an empirical study on several datasets to assess qualities of proposed methods.

4.1 Background

Let us now formalize our pattern mining task. It is an instance of Problem. 1, adapted to sequences of itemsets, which are **database objects**. **A pattern** is a subsequence, i.e., a sequence. **The covering relation** between patterns and database objects and the **binary relation** between patterns, ordering the search space, has been given in Definition. 14.

Other definitions need to be introduced here to understand proposed methods. We summarize the notations in Table 4.1.

Definition 15 (Set-extension) *A sequence o_b is a set-extension by $x \in \mathcal{I}$ of a sequence $o_a = \langle X_1 X_2 ... X_n \rangle$ if $\exists i, 1 \leq i \leq n+1$ such that $o_b = \langle X_1 ... \{x\}_i ... X_{n+1} \rangle$. In other words, we have inserted an itemset $X_i = \{x\}$ at the i^{th} position of o_a.*

Definition 16 (Item-extension) *A sequence o_b is an item-extension by $x \in \mathcal{I}$ of a sequence $o_a = \langle X_1 X_2 ... X_n \rangle$ if $\exists i, 1 \leq i \leq n$ such that $o_b = \langle X_1 ... X_i \cup \{x\}, ..., X_{n+1} \rangle$.*

For example, $\langle \{a\}\{c\}\{b\} \rangle$ is a set-extension of $\langle \{a\}\{b\} \rangle$ and $\langle \{a, b\}\{b\} \rangle$ is an item-extension of $\langle \{a\}\{b\} \rangle$.

Table 4.1: Notations

Notation	Description
$\mathcal{I}$	set of possible items
$m = \lvert\mathcal{I}\rvert$	number of possible items
$x \in \mathcal{I}$	item
$X \subseteq \mathcal{I}$	itemset
$\mathcal{D}$	database
C	set of classes
$\mathcal{S}$	set of all subsequences, i.e., search space
$o = \langle X_1 ... X_n \rangle$	sequence of itemsets
X_i^j	the i^{th} itemset in o_j
n	size of a sequence $o = \langle X_1 ... X_n \rangle$
$l = \sum_{i=1}^{n} \lvert X_i \rvert$	length of a sequence
$c \in C$	class
$p \sqsubseteq o$	p is a subsequence of o
$ext(p)$	extent of p
$supp(p)$	support of p
$freq(p)$	frequency of p
φ	quality measure
$Neighborhood(s)$	neighborhood of s

Definition 17 (Reduction) *A sequence o_b is a reduction of o_a if o_a is an set-extension or item-extension of o_b.*

Definition 18 (Local optimum) *Let $Neighborhood(p)$ be the neighborhood of p, i.e., the set of all item-extensions, set-extensions and reductions of p. $r^\star$ is a local optimum of $\mathcal{S}$ w.r.t. the quality measure φ iff $\forall r \in Neighborhood(r^\star)$, $\varphi(r^\star) \geq \varphi(r)$.*

In the context of sequential pattern mining, the search space is *a priori* infinite. However, we can define the border of the search space (the bottom border in Fig. 4.1) by excluding patterns having a null support. As the most specific patterns having a non-null support are database sequences, each element of this border is a sequence within the database. Therefore, the search space shape depends on the data.

4.2 SeqScout: SEQuential patterns Scouting

4.2.1 Adapting the multi armed bandit model to subsequence mining

The `SeqScout` algorithm is a sampling approach that exploits generalizations of database sequences, and searches for local optima w.r.t. the chosen quality measure. Fig. 4.1 provides an illustration of the method.

The main idea of the `SeqScout` approach is to consider each sequence of the labeled data as an arm of a multi-armed bandit when selecting the sequences for further generalization, using the Upper Confidence Bound (UCB) principle (see Algorithm 2). We recall that the idea of the

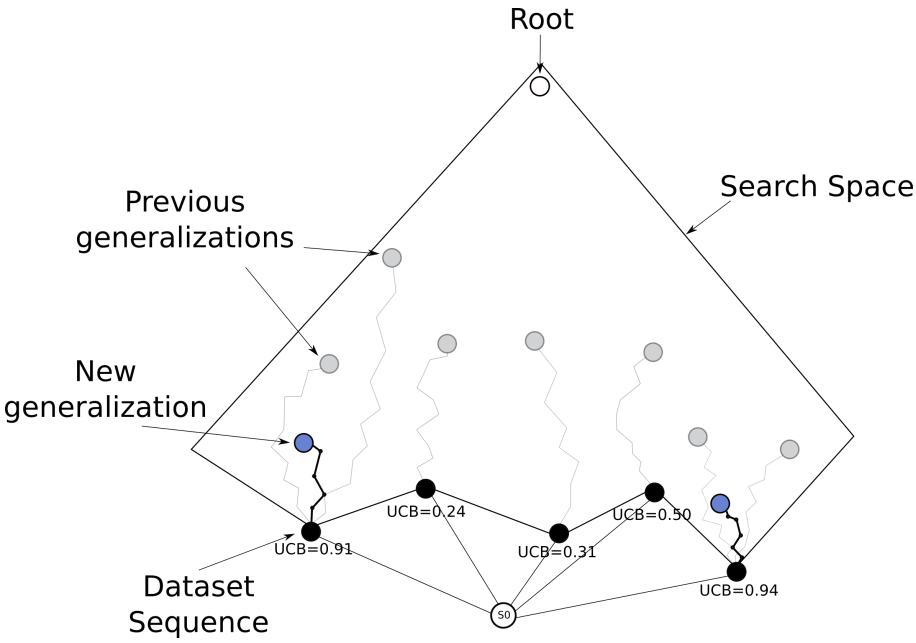


Figure 4.1: Illustration of `SeqScout`.

UCB is to give a score to each sequence, that quantifies an exploration-exploration trade-off, and to choose the sequence with the best one.

First (Lines 2-4), priority queues, π and *scores*, are created. π stores encountered patterns with their quality, and *scores* keeps in memory the list of UCB scores of each sequence of the dataset, computed by using equation 3.2 (see subsection 4.2.2). $data_+$ contains the list of all sequences of the dataset labeled with the target class. Indeed, taking sequences having the target class will lead to generalizations having at least one positive element. Then, the main procedure is launched as long as computational budget is available. The best sequence w.r.t. UCB is chosen (Line 9). This sequence is '*played*' (Line 10), meaning that it is generalized (see Section 4.2.3) and its quality is computed (see subsection 4.2.7). The created pattern is added to π (Line 11). Finally, the UCB score is updated (Line 12). As post processing steps, the top-k best non-redundant patterns are extracted from *scores* using the filtering step (see subsection 4.2.4). Finally, these patterns are processed thanks to a local optimization procedure (see subsection 4.2.5).

Moreover, `SeqScout` needs other modules that concern the selection of the quality measure (see Section 4.2.6) and the quality score computation (see Section 4.2.7).

4.2.2 SELECT Policy: Sequence Selection

We propose to model each sequence of the dataset as an arm of a *multi-armed bandit slot machine*. The action of playing an arm corresponds to *generalizing* this sequence to obtain a pattern, and the reward then corresponds to the *quality* of this pattern. Following an exploitation/exploration trade-off, sequences leading to bad quality patterns will be avoided, leading to the discovery of better pattern.

We then use the UCB1 (eq. 3.2) formula to score each sequence of the dataset, and we select

Algorithm 2 SeqScout

```
1: function SeqScout(budget)
2:     π ← PriorityQueue()
3:     scores ← PriorityQueue()
4:     data+ ← FilterData()
5:     for all sequence in data+ do
6:         scores_ucb.add(sequence, ∞)
7:     end for
8:     while budget do
9:         seq, qual, N_i ← scores.bestUCB()
10:        seq_p, qual_p ← PlayArm(seq)
11:        π.add(seq_p, qual_p)
12:        scores.update(seq, (N_i*qual+qual_p)/(N_i+1), N_i + 1)
13:    end while
14:    π.add(OPTIMIZE(π))
15:    return π.topKNonRedundant()
16: end function
17:
18: function OPTIMIZE(π)
19:    topK ← π.topKNonRedundant()
20:    for all pattern in topK do
21:        while pattern is not a local optima do
22:            pattern, qual ← BestNeighbor(pattern)
23:        end while
24:    end for
25:    return pattern, qual
26: end function
```

the one maximizing it.

4.2.3 ROLLOUT Policy: Subsequence Generalization

After the best sequence w.r.t. UCB1 is chosen, it is generalized, meaning that a new more general pattern is built. It enables to build a pattern with at least one positive element. Indeed, most of the patterns in the search space have a null support [107]. `SeqScout` generalizes a sequence s in the following way. It iterates through each item within each itemset $X_i \in s$, and it removes it randomly according to the following rule:

$$\begin{cases} remain, & \text{if } z < 0.5 \\ remove, & \text{if } z \geq 0.5 \end{cases}, \text{ where } z \sim \mathcal{U}(0,1).$$

The quality of the pattern is then computed, to update the UCB1 value of the sequence from which the pattern has been generated.

4.2.4 Filtering step

To limit the redundancy of found patterns, a filtering process is needed. We adopt a well-described set covering principle from the literature (see, e.g., [22, 79]) that can be summarized as follows. First, we take the best element, and then we remove those that are similar within our priority queue π. Then, we take the second best, and continue this procedure until the k best non-redundant elements are extracted.

4.2.5 Local optimum search

Finally, a local optimum search is launched w.r.t. Definition 18. Various strategies can be used. The first possible strategy is the Steepest Ascend Hill Climbing [113]. It computes the neighborhood of the generalized pattern, *i.e.*, all its item-extensions, set-extensions and reductions. Then, it selects the pattern among those of the neighborhood maximizing the quality measure. This is repeated until there is no more patterns in the neighborhood having a better quality measure. Another possible strategy is the Stochastic Hill Climbing [113]: a neighbor is selected at random if its difference with the current one is "large enough". Notice however that it introduces a new parameter. Depending on the dataset, the branching factor can be very important. Indeed, for m items and n itemsets in the sequence, there are $m(2n+1)$ patterns in its neighborhood (see Theorem 1). To tackle this issue, we use First-Choice Hill Climbing [113]. We compute the neighborhood until a better pattern is created, then we directly select it without enumerating all neighbors.

Theorem 1 *For a sequence s, let n be its size, l its length, and m the number of possible items, the number of neighbors of s, denoted $|Neighborhood(s)|$, is $m(2n+1)$.*

Proof 1 *The number of item-extensions is given by:*

$$|Iext| = \sum_{i=1}^{n} |\mathcal{I}| - |X_i| = nm - \sum_{i=1}^{n} |X_i| = nm - l.$$

We have now to sum the number of reductions, set-extensions and item-extensions:

$$|Neighborhood(s)| = l + m(n+1) + |Iext| = m(2n+1).$$

4.2.6 Quality Measure Selection

The choice of the quality measure φ is application dependent. Our approach can deal with any known measures that support class characterization, such as the $F1$ score, informedness or the Weighted Relative Accuracy [76].

We consider objective quality measures that are solely based on pattern support in databases (whole dataset, or restricted to a class). It enables a number of optimizations. Using random draws makes it particularly difficult as each draw is independent: we cannot benefit from same data structures as classical exhaustive pattern mining algorithms do (see, e.g., [7]).

4.2.7 Efficient Computation of Quality Scores

To improve the time efficiency of support computing, bitset representations have been proposed. For instance, SPAM uses a bitset representation of a pattern when computing an item- or set-extension at the end of a sequence [7]. In our case, we consider that an element can be inserted anywhere. Therefore, we use a bitset representation that is independent from the insertion position. Its main idea lies in keeping all bitset representations of encountered itemsets in a hash table (memoization), and then combining them to create the representation of the desired sequence.

The main idea of our strategy is given in Fig. 4.2. Assume we are looking for the bitset representation of $\langle\{ab\},\{c\}\rangle$. Let $\langle\{c\}\rangle$ be an already encountered pattern (i.e., its representation is known) while $\langle\{ab\}\rangle$ was not. This can not be handled by the SPAM technique as a new element has to be added *before* a known sequence. The algorithm will first try to find the bitset representation of $\langle\{ab\}\rangle$. As it does not exist yet, it will be generated and added to the memoization structure. Then, position options for the next itemset are computed (Line 2 in Fig. 4.2). The latter is then combined with a bitset representation of $\langle\{c\}\rangle$ using bitwise AND (Line 4). The support of the generated sequence can then be computed.

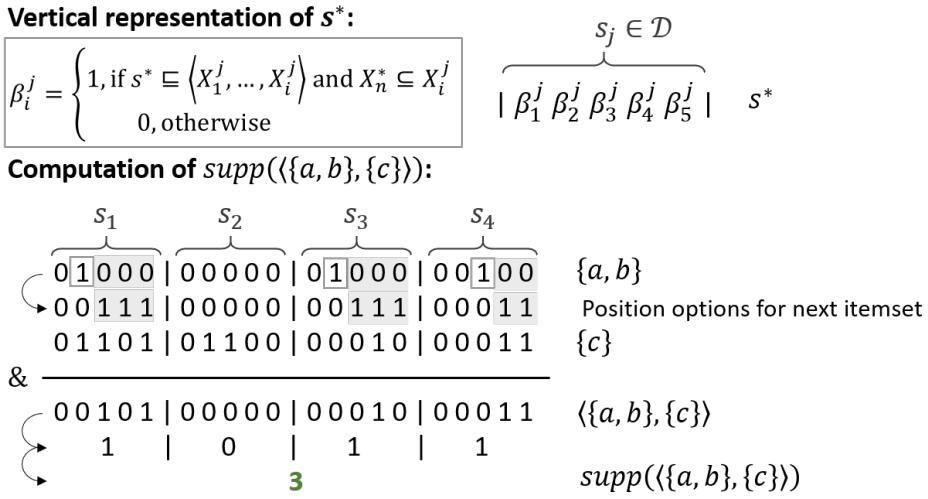


Figure 4.2: Bitset representation and support computing.

4.3 MCTSExtent

4.3.1 Applying MCTS in a bottom-up way

We now propose an extension of `SeqScout` that we call `MCTSExtent`. It is a logical evolution of `SeqScout` when looking towards a better trade-off between exploration and exploitation of a sampling strategy over the search space. It is based on the Monte Carlo Tree Search. Like for `SeqScout`, our idea here is to explore the search space in a bottom-up way, contrary to classical search space exploration in pattern discovery. Indeed, instead of beginning the exploration by selecting general patterns, we start by isolating very specific patterns covering only few objects having the target label, and we construct better ones by adding other interesting objects of the database. In other terms, we directly explore groups of objets, *i.e.*, extents having at least one positive element.

4.3.2 Algorithm Description

The pseudo-code of `MCTSExtent` is given in Algorithm 3. The main loop of the algorithm runs as long as a computational budget is available (Lines 4-11) making the algorithm anytime. Each node of the MCTS tree contains a list of positive instances, its extent, and a pattern covering them and only them.

The first step is to SELECT a node (Lines 15-23), *i.e.*, choosing the best node in the tree following the exploration-exploitation trade-off w.r.t UCB value. This step helps to guide the search towards promising areas of the search space, having good quality patterns, without ignoring that other non-explored parts can also be interesting. Thus, at each iteration, the algorithm checks if the current node is fully-expanded, according to Definition 19. If not, it is selected, and if yes, the SELECT procedure continues.

Definition 19 (Fully-expanded Node) *A fully-expanded node is a node which has already been expanded in all possible ways. Here it means there are no positive sequence to add to its extent to compute an unseen Longest Common Subsequence (LCS see Subsection 4.3.4).*

Then, a new node is created with the EXPAND step (Lines 25-29):

- a new positive different object is added to the ones that are in the selected node
- the Longest Common Subsequence (see Subsection 4.3.4) between this object and the pattern of the selected node is computed
- the extent of this LCS is then computed

We need to perform this last step because the LCS can cover more objects of the database than the union of the previous extent and the new positive object. It enables to get one of the most specialized pattern covering at least selected node objects and the new positive object, and its computational cost is negligible compared to support counting. Moreover, it creates a pattern having positive elements: it can lead to the creation of a good quality pattern if it covers less negative elements.

The next step is the ROLLOUT (Lines 31-38), where the node is generalized the same way as explained in Subsection 4.2.3. Finally, the score of the ROLLOUT is used to update the quality

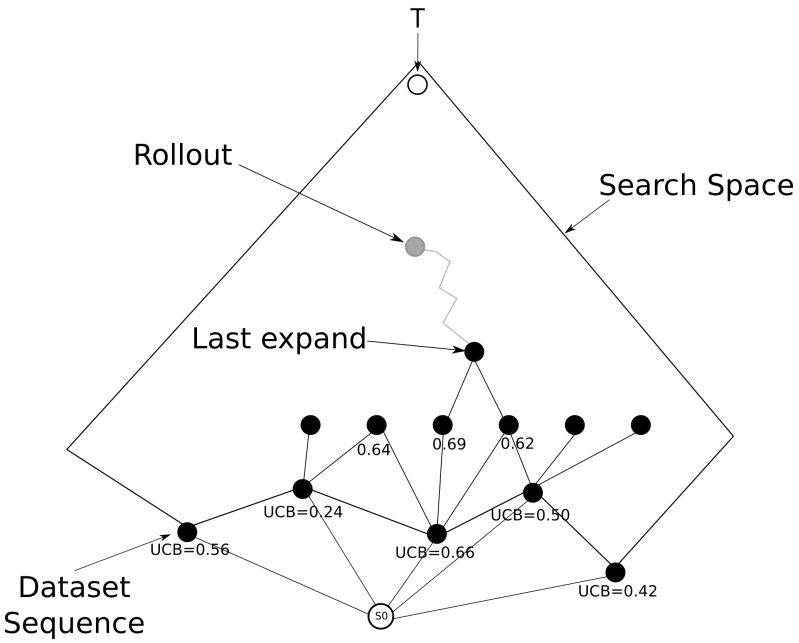


Figure 4.3: `MCTSExtent` Principle.

of the path followed to reach it: the expanded node, the selected node, and all its parents until the root. This step can be seen as a back-propagation of the result: the search space has been sampled, and we update the quality of nodes of the tree to indicate if this area is interesting or not for the next iterations.

Finally, once time budget is reached, the algorithm returns the top-k non-redundant elements, i.e., performing the same redundancy filtering step as `SeqScout`.

4.3.3 Example

An example of `MCTSExtent` steps is given in Figure 4.4. First, the SELECT function starts from the root node and it selects the best node to expand thanks to UCB. Node containing Object 3 is not fully expanded, so it is selected. Then the EXPAND function first takes a random positive element, which is 2 in this case, and adds it to the node. Then the LCS between the pattern of the selected node and Object 2 is computed: the extent of the node is now {1,2,3}. From this node, we can now make a ROLLOUT and the pattern is generalized. Finally, nodes from the path are updated with the reward of the ROLLOUT to provide feedback about the quality of this area of the search space.

4.3.4 Computing a Longest Common Subsequence

`MCTSExtent` needs the classical concept of Longest Common Subsequence (LCS). Hirschberg et al. have described a dynamic programming algorithm that solves this problem in polynomial time for sequences of items [63]. However, it does not work on sequences of itemsets. Vlachos et al. [127] introduced an algorithm for sequences of multidimensional real-values items, having parameters to enforce constraints on the difference between real values. This is a different

Algorithm 3 MCTSExtent

```
1: function MctsExtent(budget)
2:     π ← PriorityQueue()
3:     create s_0 empty root having all instances as children
4:     while computational budget do
5:         s_sel ← Select(s_0)
6:         s_exp, qual_exp ← Expand(s_sel)
7:         s_roll, Δ ← Rollout(s_exp)
8:         Update(s_exp, Δ)
9:         π.add(s_exp, qual_exp)
10:        π.add(s_roll, Δ)
11:    end while
12:    return π.topKNonRedundant()
13: end function
14:
15: function Select(s)
16:    while s is not root do
17:        if s is not fully-expanded then
18:            return s
19:        else
20:            s ← BestChild(s)
21:        end if
22:    end while
23: end function
24:
25: function Expand(s)
26:    s_+ ← randomly choose a positive instance
27:    s_exp ← ext(LCS(s, s_+))
28:    return s_exp, reward_{s_exp}
29: end function
30:
31: function Rollout(s)
32:    for item in each itemset in s do
33:        if random > 0.5 then
34:            s.remove(item)
35:        end if
36:    end for
37:    return s, reward_s
38: end function
39:
40: function Update(s, Δ)
41:    while s ≠ s_0 do
42:        Q(s) ← (N(s)*Q(s)+Δ) / (N(s)+1)
43:        N(s) ← N(s) + 1
44:    end while
45: end function
46:
47: function BestChild(s)
48:    return arg max_{s' in children of s} UCB(s, s')
49: end function
50:
```

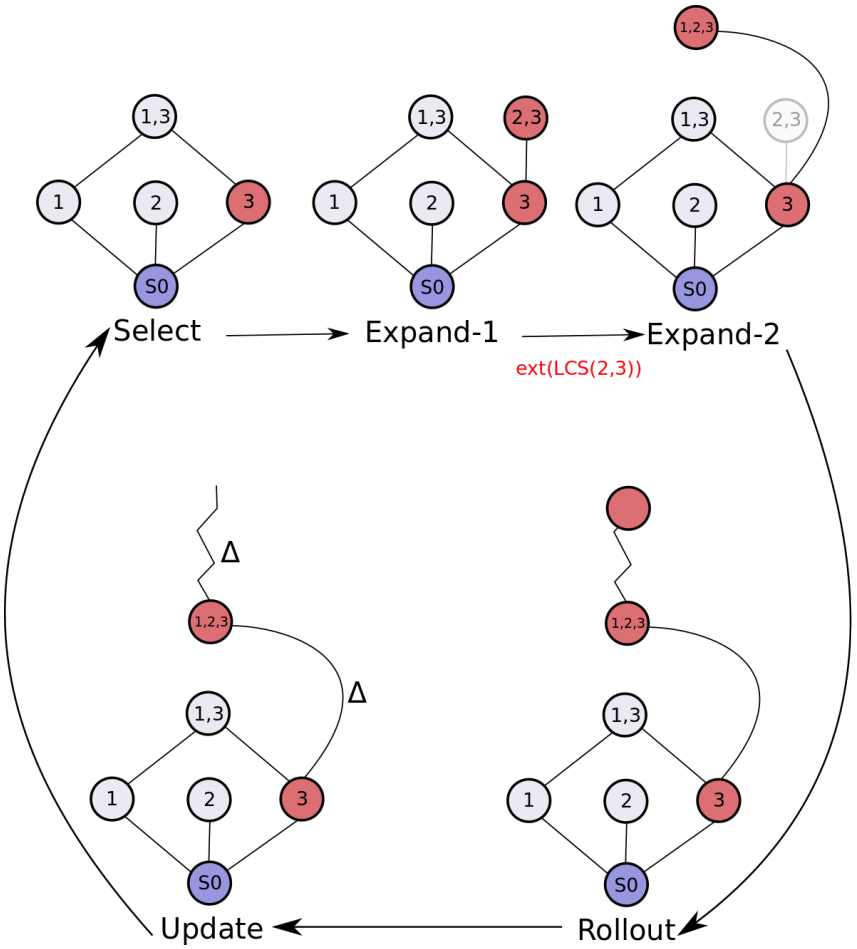


Figure 4.4: Illustration of `MCTSExtent`.

problem. Egho et al. have proposed an algorithm to compute the number of distinct common subsequences between two sequences of itemsets [45]. Such an algorithm does not generate the needed longest common subsequence.

Theorem 2 *Let two sequences of itemsets S^1 and S^2 of size n and m. We denote $S^1_{\leq i}$ the prefix of S_1, i.e., $S^1_{\leq i} = \langle X_1 ... X_i \rangle$. Let $LCS(S^1, S^2)$ be the set of the longest common subsequences of S_1 and S_2, or more formally:*

$$LCS(S^1, S^2) = \underset{s \in CS(S^1, S^2)}{\arg\max} \; length(s)$$

with length(s) the length of s, and $CS(S^1, S^2)$ the set of all common subsequences between S^1 and S^2.

We then have:

$$LCS(S^1_{\leq i}, S^2_{\leq j}) = \underset{s \in \chi}{\arg\max}\, length(s)$$

with

$$\chi = \bigcup \left\{ \begin{array}{c} LCS(S^1_{\leq i-1}, S^2_{\leq j-1}) \cup (X^1_i \cap X^2_j) \;\; (CaseA) \\ \\ LCS(S^1_{\leq i-1}, S^2_{\leq j}) \;\; (CaseB) \\ \\ LCS(S^1_{\leq i}, S^2_{\leq j-1}) \;\; (CaseC) \end{array} \right\}$$

Note that it generalizes the theorem of LCS for sequences of items from [63], where $X^1_i \cap X^2_j$ is an itemset of size 1. If last items are equals, $LCS(S^1_{\leq i-1}, S^2_{\leq j})$ and $LCS(S^1_{\leq i}, S^2_{\leq j-1})$ are less or equally long than $LCS(S^1_{\leq i-1}, S^2_{\leq j-1})) + 1$, so it is not necessary to look at it to compute the LCS.

To prove this theorem, we will need the following lemma.

Lemma 1 $\forall s_1, s_2 \in S^2, s_1 \sqsubseteq S^1, \; s_2 \sqsubseteq S^2$*:*

$$LCS(s_1, s_2) \in CS(S^1, S^2)$$

This comes from the fact that $LCS(s_1, s_2) \sqsubseteq s_1$ and $LCS(s_1, s_2) \sqsubseteq s_2$, so by transitivity, $LCS(s_1, s_2) \sqsubseteq S^1$ and $LCS(s_1, s_2) \sqsubseteq S^2$.

Proof 2 *Reductio ad absurdum: Let us assume we have an LCS which is not in a case of this theorem. We are not in Case B, a $LCS(S^1_{\leq i-1}, S^2_{\leq j})$. The LCS can then finish with an item of X^1_i. Let us assume this is the case for this demonstration. Symmetrically, it can finish with an item of X^2_j for Case C.*
The considered LCS then finishes with an itemset composed of elements from $X^1_i \cup X^2_j$ (it must be common, by definition). As it must be the longest, the last itemset of this LCS is $X = X^1_i \cap X^2_j$. We then have a LCS of the form $Y + X$, with $Y \notin LCS(S^1_{\leq i-1}, S^2_{\leq j-1})$, because we cannot be in the Case A. There is then only two possibilities for Y: whether Y is not common to S^1 and S^2, or Y is smaller than $LCS(S^1_{\leq i-1}, S^2_{\leq j-1})$. In both cases, it violates the definition of LCS. Thus, we showed by contradiction that:

$$LCS(S^1_{\leq i}, S^2_{\leq j}) \subseteq A \cup B \cup C. \tag{4.1}$$

Knowing that $S^1_{\leq i-1} \sqsubseteq S^1_{\leq i}$ *and given Lemma 1, we can derive that*

$$LCS(S^1_{\leq i-1}, S^2_{\leq j}) \in CS(S^1_{\leq i}, S^2_{\leq j}) \tag{4.2}$$

Symmetrically,

$$LCS(S^1_{\leq i}, S^2_{\leq j-1}) \in CS(S^1_{\leq i}, S^2_{\leq j}) \tag{4.3}$$

$$LCS(S^1_{\leq i-1}, S^2_{\leq j-1}) \in CS(S^1_{\leq i}, S^2_{\leq j}) \tag{4.4}$$

$$X^1_i \cap X^2_j \in CS(S^1_{\leq i}, S^2_{\leq j}) \tag{4.5}$$

We can deduce from (3), (4), (5) and (6) that:

$$A \cup B \cup C \subseteq CS(S^1_{\leq i}, S^2_{\leq j}) \tag{4.6}$$

From (2) and (7), we can conclude that the theorem is proven.

The pseudo-code of the dynamic programming procedure computing a LCS of two sequences of itemsets is presented in Algorithm 4. First the matrix C is filled with 0. Then, we use a bottom-up approach to fill the matrix with correct values, using the previous theorem. Note that a i, j cell contains the length of $LCS(S^1_{\leq i-1}, S^2_{\leq j-1})$. Once the computation of the length of the LCS is done, a backtracking procedure is launched, to construct the solution (Lines 1-21). We begin by looking at the "bottom-right" of the matrix (Line 32). We then check if there is an intersection between itemsets i and j (Line 5). If this is the case, we check if the sub-problem at rank $i-1$ or $j-1$ have the same LCS (those cases are here to check what path the LCS procedure took). Else, we add the intersection to the LCS, and we jump to the sub-problem of size $i-1$, $j-1$. If the intersection is null, we go to the sub-problem $i-1$ or $j-1$ having the maximum LCS (Lines 17-21). The procedure stops if we reach a sub-problem of 0 (Lines 2-3).

An example is given in Figure 4.5. The matrix is filled from the top-left cell to the bottom-right. At each step, following the theorem, we take the maximum value between the left cell, the upper cell, and the cell in the upper-left diagonal plus the length of the intersection of current itemsets.

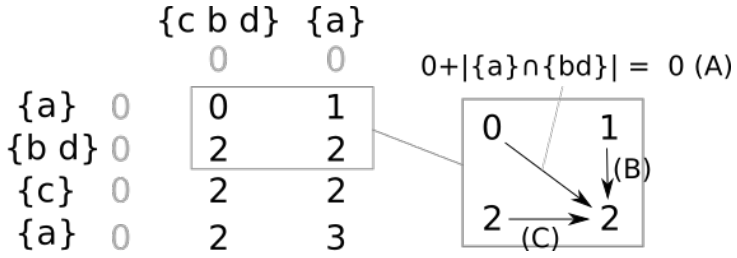


Figure 4.5: An example of the dynamic programming for LCS

Complexity Let l_1 and l_2 be the length of S^1 and S^2. Let X_{max} be the largest itemset present in the dataset. The computing of each cell of the matrix requires to look at 3 cells and to compute the intersection of two itemsets. This operation has a complexity of $O(3+|X_{max}|)$, so the time complexity of LCS is $O(l_1 * l_2 * |X_{max}|)$. The worst case of the backtracking procedure is $O(l_1 + l_2)$, which is negligible comparing to the complexity of LCS. The space complexity is $O(l_1 * l_2)$.

Algorithm 4 LCS

```
1: function BACKTRACK_LCS(C, S^1, S^2, i, j, lcs)
2:     if i = 0 or j = 0 then
3:         return
4:     end if
5:     inter ← S_i^1 ∩ S_j^2
6:     if inter ≠ ∅ then
7:         if C(i − 1, j) = C(i, j) then
8:             return backtrack_LCS(C, S^1, S^2, i − 1, j, lcs)
9:         end if
10:        if C(i, j − 1) = C(i, j) then
11:            return backtrack_LCS(C, S^1, S^2, i, j − 1, lcs)
12:        else
13:            lcs.insert(0, inter)
14:            return backtrack_LCS(C, S^1, S^2, i − 1, j − 1, lcs)
15:        end if
16:    else
17:        if C(i, j − 1) > C(i − 1, j) then
18:            return backtrack_LCS(C, S^1, S^2, i, j − 1, lcs)
19:        else
20:            return backtrack_LCS(C, S^1, S^2, i − 1, j, lcs)
21:        end if
22:    end if
23: end function
24:
25: function LCS(budget)
26:    Initialize C with dimensions size(S^1) ∗ size(S^2) filled with 0's
27:    for i=1 to size(S^1) + 1 do
28:        for j=1 to size(S^2) + 1 do
29:            inter ← S_i^1 ∩ S_j^2
30:            C(i, j) ← max(C(i − 1, j − 1) + length(inter), C(i − 1, j), C(i, j − 1))
31:        end for
32:    end for
33:    final_lcs ← list()
34:    backtrack_LCS(C, S^1, S^2, length(S^1), length(S^2), final_lcs)
35:    return final_lcs
36: end function
```

Table 4.2: Datasets

Dataset	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	l_{max}	Search Space Size
promoters [40]	106	4	57	$1.59 * 10^{41}$
context [40]	240	47	123	$5.25 * 10^{224}$
splice [40]	3,190	8	60	$3.29 * 10^{62}$
sc2 [23]	5,000	30	30	$6.48 * 10^{48}$
skating [93]	530	41	120	$1.16 * 10^{212}$
jmlr [121]	788	3,836	228	$1.84 * 10^{853}$
rl	148	16	157	$2.77 * 10^{212}$

4.4 Experiments

4.4.1 Datasets

We used several popular benchmark datasets to evaluate the behavior of our algorithms, namely **promoters** [40], **context** [40], **splice** [40] and **skating** [93].

We also apply our algorithms on the real life dataset **sc2** that has been used in [23]. It was extracted from Starcraft II games. Starcraft II is a Real Time Strategy (RTS) game that is well known within the AI community. Recently, it has attracted more attention after the publication of the Google DeepMind AlphaStar results, an AI defeating human players [126].

We also use **jmlr**, a dataset consisting of abstracts of articles published in the Journal of Machine Learning Research [121]. In this dataset, we consider sequences of words. As class labels, we used the occurrence of the word "svm" in a sequence, *i.e.*, we label by "+" sequences of words containing the word "svm", removing words after it, and "-" for others.

Finally, we used an **original** dataset **rl** for Rocket League[5] game analytic, composed of traces of player inputs.

Table 4.2 summarizes the statistics of used datasets.

4.4.2 Baselines

To the best of our knowledge, we are the first to address the problem of supervised discriminative rule discovery in sequences of itemsets, and therefore, there are not available algorithms that can be used directly as baselines for the evaluation. However, there are several algorithms for processing sequences of items. Therefore, to evaluate `SeqScout` and `MCTSExtent`, we have modified two algorithms, namely `misère` [44] and `Beam Search` [79], such that they process sequences of itemsets.

First, we implemented a simple extension of `misère` [44], the original version of which was handling sequences of events only but not sequences of itemsets. Second, we implemented `Beam Search` as a sequence-oriented version of a beam search algorithm. To deal with sequences of itemsets, we consider item-extensions and set-extensions at each given depth. Moreover, for the sake of non-redundancy in the returned patterns, we modify its best-first search nature so that the expanded nodes get diverse as defined in [79]. Moreover, to ensure fair comparisons, we removed the post-processing optimization of `SeqScout` that is studied more precisely in Section 4.4.12.

[5] `https://www.rocketleague.com/`

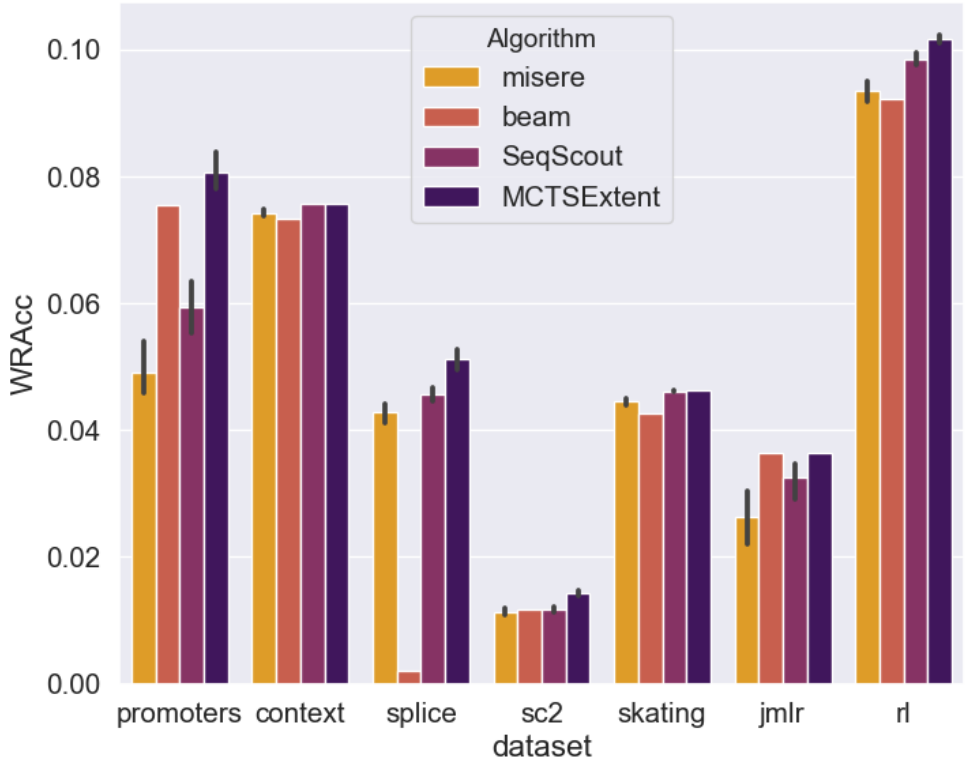


Figure 4.6: Mean $WRAcc$ of top-5 best patterns (10K iterations)

4.4.3 Settings

If not stated otherwise, we use the following settings. Each algorithm has been launched 5 times, and the reported results are averaged over these runs. For `Beam Search`, we empirically set the parameter $width = 50$. For all algorithms, we set $\theta = 0.5$, $time\_budget = \infty$, $iteration\_num = 10,000$, and $top\_k = 5$. Note that instead of giving a fixed time budget for running an algorithm on each dataset, we chose to limit the number of iterations $iteration\_num$, one iteration corresponding to a single computation of the quality measure. Indeed, this computation is the most time consuming one as such objective measures need to compute the extent w.r.t. the whole dataset. Therefore, using the same time budget on different datasets would not provide a fair comparison: having 50,000 iterations on a small dataset versus 50 on a larger one with the same time budget is not relevant.

4.4.4 Performance Evaluation using $WRAcc$

To assess the performance of the algorithms, let us first use the mean of the $WRAcc$ of the top-k non redundant patterns given by `misère`, `Beam Search`, `SeqScout` and `MCTSExtent`. Fig. 4.6 provides absolute results. `MCTSExtent` is clearly the best solution on each dataset. Interestingly, we can note that `Beam Search` is sometimes inefficient (see on **splice**).

We plotted relative improvements of algorithms (quality improvement ratio) in a one-vs-one

Table 4.3: Mean values of measures for top-5 patterns.

Dataset	Informedness				F1			
Algorithm	misere	BeamS.	SeqScout	MCTSExtent	misere	BeamS.	SeqScout	MCTSExtent
promoters	0.081	0.089	0.088	**0.144**	0.600	0.545	0.565	**0.636**
context	0.465	0.462	0.470	**0.472**	0.581	0.569	**0.586**	**0.586**
splice	0.373	0.041	0.392	**0.397**	0.428	0.086	**0.452**	0.451
sc2	0.013	0.006	0.011	**0.015**	0.531	0.533	0.541	**0.550**
skating	0.419	0.389	**0.423**	**0.423**	0.391	**0.402**	0.393	**0.402**
jmlr	0.439	**0.545**	0.449	**0.545**	0.330	0.402	0.337	**0.421**
rl	0.697	0.689	0.758	**0.779**	0.697	0.726	**0.739**	0.733

way on Fig. 4.7-4.11. We can clearly see that `MCTSExtent` and `SeqScout` perform particularly well on **splice** compared to `Beam Search`. Overall, `MCTSExtent` provides better results.

To achieve a comprehensive evaluation, we fixed a relatively small time budget of 60 seconds to compare the performances of algorithms, i.e., the limiting factor here is not the number of iterations but the given time budget. Results can be seen in Fig. 4.6. `MCTSExtent` generally outperforms other algorithms in terms of average $WRAcc$. We can also note that similarly to `misère`, `SeqScout` shows a significant decrease of performance on **jmlr**. Indeed, it seems that the strategy of taking a sequence and generalizing it is not efficient in a short time budget on this dataset. In contrast, `MCTSExtent` guides the search toward promising patterns more quickly, resulting in better performances.

4.4.5 Quality w.r.t. Number of Iterations

We show the result quality in terms of $WRAcc$ over the number of iterations. Fig. 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, depict the results for the top-5 non-redundant patterns on each dataset. Note that for the same data, the results may vary from run to run, due to the random component of `misère` and `SeqScout`. It explains some fluctuations of the quality. Nevertheless, for each *iteration_num* setting, `MCTSExtent` has shown better results.

4.4.6 Using other Quality Measures

To empirically illustrate the measure agnostic characteristic of `SeqScout` and `MCTSExtent`, we have used other quality measures, namely $F1$-score and $Informedness$. The results are shown in Table 4.3. Our algorithms generally give better results.

4.4.7 Performance Study under Varying θ

We also evaluate performances of the algorithms when varying the value of the similarity threshold θ. Fig. 4.19 shows the performance on the dataset **context** (results are similar on other datasets). We did not include the results for $\theta = 0$ because it would mean finding patterns with totally disjoint extents. It results in finding a number of patterns lesser than k for all algorithms, such that the mean would be misleading. We can see from the plot that relative performances of algorithms are approximately the same for all θ values.

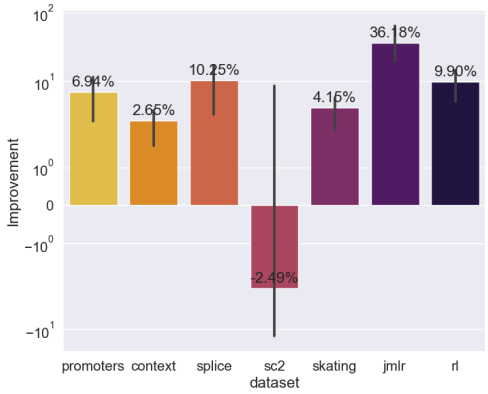


Figure 4.7: `SeqScout` vs. `misere`

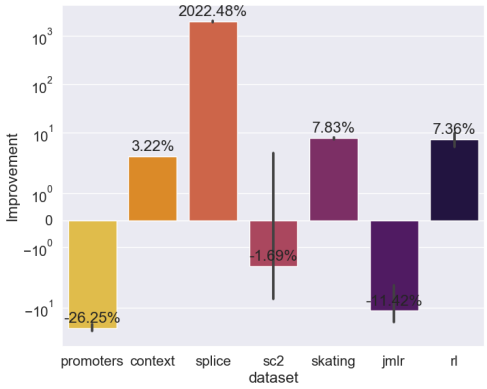


Figure 4.8: `SeqScout` vs. `Beam Search`

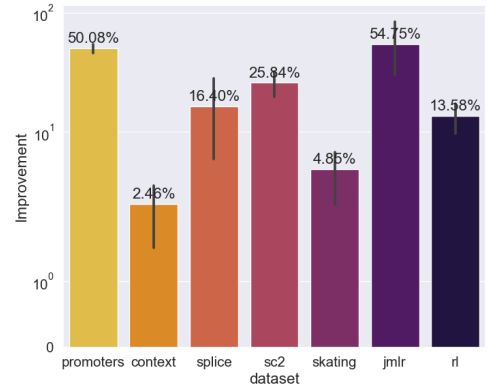


Figure 4.9: `MCTSExtent` vs. `misere`

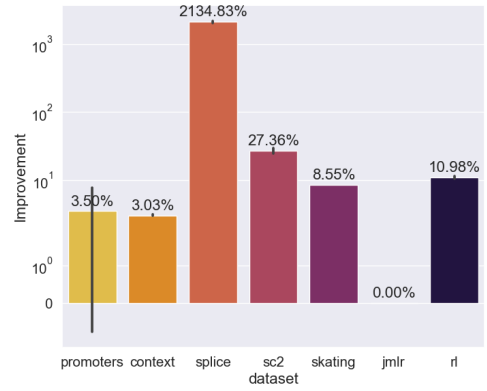


Figure 4.10: `MCTSExtent` vs. `Beam Search`

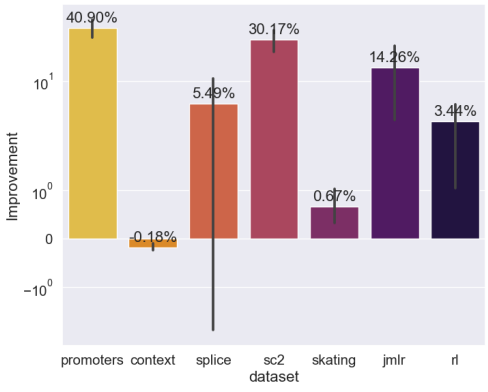


Figure 4.11: `MCTSExtent` vs. `SeqScout`

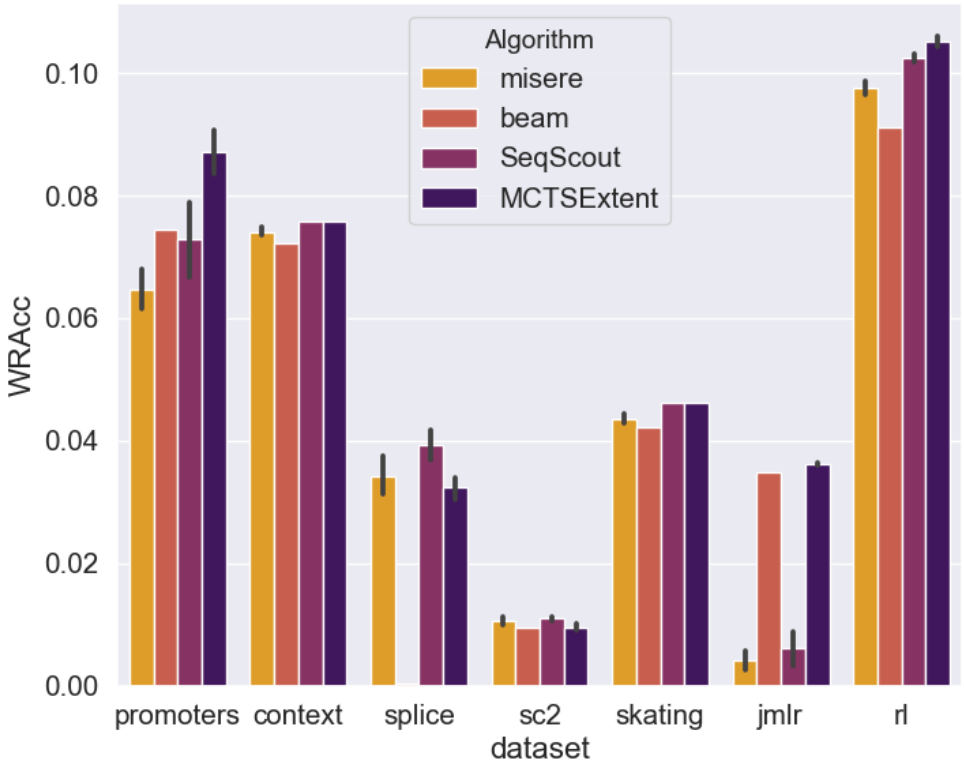


Figure 4.12: Mean $WRAcc$ of top-5 best patterns (time budget: 60s)

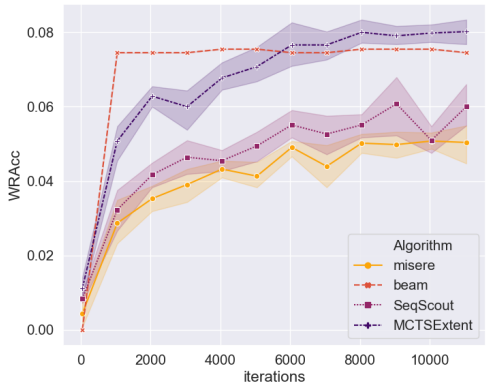


Figure 4.13: Average $WRAcc$ for top-5 patterns w.r.t. iterations (**promoters**)

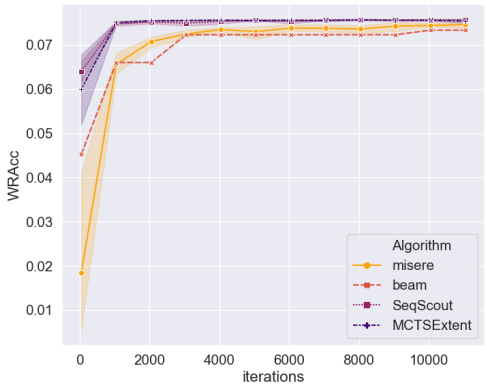


Figure 4.14: Average $WRAcc$ for top-5 patterns w.r.t. iterations (**context**)

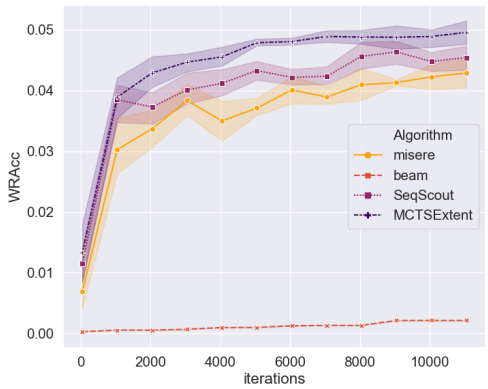


Figure 4.15: Average $WRAcc$ for top-5 patterns w.r.t. iterations (**splice**)

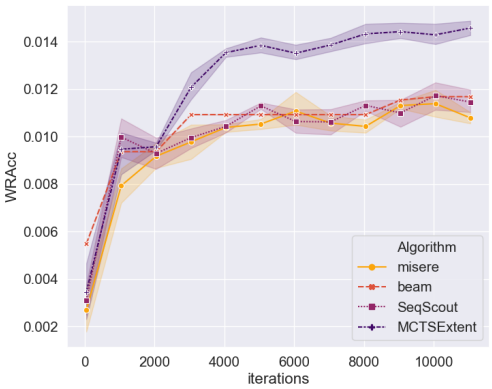


Figure 4.16: Average $WRAcc$ for top-5 patterns w.r.t. iterations (**sc2**)

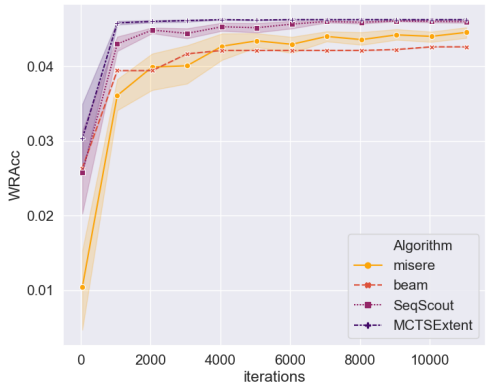


Figure 4.17: Average $WRAcc$ for top-5 patterns w.r.t. iterations (**skating**)

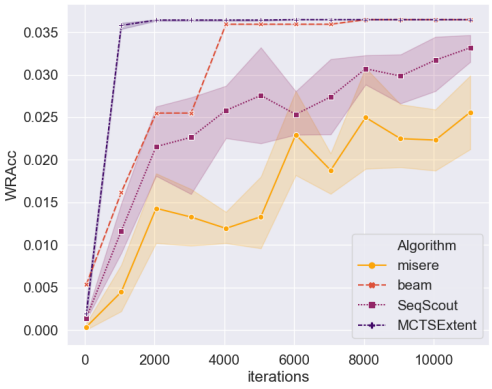


Figure 4.18: Average $WRAcc$ for top-5 patterns w.r.t. iterations (**jmlr**)

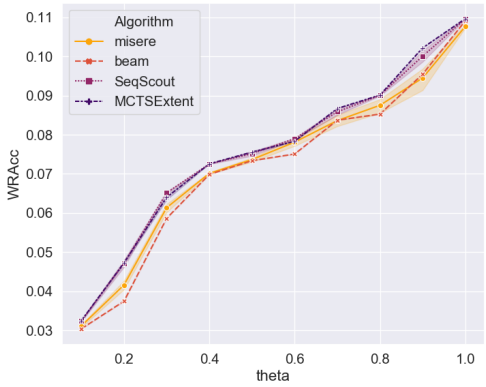


Figure 4.19: Mean $WRAcc$ of top-5 patterns vs. similarity threshold θ (**context**)

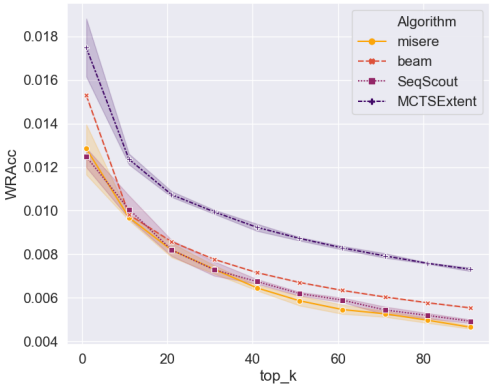


Figure 4.20: Mean $WRAcc$ of top-k patterns vs. k (**sc2**)

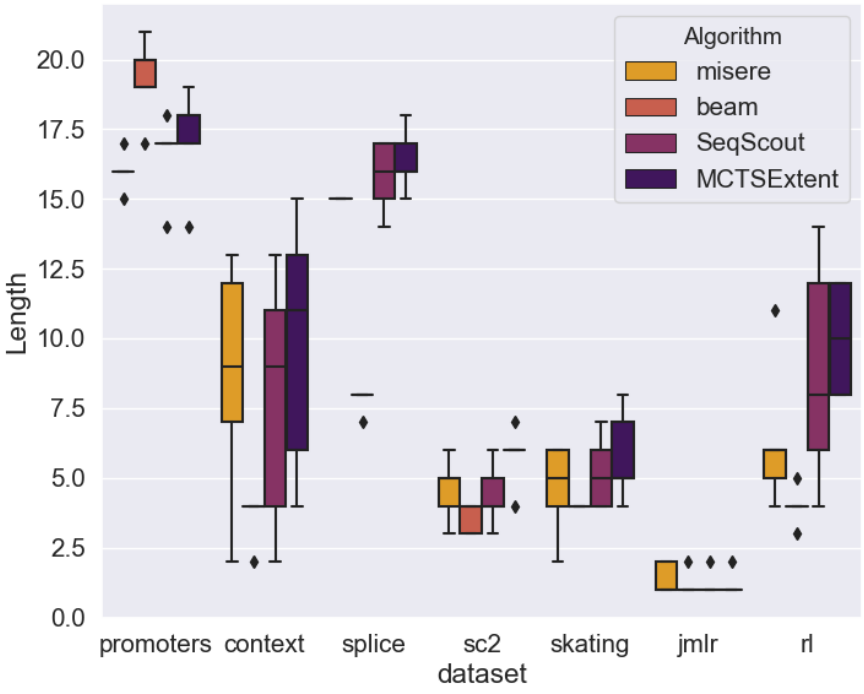


Figure 4.21: Length of top-5 best patterns - 10K iterations

Beam Search	Diversified	Non-Diversified
promoters	0.075	X
context	0.073	0.073
splice	0.002	0.002
sc2	0.116	0.119
skating	0.043	0.044
jmlr	0.036	0.036
rl	0.092	0.099

Table 4.4: *WRAcc* for top-5 patterns on diversified vs. non-diversified `Beam Search`

4.4.8 Performance Study under Varying top-k

We investigate the performance of the search for top-k patterns when changing the k parameter. Fig. 4.20 shows the results when considering the **sc2** dataset (the behaviour is similar on other datasets). `MCTSExtent` gives better results. Note that the mean *WRAcc* decreases for all algorithms, as increasing k leads to the selection of lower quality patterns.

4.4.9 Sequence Lengths

The pattern lengths on all datasets are reported in Fig. 4.21. Let us consider the **splice** dataset: `Beam Search` gives short patterns (max 8), which is significantly less than `MCTSExtent`, `SeqScout` and `misère`. This may explain why the `Beam Search` result quality is bad on this dataset (see Fig. 4.6). One hypothesis could be that it does not have enough iterations to go deep enough in the search space. Another hypothesis is that `Beam Search` cannot find good patterns having bad parents w.r.t. *WRAcc*: its good patterns are short ones. Another interesting observation is that on **jmlr**, the length of sequences is 1 or 2. This means that interesting patterns of this dataset are short ones, and are in fact itemsets: that is why `Beam Search` performs very well on it. In fact, using a subgroup discovery algorithm exploiting itemset descriptions should give similar results.

4.4.10 Non Diversified `Beam Search`

Leeuwen et Al. [79] proposed to filter redundant patterns during the `beam-search` procedure. We wanted to check if this strategy remains efficient in the case of sequences of itemsets. In fact, the main issue with the classical `Beam Search` approach is that when filtering is done in post-processing, we can get less than k patterns, depending on the configuration. In fact, a classical `Beam Search` is not relevant to solve our problem. A comparison of the two `Beam Search` strategies is given in Table 4.4. We added a "X" when the number of found patterns is lower than k.

4.4.11 Bitset vs. Integer Set Representation

We investigate the usefulness of bitset representation by comparing it against an integer set representation where each item is represented by an integer. Therefore, we have compared the number of iterations `SeqScout` made for a fixed time budget on each dataset. We set $time\_budget = 10$ seconds. The results are summarized in Table 4.5. We can see that the

Table 4.5: Number of iterations in Bitset vs. Integer set representation

Dataset	Integer Set	Bitset	Variation(%)
promoters	7,185	8,858	24
context	4,651	2,667	-47
splice	289	254	-12
sc2	943	605	-36
skating	4,001	1,283	-68
jmlr	704	31	-95
rl	8839	7799	-12

bitset representation tends to give a performance gain for datasets with smaller search space size upper bound, but leads to a decrease of performances for the majority of the datasets. For example, **context**, **skating** and **jmlr** have sequences of large lengths leading to large bitset representations. Those bitsets are then split into different parts to be processed by the CPU. If the number of splits is too large, the bitset representation becomes inefficient, and using a classical integer set representation is a better option.

4.4.12 Local Optima Search

The local optima search uses more iterations. This over cost depends on the dataset. In Fig. 4.26, we plot the ratio of the additional iterations necessary for local optima search w.r.t. the number of iterations given in the main search (also referred to as the cost). The more iterations we have, the more negligible the ratio is. However, note that we did not plot the additional cost of **jmlr**. Indeed, in the particular case of text data, the number of possible items is large, leading to a very long local optima search (110,000 iterations for 5 patterns in our experiments). Consequently, we note that the local optima search may not be the relevant choice with this kind of dataset.

We also added it as a post-processing step to each of our algorithm to compare general quality increase depicted in Fig. 4.22-4.25. As we can see, the more initial iterations are given, the better the mean quality is, and the lesser the local optima search is interesting. This emphasizes on the fact that `MCTSExtent` performs a good exploration of the search space. Note that on some dataset, like **promoters**, this local search generally leads to an important quality increase.

4.5 Conclusion

We have discussed the problem of finding dicriminative patterns in labeled sequences of itemsets. We have presented two algorithms `SeqScout` and `MCTSExtent` to discover relevant rules for this type of data. Though we are not aware of available algorithms to solve the same problem, we have implemented the adaptations of two other algorithms, namely `misère` and `Beam Search`, such that they could be applied for the case of sequences of itemsets. This has been useful to support our empirical evaluation.

Furthermore, to implement `MCTSExtent`, we have also introduced a new algorithm to compute a Longest Common Subsequence between two sequences. Our experiments have shown that `MCTSExtent` outperforms all other algorithms, without the need for additional parameter tuning as needed in the case of `Beam Search`.

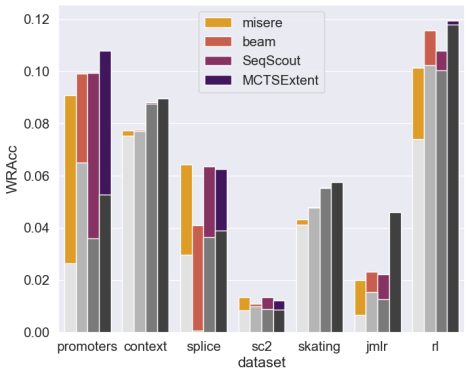


Figure 4.22: Added value of local optima search for 1,000 iterations

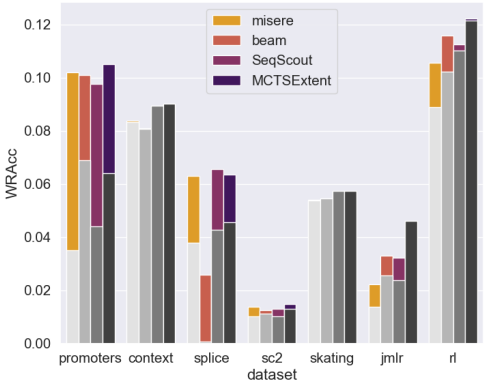


Figure 4.23: Added value of local optima search for 3,000 iterations

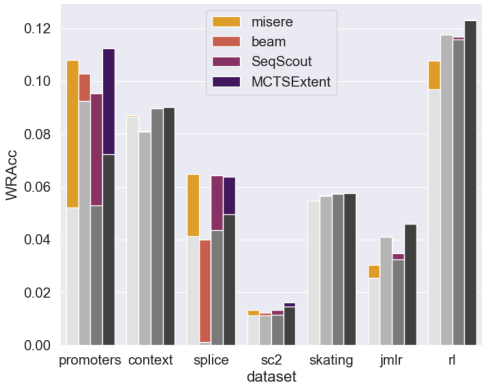


Figure 4.24: Added value of local optima search for 6,000 iterations

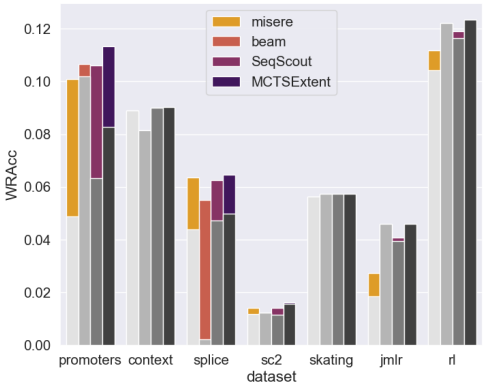


Figure 4.25: Added value of local optima search for 10,000 iterations

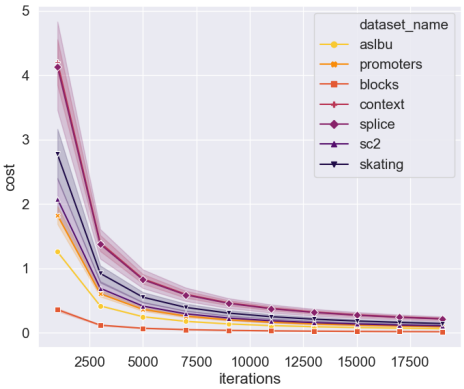


Figure 4.26: Additional cost of local optima search

Interestingly, we can note that the method of `MCTSExtent` can be easily adapted to other pattern description languages, by redefining the "ROLLOUT" step (or generalization) and the "Common pattern" step, i.e., the Longest Common Subsequence here.

Chapter 5

Mining Interval patterns in high dimensional numerical data

5.1 Introduction

In this chapter, we address the problem of supervised rule discovery in time series as the challenging problem of *supervised rule discovery in high-dimensional numerical data*. Indeed we can transform the original time series into potentially high-dimensional numerical data, and then take advantage of the numerical pattern interpretability.

Exhaustive algorithms for categorical data are applied on discretized data or directly on numerical data with greedy discretization during the exploration (`SD-MAP` [5]). Greedy algorithms such as beam search (e.g., `DSSD` [79]) are widely used and have been popularized by the "Cortana" platform[6]. Finally, two anytime algorithms have been recently proposed, attempting to take the best of both paradigms: a best first search that explores the whole search space if given enough time budget and which produces a result anytime keeping improving over time. First, Monte Carlo Tree Search (MCTS) can explore the whole interval pattern search space focusing on both unvisited and promising parts thanks to an exploration-exploitation trade-off (`MCTS4DM` [22]). Second, an exhaustive interval pattern search can be iteratively operated on finer data discretizations (`Refine&Mine` [16]). Note that in the case of high-dimensional numerical data, both `MCTS4DM` and `Refine&Mine` are inefficient. `MCTS4DM` considers interval patterns in a top-down fashion sampling the search space with random draws that are costly, and thus, it hardly reaches lower parts of the search space (where interesting patterns can be hidden). `Refine&Mine` runs fast on very rough data representation, but not when it reaches some finer representations for high dimensional numerical data.

Our contributions can be summarized as follows:

- We extend previous works on *the closed on the positive* (COTP) interval patterns [52, 16, 15] by defining and proving a key property, allowing to explore COTPs only without loss of the relevance of the search space for discriminating the target class.
- We propose a MCTS-based algorithm called `MonteCloPi` that performs a bottom-up best first search exploration based on the above mentioned mathematical property.

[6] `http://datamining.liacs.nl/cortana.html`

- Through an extensive set of experiments, we demonstrate that `MonteCloPi` can be applied to collections of multivariate time series of both equal and different lengths and show that the extracted interval patterns enable to derive competing interpretable classification models.

The chapter is organized as follows. We formalize our rule discovery task in Section 5.2. We define COTP interval patterns and discuss one of their properties in Section 5.3. Section 5.4 describes our algorithm and some related work is discussed in Section 5.5. A quantitative experimental study is reported in Section 5.6 before concluding.

5.2 Supervised Rule Discovery in Numerical Data

Let us again formalize our problem as in instance of Problem. 1. **Database objects** of a numerical dataset with n attributes are points in the n-dimensional Euclidean space. Interval **patterns** are multi-dimensional intervals, i.e., boxes or hyper-rectangles with sides parallel to the plane axes [70]. The **covering function** between patterns and database objects is given in Definition. 21. The extent of an interval pattern is thus the set of data points that fall within this rectangle. The **binary relation** between patterns is the interval inclusion, structuring the search space.

Definition 20 (Numerical dataset $\mathcal{D}(\mathcal{O}, \mathcal{A}, \mathcal{C})$) *Let $\mathcal{O}$, $\mathcal{A}$ and $\mathcal{C}$ be respectively a set of objects, a set of numerical attributes, and a set of class labels. When an object o takes a value v for an attribute $attr$, we write $attr(o) = v$. The finite domain of a numerical attribute $attr \in \mathcal{A}$ in a given dataset is denoted by $Dom(attr)$: it is the set of distinct values it takes in the dataset. The mapping $f : \mathcal{O} \mapsto \mathcal{C}$ associates each object to a unique class label.*

Definition 21 (Interval pattern and covering function) *An interval pattern of length n is a vector of intervals $p = \langle I_1, I_2..., I_n \rangle$, with $I_i = [a_i, b_i]$ $a_i, b_i \in \mathbb{R}^2$ $a_i \leq b_i$. An object $o \in \mathcal{O}$ is covered by an interval pattern i.f.f: $\forall I_i \in p$, $attr_i(o) \in I_i$.*

Definition 22 (Interval pattern ordering and search space) *The search space of all interval patterns is ordered with the inclusion as the **binary relation** of this problem: $p_1 \sqsubseteq p_2 \iff [a_i^2, b_i^2] \sqsubseteq [a_i^1, b_i^1], \forall i \leq n$. It is composed of $\prod_{i=1}^{n} \frac{|Dom(attr_i)| \times (|Dom(attr_i)|+1)}{2}$ elements.*

A subgroup is composed of a pattern, its extent, and a score that reflects its interestingness such as the Weighted Relative Accuracy [76]. When the context is clear, we use the terms pattern and subgroup interchangeably.

Example 1 *Table 5.1 provides a numerical dataset with 4 objects, 6 attributes and 2 labels. $p = \langle [1, 2], [4, 5], [3, 4], [7, 8], [5, 7], [8, 9] \rangle$ is an interval pattern whose support is 2. Accordingly $(o1, o3)$ is the subgroup w.r.t. p and $WRAcc(p, \mathcal{D}, \mathcal{D}_+) = 0.25$. As example for ordering relation, we have $\langle [1, 4], [2, 4] \rangle \sqsubseteq \langle [1, 3], [3, 3] \rangle$.*

Note that the minimal and maximal values of the $WRAcc$ depends on the proportion of the target class in the dataset. Using directly its value to assess its discriminating power could then bias user interpretation. For example, a $WRAcc$ of 0.09 for a class which is represented at 10%

Table 5.1: Toy dataset

ID	$attr1$	$attr2$	$attr3$	$attr4$	$attr5$	$attr6$	$class(.)$
o1	1	5	3	8	7	9	+
o2	8	9	0	1	2	1	-
o3	2	4	4	7	5	8	+
o4	5	5	1	2	3	2	-

would mean that the pattern covers all elements having this class, and only them, i.e., it would be somehow the best pattern. However, on a dataset with a representation of this class at 50%, it would be much less relevant. We then propose to normalize the $WRAcc$.

Definition 23 (Normalized Weighted Relative Accuracy) *The Normalized Weighted Relative Accuracy, or NWRAcc, takes its values in* $[-1, 1]$ *and is defined by:*

$$NWRAcc(p, c) = \frac{WRAcc(p, \mathcal{D}, \mathcal{D}_c)}{WRAcc_{max}(\mathcal{D}, \mathcal{D}_c)}$$

$NWRAcc$ can be used to compare patterns that discriminate different classes of different sizes.

Lemma 2 *The maximum value the WRAcc can take on a dataset* $\mathcal{D}$ *for a class* c*,* $\mathcal{D}_c$ *being the set of its elements labeled by class* c*, is* $WRAcc_{max}(c) = \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\left(1 - \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\right)$.

Proof 3

$$WRAcc(p, \mathcal{D}, \mathcal{D}_c) = \frac{supp(p, \mathcal{D})}{|\mathcal{D}|} \times \left(\frac{supp(p, \mathcal{D}_c)}{supp(p, \mathcal{D})} - \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\right)$$

Note that $supp(p, \mathcal{D}) = supp(p, \mathcal{D}_c) + supp(p, \overline{\mathcal{D}_c})$*. We then have:*

$$WRAcc(p, \mathcal{D}, \mathcal{D}_c) =$$
$$\frac{1}{|\mathcal{D}|} \times \left(supp(p, \mathcal{D}_c)(1 - \frac{|\mathcal{D}_c|}{|\mathcal{D}|}) - supp(p, \overline{\mathcal{D}_c})\frac{|\mathcal{D}_c|}{|\mathcal{D}|}\right)$$

To maximise the WRAcc, the rightmost term must be minimized. This is the case for $supp(p, \overline{\mathcal{D}_c}) = 0$*: it means the pattern does contain only element with positive class. Notice also that* $supp(p, \mathcal{D}) = \lfloor \beta |\mathcal{D}_c| \rfloor$*, with* $0 \leq \beta \leq 1$ *as the number of elements having the target class must be a subset of all elements having the target class. We then have:*

$$WRAcc_{max}(\mathcal{D}, \mathcal{D}_c) = \beta \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\left(1 - \frac{|\mathcal{D}_c|}{|\mathcal{D}|}\right)$$

Directly, β *needs to be set to 1 to maximize the WRAcc.*

5.3 Closed on the Positive Interval Patterns

It has been shown that our problem is equivalent to the discovery of *closed on the positive* patterns (COTP) [52] as a general case, and to the discovery of *closed on the positive interval patterns* within numerical data [16, 15]. The key idea is that any sub-interval of a COTP will drop at least one positive object, thus, reducing the *true positive rate*, hence many measures such as $WRAcc$ (see [16] for details).

Definition 24 (MEET operator) *For* $\forall p, q \in \mathcal{P}^2$, *the MEET operator* $\sqcap$ *is defined as* $\sqcap(p, q) = \langle I_1, ..., I_n, \rangle$ *with* $I_i = [min(a_i^p, a_i^q), max(b_i^p, b_i^q)]$. *For example* $\langle [1, 2], [6, 7] \rangle \sqcap \langle [1, 1], [8, 9] \rangle = \langle [1, 2][6, 9] \rangle$.

Proposition 1 *The MEET of two COTP interval patterns is a COTP.*

Proof 4 *For any interval pattern* p' *s.t.* $p \sqsubseteq p'$, *where* p *is the MEET of two closed on the positive interval patterns, we have a restriction (i.e., a smaller interval) on at least one interval compared to* p. *As bounds of those intervals are extracted from values of COTP patterns, i.e., they are as "tight" as possible, any restriction on it would lead to a decrease of the extent of positive elements of* p. *By definition,* p *is then a COTP interval pattern. Note that any "widening" of an interval would create an interval pattern* p''. *If* p'' *could have the same extent as* p, *we would have* $p \sqsubseteq p''$ *and* $ext^+(p) = ext^+(p')$, *which is a contradiction with the definition of a COTP pattern for* p: *the MEET is unique by definition.*

5.4 MonteCloPi: Monte carlo tree search on Closed on the PosItives

5.4.1 Applying MCTS in a bottom-up way on interval patterns

Classical lattice exploration methods, like `Beam Search`, are top-down by nature, going from the most general description (i.e., pattern) down in the search space through successive refinements. In contrast, we suggest to use a *bottom-up* strategy. Thus, the idea is to consider positive objects of the database, i.e., objects labeled with the target class, and to compute their MEET with other positive objects, successively climbing the search space of COTP, containing discriminative patterns. We aim at creating patterns that are the most restrictive descriptions that cover a set of chosen positive elements, i.e., they are by definition discriminating the positive class. We then use a Monte Carlo Tree Search (MCTS) strategy. The idea is to iteratively explore the search space using sampling and provide an *anytime* property. We propose an algorithm MonteCloPi (MONTE carlo tree search for CLOsed on the PosItives). MCTS typically consists of 4 steps (i.e., SELECT, EXPAND, ROLLOUT, UPDATE) that are repeated successively. Its idea is to grow an asymmetric tree following an exploration-exploitation trade-off to quickly find interesting area of the search space. Hereafter, we explain these generic steps and how we instantiate them to our problem. The schema describing MonteCloPi is given in Fig 5.1, and its pseudocode is given in Alg. 5. Note that each node of the tree is composed of a set of database objects, i.e., its extent, and a pattern covering those objects.

5.4.2 SELECT Policy: numerical object selection

The first step consists in selecting a non fully-expanded node. The idea is to choose the most relevant area of the search space to explore. To do so, we use the UCB function (see equation 3.2) to guide us through the tree, using an exploration-exploitation trade-off. We begin with the $\perp$ node, being the starting point node of the search having all database objects as children. We compute the UCB of all its children and select the one maximizing it. We perform this step recursively until reaching a non fully-expanded node. Note that UCB gives a better score to nodes (i.e., patterns with their extent) that are good patterns, but also to nodes that are less explored.

Algorithm 5 MonteCloPi algorithm

```
1: function MonteCloPi(timeBudget)
2:     π ← PriorityQueue()
3:     create ⊥ initial node having all objects as children
4:     while within timeBudget do
5:         p_sel ← Select(⊥)
6:         p_exp, qual_exp ← Expand(p_sel)
7:         p_roll, Δ ← Rollout(p_exp)
8:         Update(p_exp, Δ)
9:         π.add(p_exp, qual_exp)
10:        π.add(p_roll, Δ)
11:    end while
12:    return π.topK()
13: end function
14:
15: function Select(p)
16:    while p is not ⊤ do                    ▷ ⊤ is the most general pattern
17:        if p is not fully-expanded then
18:            return p
19:        else
20:            p ← BestChild(p)
21:        end if
22:    end while
23:    return p
24: end function
25:
26: function Expand(p)
27:    o+ ← randomly choose a positive object
28:    p+ ← IntervalPattern(o+)               ▷ Transform each a_i into [a_i, a_i]
29:    p_exp ← meet(p, p+)
30:    return p_exp, φ(p_exp)
31: end function
32:
33: function Rollout(p)
34:    j ← Rand(1, n)          ▷ Randomly pick an index of one of n intervals constituting p
35:    for i in [1, j − 1] ∪ [j + 1, n] do    ▷ Exclude the j^th interval I_j of p, i.e., p \ I_j
36:        p[i] ← [−∞, +∞]
37:    end for
38:    return p, φ(p)
39: end function
40:
41: function Update(p, Δ)
42:    while p ≠ ⊥ do
43:        φ(p) ← (N(p) ∗ φ(p) + Δ) / (N(p) + 1)   ▷ N(p) is the number of times p has been chosen
44:        N(p) ← N(p) + 1
45:        p ← parent of p
46:    end while
47: end function
48:
49: function BestChild(p)
50:    return argmax_{p′∈C(p)} UCB(p, p′)      ▷ C(p) is the set of children of p
51: end function
```

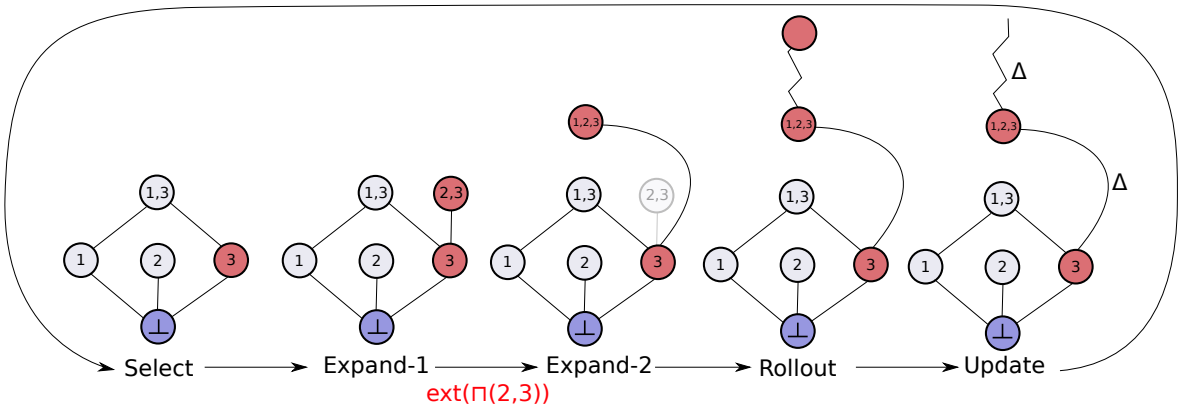


Figure 5.1: Simple illustration of MonteCloPi steps

Note that in Alg. 5, $\top$ corresponds to the most general pattern, i.e., the empty pattern present in all database objects. In Fig. 5.1, the blue node is the initial bottom node $\bot$. It is fully expanded, so the SELECT procedure will choose its best child w.r.t. UCB. Suppose the node "3" has the best UCB, and has not been fully expanded, then, it is selected (given in red).

5.4.3 EXPAND Policy: meet with positive object

Then, the selected node is expanded. This step consists in taking a positive element (o_+ in Alg. 5), i.e., an object labeled with the target class, transforming it to an interval pattern, and computing its MEET with the interval pattern of the current node. Next, the extent of this MEET is computed: we create a pattern with the most restrictive description that covers elements from the selected node plus this positive element. Note that the transformation step from object to pattern is straightforward: each attribute $attr_i(o)$ becomes an interval $[attr_i(o), attr_i(o)]$.

In Fig. 5.1, we take the node covering the positive element "3", add the positive element "2" (node "2,3"). We then compute the MEET of their pattern, its quality, and finally the extent of this MEET: it covers elements "3", "2", but also "1" (node "1,2,3"). Note that the pattern obtained from an object labeled with the target class o_+ is COTP, and the first iterations of the current procedure consider single objects descriptions, i.e., COTP patterns. From Proposition 1, it follows that this EXPAND step recursively creates a new COTP pattern.

5.4.4 ROLLOUT policy: interval pattern generalization

The ROLLOUT step here consists in taking the corresponding interval pattern of the expanded node, randomly choosing an interval among n available ones (I_j), and removing restrictions on all others, i.e., setting them to $[-\infty, \infty]$. We then widen this interval by picking a random upper and lower value among possible ones in the dataset. This corresponds to going "up" in the search space, until reaching the first level of the search space in a top-down approach (see the top red node in Fig. 5.1). We then compute the quality of this pattern. Note that Monte Carlo strategies consist in drawing a large number of samples, so they require a fast ROLLOUT policy to be efficient.

5.4.5 UPDATE

Finally, we update parent nodes of the ROLLOUT node with its computed quality Δ. These parent nodes include the expanded node, the selected node and all nodes chosen in the SELECT step until reaching the selected node. The UCB will orient the tree exploration towards the nodes whose ROLLOUTs give good qualities, so they will be prioritized on the next iterations of MCTS, leading to the discovery of interesting patterns.

Steps 1-4 are repeated iteratively until the time budget has not been reached or until the whole search space is explored.

The time complexity of MonteCloPi is proportional to the number of iterations it performs ($IterNum$). For one iteration, the most costly task is the computation of the quality measure, in the step 3 and 4. This complexity is $O(n|\mathcal{D}|)$, with n the number of attributes of time series. The memory complexity is $O(IterNum)$, as the whole tree needs to be held in memory.

Theorem 3 *Our described enumeration method is equivalent to exploring all closed-on-the-positive interval patterns.*

Proof 5 *We have shown that the EXPAND step creates a COTP. We now need to show that $\forall c \in COTP$, c will be enumerated by our method. By construction: let c be a COTP interval pattern. If we remove positive objects from $ext(c)$ one by one, and compute the corresponding COTP interval pattern, we create a list of COTP interval patterns until we reach a COTP interval pattern containing only one object (or a set of identical objects). This list of successive COTP interval patterns is exactly the list built by our method in reverse order, by adding objects one by one and computing the meet in Expand step.*

As a COTP interval pattern created from the meet of two COTP interval patterns is unique (see Proposition 1), our enumeration method will create c.

5.4.6 Adaptation of MonteCloPi for Time Series of Different Lengths.

To deal with time series of different lengths that we can collect in many application settings, two small modifications of the algorithm are required. First, we adapt the computation of the MEET for two interval patterns: we randomly remove intervals from the longest one to have two interval patterns of the same length, and then compute their MEET. Second, we re-define the notion of *coverage*: a time series transformed into a numerical object o of length n_{ts} is said to be covered by an interval pattern p of length n_p iif: $\exists 1 \leq i_1 \leq \ldots \leq i_{np} \leq n_{ts}$ *s.t.* $attr_{i1}(o) \in I_1, \ldots, attr_{inp}(o) \in I_{np}$. Note that we loose here the guarantee of exhaustiveness on COTP, as the MEET operator is no more unique.

To summarize, MonteCloPi is an *anytime* algorithm. It explores *COTP patterns*, with a guarantee of *exhaustiveness*, if given enough time budget. It is agnostic of the quality measure. In contrast to `MCTS4DM`, it explores the search space in a bottom-up way, exploring only COTPs by redefining in an original way the EXPAND and the ROLLOUT steps.

5.5 Related Work

Mörchen and Ultsch [93] proposed a new language (TSKR) on interval time series. The addressed problem is different from ours: events have a duration instead of being instantaneous

(e.g., injection of a drug in medical conditions), and they look for frequent rather than discriminative patterns. Similarly, Batal *et al.* [12] worked on predictive frequent time-interval patterns. Atzmueller *et al.* [108] propose a method for the classification of time series. They discretize data with SAX, incurring information loss. Features are extracted with the FRESH algorithm from raw time series, and anomaly detection is done to improve the overall workflow. It requires human-in-the-loop. Nanlin Jin *et al.* used SD in Smart Electricity Meter Data [67]. Using `Beam Search`, they explored different quality measures on data similar to time series but also containing categorical data. They showed the relevance of a SD approach to create interpretable rules used further in classification in the context of energy consumption. They claim that "The usefulness and effectiveness of subgroup discovery algorithms need to be empirically evaluated through their predictive power and classification accuracy, both being important to industrial practitioner". Nguyen *et al.* [96] propose a method that learns a linear classifier on discretized data (SAX or SFA). A feature (pattern) with the largest weight is considered the most discriminating. The goal is first to classify, then to extract patterns from the model. They use a smart pruning strategy to minimize the loss function, dynamically selecting features. However, such an approach ignores the quality measure (e.g., it is impossible to specify to $Precision$ or $Lift$ maximization). Notice also that they cannot process multivariate time series. Top-down approaches using tight optimistic estimates like the recent work of Millot *et al.* [90], or using those optimistic estimates to prune the search space for `Beam Search` can also be considered. However, they are dependent on the quality measure. Besides, using optimistic estimates when looking for a non-redundant pattern set, where non-redundancy is performed as post-processing, is quite difficult. To avoid pruning interesting search space areas, it would require to know the quality of the last top-k non-redundant patterns w.r.t. the quality measure at any time. It would have a high computational cost. `misère`, by Egho *et al.* [44] has been proposed to sample interesting rules (originally in the case of sequences of itemsets) in a simple way, giving good results when those rules are used to build a classifier. Genetic algorithm can be considered to tackle this problem, but they do not offer strong guarantees like exhaustiveness or exploring only COTPs. For example, Lucas *et al.* proposed `SSDP` to mine top-k discriminative rules in high-dimensional data [84]. However, they only consider categorical and discrete attributes.

Considering the classification of time series, most of the recommended methods (e.g., [9]) require time series to have the same length, whether it is BOSS [115], COTE [10], Shapelet Transform [62], or Time Series Forest [36]. In the use case presented later on, time series have different lengths. The 1-Nearest Neighbor Dynamic Time Warping `1NN DTW` [94] is said to be a good baseline. However, the prediction step in K-NN can take a long time, even more because the DTW distance has a complexity of $O(n^2)$ for time series of size n.

5.6 Quantitative Experimental Study

We conduct a series of experiments to assess the performance of MonteCloPi. In the first set of experiments, we focus on supervised rule discovery in time series of equal lengths.

5.6.1 Datasets

We consider benchmark datasets of time series domain [11]. We add our original dataset *RocketLeague* containing annotated video game traces. We describe its generation process in chapter 6. We transform time series data into numerical data similarly to approach [67]. Basically,

a multivariate time series of fixed length dataset is composed of a set of variables X at each timestamp in T. The corresponding numerical dataset is then given by $(\mathcal{O}, X \times T, \mathcal{C})$: each numerical attribute gives the value of a variable at a given time. The properties of the resulting datasets are summarized in Table 5.2.

5.6.2 Baselines

To provide a fair evaluation, we test MonteCloPi against the following algorithms:

- `Beam Search` [42]: Starting from the most general subgroup, i.e., containing intervals in the form $[min(Dom(attr)), max(Dom(attr))], \forall attr \in \mathcal{A}$, it explores the search space level-wise, taking the best elements (beam width) at each level. Each restriction (going down by one level) consists in taking an interval and splitting it in 6 equal-sized bins []. It prevents the search space from exploding, but forfeits exhaustiveness. We set the beam width to 50. Other settings can be used, but they do not impact significantly the reported results.
- `misère` [44]: It draws uniformly an object from the data, and then generates random generalizations to find the best possible prediction rules for this object. It iterates until the time budget is exhausted. Notice that this is an exploration-only search algorithm.
- `MCTS4DM` [22]: It is a MCTS-based top-down approach that processes numerical data without discretization. Its strategy guides the search following an exploration-exploitation trade-off to compute non-redundant patterns. It explores the lattice of patterns, contrary to MonteCloPi that explores the lattice of extents.

We also tried to use `Refine&Mine` [16]. This algorithm dedicated to numerical subgroup discovery was able to provide only extremely rough initial discretization on all the datasets. We decided not to report further.

For each of the algorithms, we apply the following procedure. Given a time budget, each algorithm extracts the best patterns. Next, the following post-processing routine is applied for removing redundant patterns: we sort patterns by quality, keep the best, while removing similar patterns (in terms of Jaccard index) of poorer quality. Then we take the second best, etc, until we obtain k patterns or there is no pattern left.

Table 5.2: Datasets statistics.

Dataset	$\lvert\mathcal{D}\rvert$	n	m	$\lvert(C)\rvert$	Multivariate	Search Space Size
Computers	250	720	1	2	False	$1.31 * 10^{9}$
Earthquakes	322	512	1	2	False	$1.02 * 10^{11}$
ECG5000	500	140	1	5	False	$3.41 * 10^{11}$
Gunpoint	50	150	1	2	False	$4.16 * 10^{9}$
Lightning2	60	637	1	2	False	$3.20 * 10^{11}$
Worms	182	900	1	5	False	$2.37 * 10^{12}$
Yoga	300	426	1	2	False	$3.46 * 10^{12}$
Handwriting	150	152	3	26	True	$2.85 * 10^{25}$
JapaneseVowels	270	29	12	9	True	$6.38 * 10^{84}$
NATOPS	180	51	24	6	True	$3.88 * 10^{184}$
RacketSports	151	30	6	4	True	$1.77 * 10^{41}$
RocketLeague	298	64	8	18	True	$1.44 * 10^{55}$
StandWalkJump	12	2,500	4	3	True	$1.26 * 10^{24}$
UWaveGestureLibrary	2,238	315	3	8	True	$7.07 * 10^{27}$

If not specified otherwise, the default parameters for the algorithms are the following: 60s of time budget, $\theta = 0.8$, $top\_k = 5$. All experiments were conducted on a laptop (Intel Core i7-8750H CPU and 16GB RAM).

Quality Measure We explore performances of algorithms on different datasets, and study the impact of varying available time budget. In the following, if not stated otherwise, we use $NWRAcc$ as the quality measure. We average it over top-5 non-redundant patterns of 5 runs of the algorithms. Note that our approach is agnostic of the quality measure, and that any other discriminative quality measure could be considered.

Classification To assess the usefulness of MonteCloPi, we evaluate the classification accuracy of a classifier built upon mined patterns. We propose an instance of the `LeGo` framework [72], where a set of interesting patterns is mined and then used as features to build a global model. Mind that our goal here is to analyse the interest of this pattern mining approach to build intelligible models. Therefore, creating a time series classifier to compete with the state-of-the-art algorithms is out of the scope of this work. First, we launch MonteCloPi on each target class in a one-*vs*-rest fashion. We then re-encode the dataset using extracted patterns: each pattern corresponds to a binary feature, with value 1 set if the pattern appears in the transaction, and 0, otherwise. Finally, we train a Random Forest classifier of 100 trees on this re-encoded dataset to evaluate the performance of this pipeline. Note that, as explained by Knobbe *et al.* in [72], the non-redundant step is important to remove redundant features that tend to hinder machine learning procedures.

In the case of univariate time series, we compare our classifier to `Mr-SEQL` [96], and `1NN DTW` [9]. The latter is considered to be a strong baseline that is hard to beat [9]. For the multivariate case, we choose to compare to `1NN DTW` that remains a strong baseline. Notice that `Mr-SEQL` is unable to deal with multivariate time series.

5.6.3 Overall Performance

First, we examine the overall performance of MonteCloPi in terms of $NWRAcc$ compared to `Beam Search`, `misère` and `MCTS4DM`. In Fig. 5.2, we plot the results for all datasets. As we can see, MonteCloPi is the best approach on nearly all the datasets. `Beam Search` may also be a good solution, as the algorithm is simple, and results are quite stable over datasets. However, our approach clearly outperforms it in the majority of cases. As for `misère`, its results vary significantly depending on the dataset. For instance, it performs very well on **GunPoint**, while returning patterns having a zero quality on **UWaveGestureLibrary**. Finally, `MCTS4DM` that performs a MCTS-based approach but in a "top-down" way on patterns clearly gives less interesting results on such datasets.

5.6.4 Varying Time Budget

We traced the performance of MonteCloPi, `Beam Search`, and `misère` versus time budget in Fig. 5.3. As we can see, MonteCloPi quickly gives better results, improving over time. Moreover, those results are quite stable over runs of the same duration, contrary to `misère`. Our hypothesis is that this is due to the exploration-exploitation trade-off, where we focus on interesting areas of the search space, whereas `misère` performs a pure exploration. Another

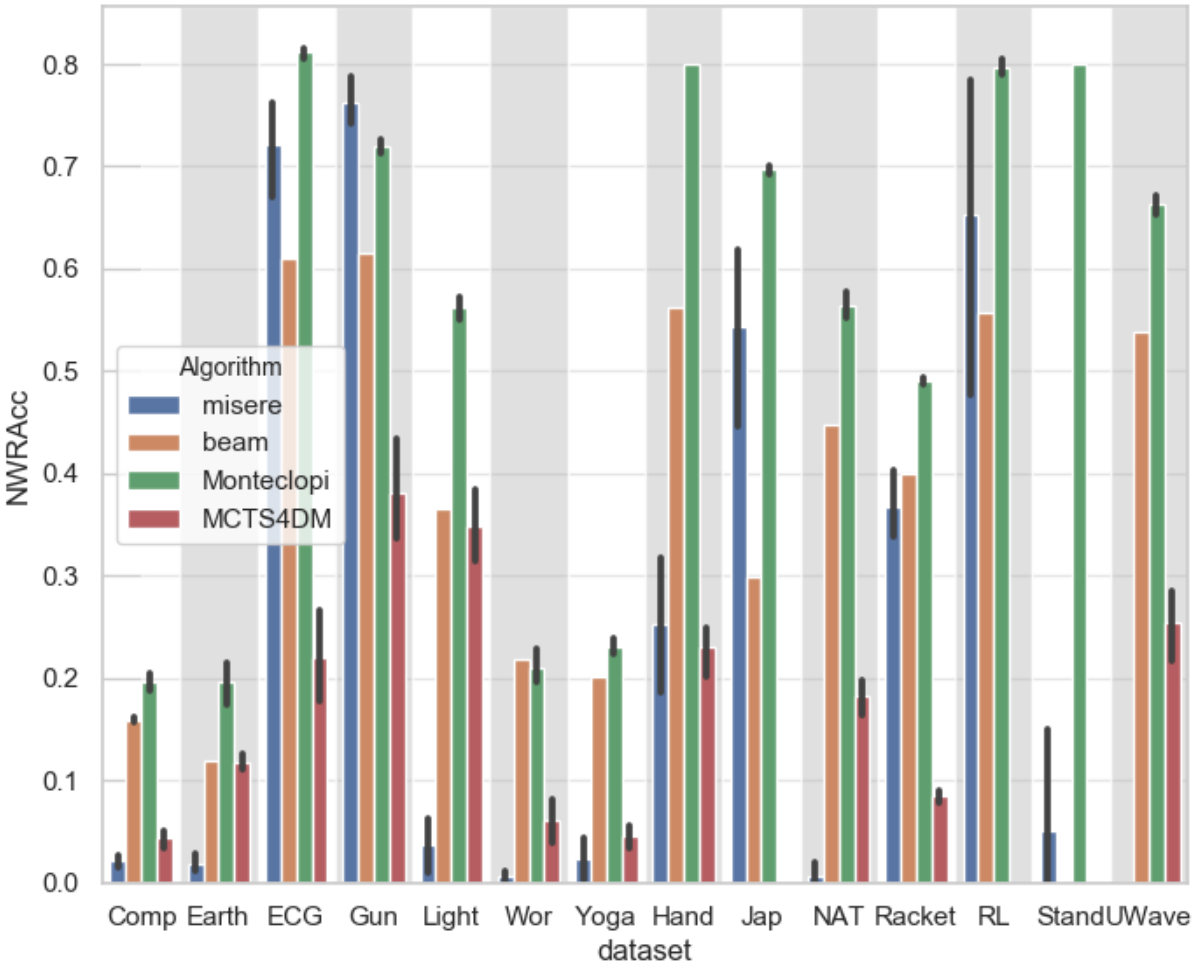


Figure 5.2: Mean *NWRAcc* of top-5 patterns with different algorithms. Bar errors represent the 95% confidence intervals

problem faced by `misère` is that for this type of data, the number of patterns one can create from a time series is very large. It results in a strong dependency of the quality of a pattern on the object that was randomly drawn. It explains the large variability of the results for `misère`.

5.6.5 Classification Performance

First, we consider univariate time series. We set the time budget to 60s to mine patterns for each class. We also compare classification performances using patterns from `misère` and `Beam Search`. The results in terms of accuracy are given in Table 5.3. Unsurprisingly, as quality of patterns from `MonteCloPi` were better, resulting classifier outperforms those build upon patterns of `misère` and `Beam Search`. As we can see, the MonteCloPi pipeline achieves results comparable to the state-of-the-art classification algorithms on most datasets, except **Worms** and **Yoga**, where the results are somewhat inferior. However, it should be noted that our method achieves these good results within the training time of only few minutes, contrary to `1NN DTW`, which takes several hours to finish in our implementation (for the comparison given

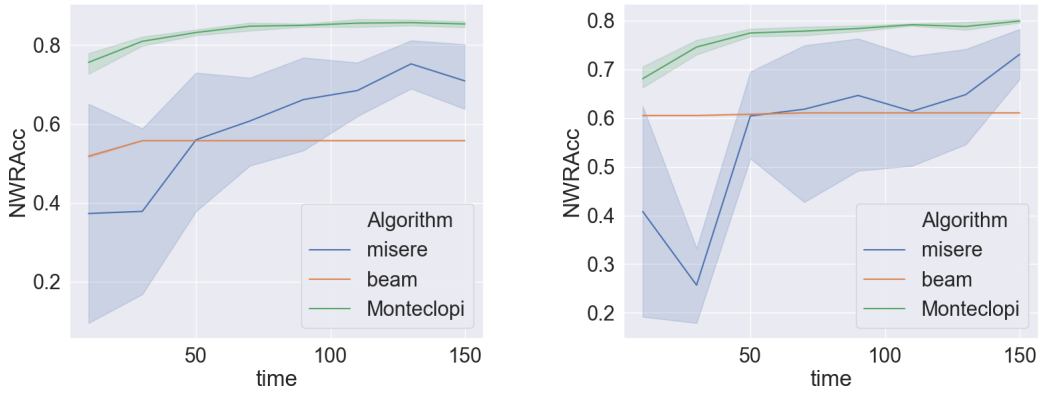


Figure 5.3: *NWRAcc* vs *time budget* on **RocketLeague** (left) and **ECG5000** (right) averaged over 5 runs. The shaded areas represent the 95% confidence interval.

in Table 5.3, we used the results of `1NN DTW` reported in [9]), and more importantly, once the classifier is trained it can classify in real time. Results of `Mr-SEQL` are very good, however, this algorithm is focused on classification and not on pattern mining: we cannot control the type of patterns we are interested in with a quality measure, the algorithm is not anytime, and it does not work on multivariate time series.

Let us now consider multivariate time series. As we can see in Table 5.4, the results of MonteCloPi are overall quite good, except for the **HandWriting** dataset. However, contrary to `1NN DTW`, our method trains a model that gives instant results at the classification step (a few milliseconds), when `1NN DTW` can take much longer (over fifteen minutes), just like in the univariate case. Therefore, in a context where data are multivariate time series of different lengths, and where a real time classification is required, as in the case of the *Rocket League* use case described in next chapter, `1NN DTW` cannot be used, and the MonteCloPi pipeline becomes particularly relevant.

5.7 Conclusion

We presented MonteCloPi, an efficient solution for subgroup discovery in numerical data. It is generic and can be easily adapted to other types of data. To do so, only two modifications are required: redefinition of the MEET operator and adaptation of the ROLLOUT step. For

Table 5.3: Classifiers accuracy for univariate datasets

Dataset	MonteCloPi	Mr-SEQL	1NN DTW	Beam	Misere
Computers	0.656	0.720	0.659	0.612	0.596
Earthquakes	0.748	0.748	0.747	0.748	0.748
ECG5000	0.918	0.934	0.940	0.891	0.912
Gunpoint	0.940	0.980	0.964	0.833	0.953
Lightning2	0.688	0.672	0.710	0.606	0.540
Worms	0.519	0.623	0.673	0.467	0.428
Yoga	0.692	0.871	0.863	0.720	0.540

instance, we can examine the possibility of applying MonteCloPi to graphs.

MonteCloPi can deal with multivariate time series of different lengths, and be used to classify them in real time, providing valuable interpretable features. Though we reported results on a video game skill classification use case, various industrial processes can benefit from our findings. Time series are ubiquitous, and the need for interpretable rules is constantly growing.

Moreover, the property of exhaustiveness on COTP, that contains discriminating patterns and are a subset of all possible patterns, gives the guarantee of finding the best elements and automatically stopping the algorithm, if given enough time. However, this property does not remain valid for the case of time series of different lengths, in a same manner as MCTSExtent. To overcome this limitation, one could compute all possible maximal common patterns between two elements in the EXPAND step. More investigations are needed, as the number of these patterns could be significantly high depending on the pattern description language.

Table 5.4: Classifiers accuracy for multivariate datasets

Datasets	MonteCloPi	1NN DTW	Beam	Misere
Handwriting	0.161	0.316	0.095	0.101
JapaneseVowels	0.932	0.959	0.173	0.873
NATOPS	0.833	0.850	0.667	0.167
RacketSport	0.809	0.842	0.618	0.816
RocketLeague	0.801	0.574	0.482	0.726
StandWalkJump	0.333	0.333	0	0.333
UWaveGestureLibrary	0.840	0.868	0.584	0.125

Chapter 6

Application to Game Analytics: player behaviour detection

6.1 Introduction

Competitive gaming, or esports, is now a well-established phenomena [122]. Online and offline tournaments flourish for hundreds of video games, at any level of player expertise. The most prestigious offer cash prizes up to several US$ millions, and are widely followed on video game live streaming platforms [71, 123]. For the game industry, designing games that can be played as an esport comes with a difficult trade-off between game difficulty and reward/fun. Indeed, a game shall be difficult enough so that professional gamers exhibit extraordinary skills that casual players will enjoy to watch on live streaming platforms. However, the game should also provide an attractive learning curve and clear segments of skills. One should play with/against players of approximately the same level of skills. They finally should also feel a progress in the learning of the game. As popularized by the famous board game Othello's slogan, many esports take "A minute to learn, a lifetime to master". When the trade-off between game difficulty and reward is well handled, a game has better chances to see its life time extended, a major goal for the industry, especially for games with staggering budgets.

Consequently, a major challenge is to understand the level of a player and to evaluate its progression. Match making systems rank players, generally, following an ELO-like scoring [61], and depending only on previous victories and losses of the player. It suffers from the cold-start problem, but most importantly, it does not bring any factual elements such as a player profile of skills and mastered strategies. Furthermore, skills and strategies are not given beforehand - and this is probably one of the reasons why such games are so attractive - but are discovered with time. Hopefully, several games provide access to game logs storing enough information to replay the game. Analyzing these logs enables one to exhibit behavioral patterns, that is, discovering patterns that correspond to strategies and skills. Logs are generated not only for competitive games, but also training sessions, so a profile can be updated with any game.

In this chapter, we focus on the game Rocket League played as an esport, which can be described as "soccer, but with rocket-powered cars". Released in 2015, it now counts more than 10 millions sales and tens of thousands active players [129]. The game should be played competitively at an international event for the next Olympic Games in Tokyo [65]. Each player controls a car on a soccer field, and has to score just like for classical soccer. The rules are

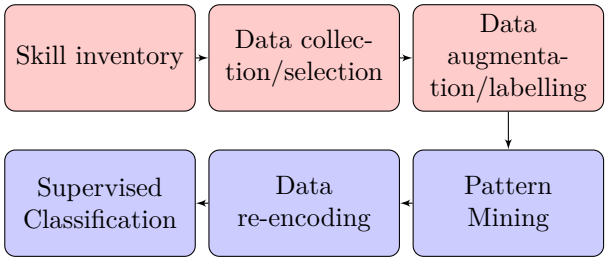


Figure 6.1: Methodology (domain expert interventions in red).

simple, however the control of the car is very precise and requires thousands of hours of play to master. Actually, even years after the release of the game, the community discovers new ways of playing, and new skills that impact the way professionals play. Those skills are recognizable by commentators when games are streamed. Adding a system that can detect them automatically could enhance players ranking, help commentators recognize skills among the spectrum of available ones, or create new game modes based on skills execution (rewarding players given difficulty of skills they executed during the game).

Identifying skillshots from game traces is challenging as each occurrence is unique from a pure data point of view. Therefore, advanced data analysis techniques must be involved. Our contributions are as follows: (i) we provide an original dataset, composed of sequences of *Rocket League* game states (replay), manually labeled with skillshots, and augmented with player actions thanks to a self-made capture program, (ii) we design a non trivial data-centric approach for the automatic detection of skillshots involving discriminant pattern mining based on an adaptation of SeqScout and supervised classification, (iii) we support our claims with an extensive set of experiments. All our experiments, data collection workflow and data are provided with the code of our solution online: `https://github.com/Romathonat/RocketLeagueSkillsDetection`.

This chapter is organised as follows. Section 6.2 describes our data and methodology. Section 6.3 provides related works. Section 6.4 presents experimental results supporting our claims, and Section 6.5 discusses them.

6.2 Data and Methodology

We focus here on building an interpretable prediction model taking a game log in real-time and outputting each time a skillshot is detected. To the best of our knowledge, there exists no dataset of *Rocket League* game logs labeled with skillshots. Thus, we first build a dataset, perform a number of transformations and manually annotate it with skillshots. Then, the resulting dataset is mined and behavioral patterns that strongly characterize skillshots are extracted. Finally, such patterns are used to re-encode the initial dataset, and a model is trained to predict skillshots given unseen player's trace segments. The fact that patterns are used in training enable to produce an interpretable model (in contrast to black box models) which is required to build an interpretable player skill profile. The general workflow is given in Figure 6.1.

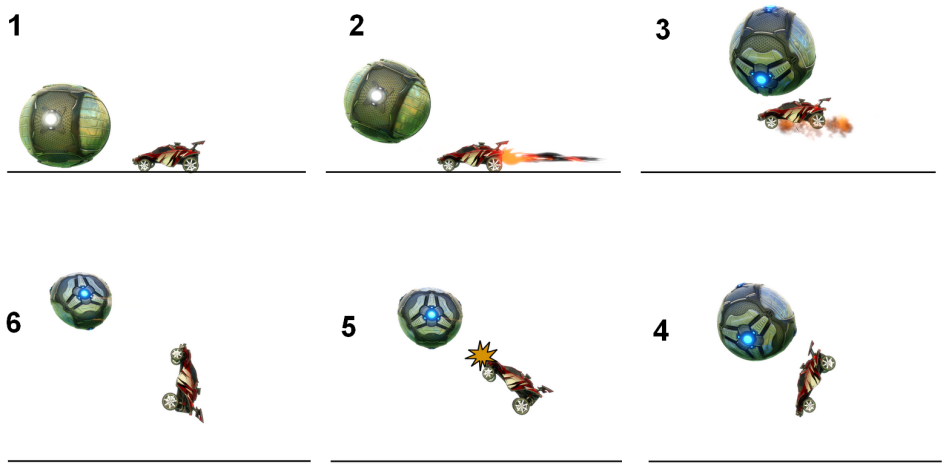


Figure 6.2: Decomposition of the "Musty Flick"

6.2.1 Skill inventory

This expert knowledge can be found on community websites, e.g., [109]. We focus on the most popular skillshots, namely *Ceiling Shot*, *Power Shot*, *Waving Dash*, *Air Dribbling*, *Front Flick*, and *Musty Flick*. Figure 6.2 illustrates the *Musty Flick*: first, the player accelerates and boosts to make the ball roll on top of the car (1, 2), then she jumps making both the car and the ball go up (3). Next, she orientates the car towards the floor (4), jumps again, orientating the car backward (5), resulting in lobbing the ball (6).

6.2.2 Data collection and Feature Selection

After a game of *Rocket League*, the game client stores a replay file. It is composed of contextual information about the cars and the ball for every frame of the game. It allows to replay the game with the game engine at any moment. Replays can be parsed[7] to extract dozens of variables on the ball and players such as a speed vector, positions vector, rotations vector, rotation speed, each composed of 3 dimensions, and this for each actor on the field (players and balls), several times per second. It results in a large sequence of game states, each containing values for a set of variables, e.g., $\langle\{time : 1.256, P_x : 578, P_y : 5768, P_z : 2245, P_{vx} : 22425, P_{vy} : 15848, P_{vz} : 354, P_{rx} : 0, P_{ry} : 589, P_{rz} : 23, Ball_x : 5588, Ball_y : 789, Ball_z : 22, ...\}, \{time : 1.298, P_x : 7578, P_y : 254, P_z : 4678, P_{vx} : 511, P_{vy} : 555, P_{vz} : 7863, P_{rx} : 6365, P_{ry} : 5665, P_{rz} : 6, Ball_x : 568, Ball_y : 8663, Ball_z : 665, ...\}, \{...\}, ...\rangle$, with P_i being a position of the car in dimension i, P_{si} its velocity, R_{ri} its rotation, etc.

In order to deal with such a number of features, we can reduce it with feature selection/engineering, using expert knowledge. As an example, using the information of the position of a static boost pad (an item on the field giving boost to players) will not help to classify the action the player is performing. The list of relevant information is given in Tab. 6.1. The wall distance,

[7] `https://github.com/jjbott/RocketLeagueReplayParser`

ceiling distance and ball distance can easily be computed using positions of the ball and players by retro-engineering the positions of the walls and ceiling in replays of games already played.

Contextual information	Data type
Wall distance (not backboard)	Numeric
Ceiling distance	Numeric
Ball distance	Numeric
Ball speed	Numeric
Ball acceleration	Numeric
Car speed	Numeric
Goal Scored	Boolean

Table 6.1: Contextual information selected by the expert

6.2.3 Merging with inputs data

Unfortunately, *Rocket League* replay files contain only contextual information on the cars and the ball (position, speed, ...), but do not contain player inputs (*turn left*, *accelerate*, ...). To overcome this limitation, during each game, we also gathered player inputs. To do so, we executed a program listening to the game controller (joystick) to detect sequences of buttons, say inputs, a player presses. It then generates a sequence of inputs, for example: $\langle\{accelerate, boost, time : 1.2\}, \{accelerate, right, boost, time : 1.25\}, \{jump, time : 1.34\}, \{accelerate, up, left, time : 1.6\}, \{accelerate, jump, down, boost, time : 1.7\}, ...\rangle$.

This sequence is then merged with the contextual sequence. Indeed, the sequence of player inputs alone is not enough to tell if a skill has been executed. For example, entering the sequence of inputs for a "Power shot" when the ball is far away will just result in flipping the player. On the other hand, contextual information alone does not provide player inputs, which are required to produce interpretable behavioral patterns.

The workflow of the data augmentation process is given in Fig 6.3. The sequence of contextual information (obtained from the game replay on which feature selection is operated) and the sequence of player inputs are merged. Then, in order to improve the computational efficiency of the next steps, we filter out a sequence state X_i (built in the previous subsection) if its previous state X_{i-1} has the same inputs. Indeed, behavioral patterns will be built only on states resulting from player's actions.

An expert then labels sequences with skillshots performed using the in-game replay viewer: she defines the beginning and the end of sequences corresponding to a particular skill, leading to the creation of our labelled dataset. Sequences are then split according to the beginning and the end chosen by the expert (the end is often set when a goal is scored). A simplified example of a labeled sequence is the following (a for *accelerate*, DW for *DistanceWall* etc.): $\langle(\{a, b\}, \{Time : 1.2, DW : 2131, ...\}), (\{\{a, r, b\}, \{Time : 1.25, DW : 1801, ...\}), (\{j\}, \{Time : 1.34, DW : 1325, ...\}), (\{a, u, l\}, \{Time : 1.6, DW : 600, ...\}), (\{a, j, d, b\}, \{Time : 1.7, DW : 233, ...\})\rangle, figure : MustyFlick$.

Note that those steps that may appear straightforward as described here are in fact technically complex. As replays are meant to be replayed and not to extract data, a lot of retro engineering is necessary to adjust x, y, z orientations, and to synchronize sequences from replays and players actions inputs. Note also that when visualizing the replay for labelling, the time is

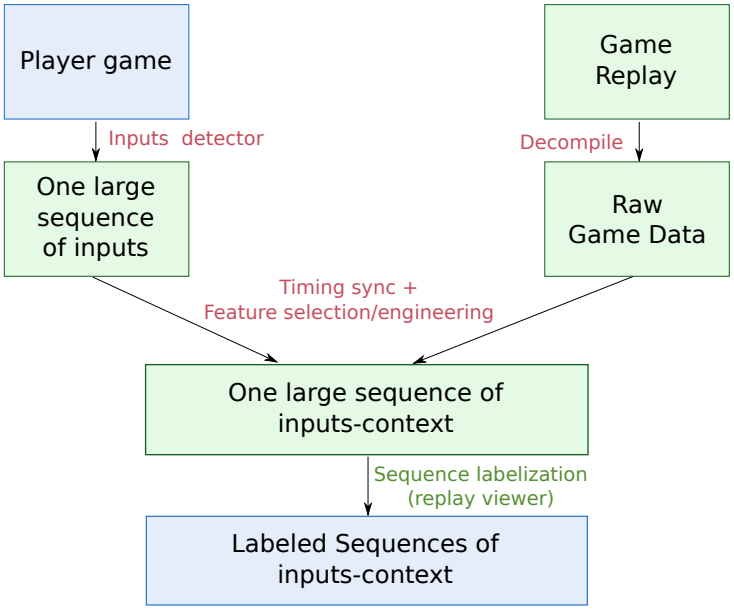


Figure 6.3: Data augmentation

a bit faster than the real play, leading to difficulties of synchronization with timing of inputs that we corrected thanks to a linear regression.

6.2.4 Interesting rules discovery

6.2.4.1 Background

Using supervised rules discovery for sequences has shown to be relevant patterns in the case of game analytic (e.g., [23]). In this subsection, we first formally introduce an original behavioral pattern mining technique, put then in practice with an adaptation of an existing algorithm.

Definition 25 (Complex Event Sequence) *Let $\mathcal{I}$ be a set of items. Each subset $I \subseteq \mathcal{I}$ is called an itemset. A complex event sequence, or database object, is an ordered list of states $o = \langle X_1 ... X_n \rangle$ where each state $X_i = (t_i, I, N)$ is composed of a timestamp t_i, an itemset I and a list of numerical valued variables $N \times \mathbb{R}$. Note that each state is composed of a list of the same numerical variables.*

Each labeled sequence produced in the previous step can be represented as a complex event sequence (Table 6.2).

Definition 26 (Behavioral pattern) *A behavioral pattern is a complex event sequence generalization and can be written as an ordered list of states $p = \langle X_1 ... X_m \rangle$ where each state $X_i = (I, N)$ is composed of an itemset I of player actions and a list of numerical interval valued variables $N \times [a, b]$ with $a, b \in \mathbb{R}$ denoting the contextual information ranges of the event.*

It is precisely the ranges of contextual variables (intervals) that enable behavioral patterns to grasp slight variations of the different executions of a skillshot, along with common subsets of

id	Sequences	class
1	{a,b}, {a,r,b},{j}, {a,u,l}, {a,j,d,b}	+
2	{a,r}, {j}, {j}, {l}	-
3	{a,b}, {a,b},{a,j}, {a,u}, {r}, {j,d}	+
4	{a,b}, {a,b},{a,j}, {j,u}	-

Table 6.2: Toy dataset. "+" means the sequence is a "Musty Flick". Her we only included player inputs for readability.

player actions. Patterns thus represent skillshot generalizations, resistant to noise and variations of execution.

Definition 27 (Covering relation of behavioral pattern) *A behavioral pattern* $p = \langle X_1 ... X_{m_p} \rangle$ *covers a complex event sequence, , i.e., is a generalization of,* $o = \langle X'_1 ... X'_{m_o} \rangle$*, denoted* $p \sqsubseteq o$*, iff there exists* $1 \leq j_1 < ... < j_{m_p} \leq m_o$ *such that* $X_1 \subseteq X'_{j_1}, ..., X_{m_p} \subseteq X'_{j_{m_p}}$*. Here,* $X_i \subseteq X_j$ *means that* $A_i \subseteq A_j$ *and that* $\forall [a_i, b_i]_k \in N_i, n_{jk} \in N_j, a_{ik} \leq n_{jk} \leq b_{ik}$*. o is then* ***covered*** *by p.*

Example 2 *Example: The pattern* $\langle (\{a\}, \langle [1,3], [2,5] \rangle), (\{b\}, \langle [2,3], [4,5] \rangle) \rangle$ *is a generalization of the complex event sequence* $\langle (\{a,b\}, \langle 2,2 \rangle), (\{c\}, \langle 7,0 \rangle), (\{b\}, \langle 3,5 \rangle) \rangle$*.*

As the number of patterns grows exponentially w.r.t. the number of complex event sequences, we focus on patterns that discriminate skillshots, that is, whose support sequence are strongly correlated to a skillshot. .

Example 3 *Given the arbitrary pattern p, using only player inputs for simplification,* $p = \langle (\{a,b\}), (\{j,d\}) \rangle$ *and the data* $\mathcal{D}$ *given in Table 6.2, we have* $ext(p, \mathcal{D}) = \{1,3\}$*,* $support(p, \mathcal{D}) = 2$*, and* $WRAcc(p, +, \mathcal{D}) = \frac{2}{4} \times (\frac{2}{2} - \frac{2}{4}) = 0.25$*. Notice that we have more than two labels for our skillshot classification task. We consider a one versus all scheme, i.e., we will focus on a target class, say the positive class while all the others will be merged to build the negative one.*

In order to demonstrate visually the challenge of the task, we plotted two examples of sequences of inputs, without representing contextual information for the sake of clarity, in Fig. 6.5 and Fig. 6.7. The two of them are performances of ceiling shots. Our goal is to find patterns appearing in a given skillshot, and not in others. An example of such patterns is represented in Fig. 6.6 and Fig. 6.8.

6.2.4.2 SeqScout adaptation

The `SeqScout` algorithm [89] presented in Chapter. 4 can mine discriminative patterns in sequences of itemsets. In the following, present a slight adaptation of SeqScout to mine behavioral patterns.

We recall that the idea of `SeqScout` is to iteratively select a sequence of the dataset following a trade-off between exploration and exploitation, using UCB [6], and then to generalize this element ("going up" in the search space) creating a new pattern. The quality of this pattern, i.e., its discriminating power, is then computed with the chosen quality measure, the $WRAcc$ in our case. Once the time budget has been reached, patterns are filtered to make sure they

Figure 6.4: Player inputs legend

Figure 6.5: Sequence of inputs ceiling shot 1

Figure 6.6: Pattern p appearing first ceiling shot

Figure 6.7: Sequence of inputs ceiling shot 2

Figure 6.8: Pattern p appearing in second ceiling shot

are non-redundant following Jaccard index, using a parameter θ (see [89], and the top-k are returned. To adapt this algorithm to our problem, we need to reconsider the generalisation step as complex event sequences also contains vectors of intervals.

6.2.4.3 Sequence generalisation

The sequence generalisation consists of two steps. In the first, each itemset of inputs I is considered, for the selected sequence. Each item in I is removed following the rule:

$$\begin{cases} remain, & \text{if } z < 0.5 \\ remove, & \text{if } z \geq 0.5 \end{cases}, \text{ where } z \sim \mathcal{U}(0,1).$$

If I is empty, the entire corresponding state X is removed. Then in the second step, each numerical variable is considered. Each $n \in N$ is mutated following the rule, given $n_{left} \in Dom(n)$ s.t. $n_{left} \leq n$, $n_{right} \in Dom(n)$ s.t $n_{right} \geq n$ and $\alpha \in [0,1]$, where $Dom(v)$ represents the set of values taken by variable v in the dataset:

$$\begin{cases} [-\infty, \infty], & \text{if } z < \alpha \\ [n_{left}, n_{rigth}], & \text{if } z \geq \alpha \end{cases} where\ z\ \sim \mathcal{U}(0,1)$$

For example, considering the sequence:

$$\langle (1, \{jump\}, \{speed = 158, DistanceBall = 10\}),$$
$$(2, \{right, slide\}, \{speed = 102, DistanceBall = 29\}\rangle$$

One possible generalisation is:

$$\langle (2, \{slide\}, \{speed = [88, 107], DistanceBall = [-\infty, \infty]\}\rangle$$

Removing constraints of random variables leads to the creation of patterns with restrictions only on a subset of variables from the state. It helps to find more interesting patterns faster.

6.2.5 Dataset re-encoding

In the following we propose a solution that fit into the LeGo framework [72]. Once a set of patterns has been extracted, we re-encode the dataset. We create a feature for each pattern, putting a Boolean "1" value if the pattern appears in the sequence, "0" otherwise, as illustrated in Table 6.3. As explained in [44], “The binary feature construction process is certainly the most straightforward but has also shown good predictive performance”. As we know that those features are discriminative, they will give good insights about the class we need to predict. Note that we need to mine discriminative patterns of each class to classify all of them.

Once the dataset in transformed into a binary transaction labeled dataset, classical machine learning algorithms can be used to predict the skillshot the player is performing.

6.3 Related Work

Understanding and analyzing players behavior is an important subject. For unlabeled data, time series clustering approaches were applied to different games, on free-to-play game data to find relevant patterns in [114], or to discover seasonal patterns in [125].

id	Pattern 1	Pattern 2	Pattern 3	class
1	1	1	0	+
2	0	0	1	-
3	1	1	0	+
4	1	0	1	-

Table 6.3: Toy dataset re-encoded with 3 patterns

Concerning labeled data, we take interest in the identification of "skills", or "moves". In games, it is most of the time done with expert knowledge and ad-hoc techniques. For example, in fighting games, multiple precise combinations of controller inputs lead to the execution of particular attacks or combos. To do so, exact pattern matching is used [138]. However in a game like *Rocket League*, such an approach cannot be used. Indeed, each player has a full and precise control of her car, in a 3D space, including all possible rotations, speed, acceleration, even the reaction of bumping other cars. Having such a huge space of possible configurations leads to the fact that each performance of a skill is unique: positions, speeds, rotations of the car and the ball will not be the same, and even inputs pressed by the player can vary a lot (many micro-adjustments of trajectory). This makes the problem of automatically classifying (or detecting) the action of the player difficult. *Rocket League* seems to use in-game a system using ad-hoc rules such as "if the ball touched the back of the car and then scored, it is a backward goal", but this approach is limited to simple rules, based purely on expert knowledge, that would not work on more complex skills.

In our context, we need to take into account inputs of the player, to know what she wanted to perform, but we also need the contextual information of the game. For instance, performing a "front flip" in its own goal without touching the ball and doing it to hit the ball will impact the game differently. We are then in the presence of the so called complex event sequences [131].

In fact, we can transform the event part of the sequence in booleans taking 0 values by default, and 1 when the event fires. It reduces the problem to a multivariate time series classification problem [9]. Most of the recommended methods require time series to have the same length, whether it is BOSS [115], COTE [10], Shapelet Transform [62], or Time Series Forest [36]. In our case here, time series have different lengths. The 1-Nearest Neighbor Dynamic Time Warping (1-NN DTW) [94] is said to be a good baseline [9]. However, the prediction step in KNN can take a long time, even more because the DTW distance has a complexity of $O(n^2)$ for sequences of size n. It would make a good candidate to compare to in our experiments though.

Finally, it should be highlighted that we were not able to find any data analytics article on *Rocket League* in the literature. It strengthens our first contribution: it appears useful to provide a dataset mixing contextual information and player actions, along with the needed tools to create new datasets.

6.4 Experiments

The whole methodology has been fully implemented and a thorough experimental study has been carried out. All materials are publicly available, including our original dataset, scripts, algorithms, and our self-made program for capturing inputs from joysticks. In this section, we discuss a number of experiments that indicate to which extent our methodology can automatically detect skillshot from previously unseen games.

6.4.1 Dataset

Following the presented workflow, we generated data with the help of experienced players of the game. We describe the resulting dataset in Table 6.4. The *max. size* corresponds to the maximum number of states in a sequence. We recall that each state is composed of 7 variables and between 1 and 11 events, creating long sequences nearly impossible to read for a human. Note that during labelling step we added failed skills, i.e., missing goals, or not hitting the ball while performing the skill, and random non-skill sequences in games as "noise". Indeed, to train our classifier to perform in real conditions, we need to be able do detect when player fails. We also added classes distribution in Table 6.5. The min and max values of the variables are given in Figure 6.14.

6.4.2 Experimental setup

We ran the first series of experiments not reported here to determine the best default parameters for building a reasonable classifier: we will study how each parameter affects the results when other parameters are fixed to their default value. As such, when not specified, the default parameters in experiments are the following. We used 5-fold stratified cross validation to assess the robustness of our classifier. We took the top-5 best patterns given by `SeqScout`, on each class of the dataset, as features for the classifier. We gave 1,000 iterations for `SeqScout`, which seemed a reasonable trade-off between pattern quality and time taken by the algorithm. The non-redundancy parameter θ is set to 0.8. The parameter α, controlling the sequence generalisation, is set to 0.9, and the Decision Tree (from [103]) is chosen for the final classification step, for its simplicity and interpretability.

With this experimental setup, we provide answer elements to the following questions.

- How many patterns are required to maximize the accuracy of our classifier?
- How does the α parameter affect patterns quality?
- How does the qualities of computed patterns affect the quality of the classification and how many iterations does `SeqScout` need to perform well?
- How does the non-redundancy of patterns affect the classification?
- What is the best classification method to use?
- What is the performance of our method compared to a state-of-the-art algorithm like 1-NN DTW?
- Using only sequences of variables (and not player inputs), can we still accurately predict performed skills?
- How can we use our classifier in real-time to detect performed skills?

Table 6.4: Dataset

# Sequences	# Inputs	# Variables	Max. size	# Classes
298	11	7	64	7

6.4.3 Influence of the number of mined patterns

We evaluated how the number of mined patterns influences the mean accuracy of our method. Note that the pattern number corresponds to the number of mined patterns for each class with `SeqScout`. For example, this means that for 20 mined patterns, having 8 classes in our dataset, we re-encode our dataset with 160 features. Results are given in Fig. 6.9. Here we can see that the predictive power of our method quickly increases with the number of patterns, and then tends to decrease a bit. Setting this parameter between 10 and 30 seems to be a reasonable choice.

In Fig. 6.10, we tested the influence of the parameter α from the generalisation step on the quality of patterns. Note that the $WRAcc$ takes its values in $[-0.25, 0.25]$ (see [89] for more information). Here we can see that we have an optimum for $\alpha = 0.8$. This means that the method gives better results when we remove restrictions on a numeric with the probability 0.8.

6.4.4 Pattern quality w.r.t. accuracy

Increasing the time budget leads to an increase of the mean pattern quality [89]. We then propose to evaluate the impact of the time we give to `SeqScout`. As we can see in Fig. 6.11, the more iterations we give (and equivalently, time), the better are the predictions. Finding good discrimative patterns leads to the design of good discriminative features.

6.4.5 Impact of diversity on accuracy

In Fig. 6.12, we show the impact of the θ parameter. We recall that θ is the threshold above which patterns are considered similar when filtering resulting pattern. Having a low θ means we want `SeqScout` to give us patterns which have few or no sequences in common, and a high θ that we accept similar patterns as the output. Interestingly, this parameter does not seem to have an impact on the accuracy. It is important if we want to filter patterns to an end-user, to show non-redundant results, but in the case of classification, we can choose to remove this costly step without impacting the prediction quality.

6.4.6 Predictive performance of the method

We evaluated the performance of different state-of-the-art classification methods on the re-encoded dataset. In light of preceding results, we chose here the best found parameters: $\alpha = 0.8$, $\theta = 1$, 20 extracted patterns, and a number of iteration of 10,000. We tested Decision Tree (DT), Random Forest (RF), SVM, Naive Bayes from sk-learn [103] and XGBoost (XGB) from [32], all with default parameters values. As we can see in Fig. 6.13, RF, XGboost and SVM seem to give the best results.

Table 6.5: Class Distribution

Noise	Ceiling	Power	Waving	Air	Flick	Musty
43	30	60	38	45	46	36

Table 6.6: Comparaison of 1-NN DTW vs our method

	Acc Train	Acc Test	Time Train	Time Test
Best Approach	94.1	84.9	14 min	-
1-NN DTW	-	71.5	-	7 min

Table 6.7: Confusion matrix of our classifier

	Noise	Ceiling shot	Power shot	Waving Dash	Air Dribble	Flick	Musty Flick
Noise	12	38	0	0	0	38	12
Ceiling shot	0	100	0	0	0	0	0
Power shot	0	0	100	0	0	0	0
Waving Dash	0	0	0	100	0	0	0
Air Dribble	0	0	0	0	100	0	0
Flick	10	0	0	0	0	90	0
Musty Flick	0	0	12	0	0	0	88

6.4.7 Comparison to 1-NN DTW

If we transform events to booleans taking 1 values when event fires, and 0 otherwise, we can reduce the problem to a multivariate timeseries classification. We then compare ourselves to 1-NN DTW, without forgetting to z-normalize timeseries as specified in [9]. We used a DTW implementation[8]. Results are shown in Table 6.6. Our method improves significantly the accuracy. Moreover, we have the strong advantage of being able to classify in nearly-real time, contrary to 1-NN DTW. This is important to create a system that could classify directly in game what players are doing. Note also that there is a trade-off between the duration of the training and the accuracy of our method, as we can tune the number of iterations we give to `SeqScout`.

6.4.8 Using numerical variables only

We tried the approach on sequences of purely numerical variables, meaning that we removed the player inputs information. In a 5-fold stratified cross-validation, we found a mean accuracy of 73.9%. We can then deduce that the information of players inputs leads to a clear increase of accuracy of the system. It supports the need for the proposed data augmentation step.

6.4.9 Classify goals

An example of confusion matrix given in the stratified 5-fold cross-validation is given in Table 6.7.

[8] https://github.com/pierre-rouanet/dtw

As we can see, the classifier has difficulties to classify the "noise" class, composed of failed goals, and random part of games. Using sliding windows to detect skills in real time in a game would then probably give poor results. However, in real setting, the vast majority of figures are ways of scoring goals. To deal with the previous issue and to increase the accuracy of our system, we can introduce a bit more of expert knowledge: we classify the sequence only if a goal has been scored. This way, our system could be directly used in real settings, extracting the sequence of actions before a goal, and feeding it to our classifier. After filtering sequences of our dataset by keeping only those having a "goal" in it, and using our method, we have a mean accuracy of **87.6%**.

6.4.10 Performances of MonteCloPi

As explained before, the Rocket League dataset can be transformed to a multivariate time series dataset, having varying lenghts. Using MonteCloPi, we reached a classification accuracy of **80.1%**, with 5-fold stratified cross-validation, a training time of 1 hour, and **nearly immediate** classification during test time. Results are not as the good as the SeqScout adaptation. We make the hypothese that this is due to the generalisation step. With MonteCloPi, we systematically go to the top of the search space, whereas we can sample any pattern, in theory, with SeqScout. On this dataset, this generalisation strategy is then better.

6.4.11 Pattern interpretability

One of the interest of using a pattern mining approach is its interpretability.

The two following patterns are the top-1 patterns extracted respectively for the "Musty Flick" and the "Ceiling Shot" by `SeqScout`. Note that here for simplicity of notation we removed variables whose intervals were $[-\infty, \infty]$.

$$\langle \{accelerate\}, \{down\ jump\}, \{goal\} \rangle\ (WRAcc = 0.0863)$$
$$\langle \{accelerate, jump, DistanceCeil = [1.52, 1233.51]\} \rangle\ (WRAcc = 0.0755)$$

Interestingly here, the pattern corresponding to the "Musty Flick" only takes into account the inputs of the user, with no restrictions on variables. The sequence of inputs is indeed a part of the required inputs to perform the "Musty Flick". The pattern is only a "part of" the required inputs because in fact the beginning of the sequence of inputs for the "Musty Flick" is common to the "Front Flick": `SeqScout` returns only the discriminative part, i.e., the part that is present in "Musty Flick" figures, and not in others. That is why this method gives good results: the more discriminative patterns are, the better are the feature for the classification step.

Moreover, the best found pattern discriminative of the "Ceiling Shot" is in fact composed of only one state, which is enough to discriminate the skill here: if the player jumps when she is near the ceiling (1,200 corresponds approximately to the medium position between ceiling and floor), it often leads to a goal, in our dataset.

6.5 Discussion

We introduced a new original dataset collected by our means, and a method to classify skillshots of players in *Rocket League*. This methodology could be applied to other games, as the type of required data is generic enough. We obtained good results with a mean accuracy of 87.6%, a high score for a multi-class classification problem.

The classifier training can take some time, depending on the time budget the user want to allocate, but the classification can be performed in real-time, contrary to 1-NN DTW.

However, there also exists some drawbacks that could be addressed in future works. The biggest one is the noise problem, as it is difficult to discern failed skills from well executed ones. We dealt with this issue by filtering only sequences finishing with a goal. This approach would not work in the case of classifying skills that do not finish with a goal. Moreover, all existing skills are not present in this dataset, as it would require a lot of additional work with several different experts to increase the diversity of performed and labeled skills.

One may notice when looking at the confusion matrix that many examples of noise are considered as other figures by the classifier. This is due to the fact that noise is composed of different "failed" figures. As they are similar to correctly performed figures, they are difficult to discern. To deal with that, a possibility would be to add a new label for each figure corresponding to its failed or succeeded execution, and to reserve the "noise" class to random moves. Our classifier would then give the intention of the player, and another classifier would give the success or not of this intention.

Note also that this project is a proof-of-concept, that used some "hacks" in the data collection process. A production-ready system would require inputs data directly recorded by the game. In the case of new skill appearing in-game, the system would need to be trained again with new data, to keep being up-to-date with the new meta game.

Some skillshots are also not present in the dataset because our expert players cannot perform them consistently. Indeed, the better the player, the more consistent she is at performing skills, as already showed in [23], and so the easier it is to classify (professional players have less variability when repeating strategies). Moreover, at very high level, it is not only one skill at a time that is performed, but more a sequence of skills. Using a system like presented in this paper would be beneficial to several actors. For players, a histogram of skills detected during games could be used for player profiling. For game editor, such a system may help to better rank them, or to detect "smurfing", i.e., playing with another account to improve her ranking [29]. It could also improve analytics for esport structures to better know their future opponents by better understanding their play-style. It would also be interesting to create new game modes where the goal is to perform skills on increasing difficulty (like the game of "horse" in basketball). Finally, this type of analytics could help to study the evolution of the meta-game during seasons, as *Rocket League* is already a game with an history, and is probably going to keep growing over the next years.

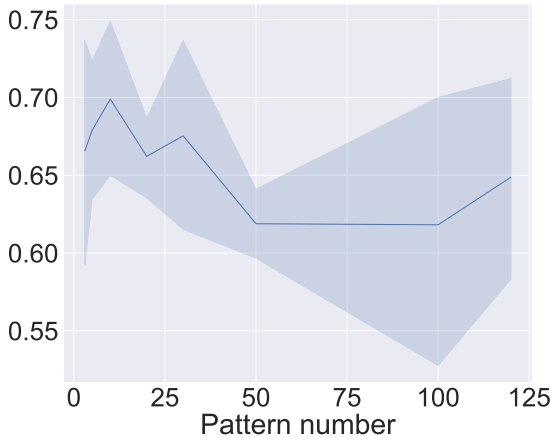


Figure 6.9: Mean accuracy w.r.t. the number of patterns

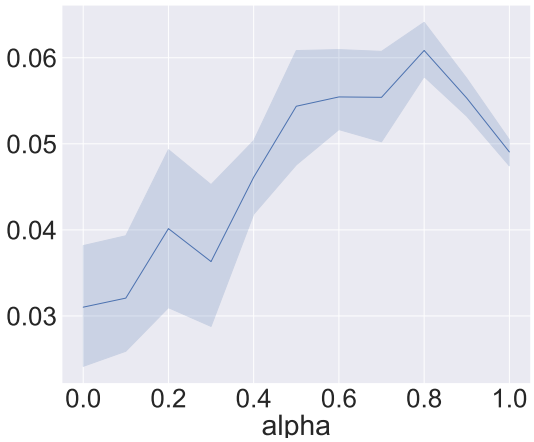


Figure 6.10: Mean WRAcc vs α

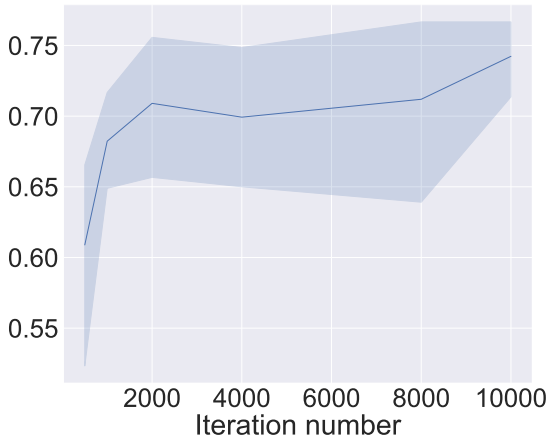


Figure 6.11: Accuracy w.r.t. the number of iterations

Figure 6.12: Accuracy w.r.t. θ

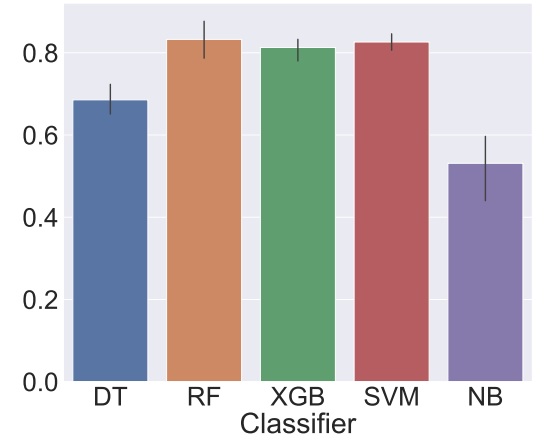


Figure 6.13: Comparison of accuracy of different classifiers

Numeric	Min	Max
BallAcceleration	-319123	294344
Time	0	13.4
DistanceWall	0	4043
DistanceCeil	0.07	2020
DistanceBall	115	9509
PlayerSpeed	27	229999
BallSpeed	0	329814

Figure 6.14: variables min and max values

Chapter 7

Conclusion

7.1 Summary

In this thesis we proposed to tackle the problem of supervised rule discovery, or subgroup discovery, for two types of data, namely sequences of itemsets and high dimension numerical data. Note that the last can also be applied directly to time series.

The problem of finding interesting rules in the presence of labeled sequences of itemsets has attracted few attention. This is probably due to the size of the search space, which quickly grows making an exhaustive search infeasible. Some rare propositions have been made, however they often focus on sequences of items, or are dependant on the used quality measure.

Some of them like `misère` [44] or the popular `Beam Search`[42] are interesting to tackle this problem, but they present several drawbacks. `misère` does not use information of previous drawings, making this algorithm focusing on *exploration* of the search space only. `Beam Search`, on the other hand, can be seen as several hill climbing procedures exploring the search space in a top-down way, making this algorithm focusing on *exploitation* only. We proposed two algorithms, namely `SeqScout` and `MCTSExtent` that are able to find a trade-off between the exploitation, i.e., visiting search space areas near already known good patterns, and exploration, i.e., exploring unknown search space areas to gather new information. Both method use the well known UCB1 [6] formula, proposed in the game theory literature, to deal with this trade-off.

`SeqScout` tackled this issue with a multi-armed bandit model, where the action of playing a bandit arm is seen as choosing a sequence and then generalising it. This method presents the advantage of being simple, and to give interesting results. `MCTSExtent` pushes further this idea by applying a Monte Carlo Tree Search [25] on the search space of extends, by successively grouping positive elements and computing their longest common subsequences, i.e., finding one of the most specific description that cover those elements. This create patterns that cover at least those interesting elements, meaning that they may probably be interesting. To do so, we also proposed a dynamic programming algorithm to find a longest common subsequence between two sequences of itemsets. This algorithm is anytime, so time budget given to it can be controlled, and results tend to improve over time, as search space interesting areas are better estimated iteration after iteration. One of the strength of those methods is that they reduce the size of the search space by considering only patterns having a non-null support, which are numerous when enumerating all possible ones for sequences of itemset.

Inspired by the idea of `MCTSExtent` and its good results, we proposed to tackle the problem of

finding interesting rules in high dimensional data, in particular because time series can be seen as an instance of this problem. We then proposed `MonteCloPi`, which has the property of exploring closed on the positive patterns, reducing the size of the search space with the guarantee of keeping only interesting patterns. In particular, we showed that when considering (multivariate) time series of same lengths, `MonteCloPi` is exhaustive on the closed on the positives.

In order to assess the relevance of our algorithms, we proposed to tackle the problem of classifying player behaviour in a competitive e-sport game, Rocket League. We described and implemented a complete workflow including domain expert knowledge and intervention to do so. We successively identified behaviours we wanted to detect, played matches, gathered data, pre-processed and merged them from different sources, and finally labeled them. Once those steps have been performed, we embraced the LeGo method [72] by adapting `SeqScout` to deal with data specificities, re-encoding the dataset to create a binary matrix, and using a random forest classifier. We showed that we were able to detect figures players performed in real-time, which is of interest to game commentators, e-sport teams analysts, or to integrate new game modes or improve the ranking system, for example.

7.2 Perspectives

7.2.1 Improving `MCTSExtent` and `MonteCloPi`

Improving the roll-out step of `MonteCloPi` to cover only closed on the positives

One way of improvement for the `MonteCloPi` algorithm would be to change the roll-out step in order to force it to create a closed on the positive. Indeed, the actual roll-out step, or generalisation, consists in removing restrictions on all intervals excepting one. Even if this strategy gave good results experimentally, it creates a pattern which can be a non closed on the positives. Moreover, as the number of iterations increases, the probability of sampling an already discovered pattern increases too. It may be more interesting to take advantage of the need to perform a roll-out do discover a new pattern each time. One proposition that may be interesting to investigate would be the following: *computing the meet of all positive elements, except k among positive elements that are not in the extent of the expanded node.* Indeed, when considering closed on the positives, the most general pattern is the meet of all positive elements. Each level of the search space of closed on the positive is in fact composed of the meet of $|D_+|-k$ positive elements, where $|D_+|$ is the number of positive elements of the dataset. The idea would then be to roll-out only unseen patterns, creating a new tree at the top of the search space. As it would cover elements of the expanded node and additional positive ones, it would indeed be a generalisation of the expanded node, and would help covering the search space of closed on the positive quicker.

Making algorithms exhaustive on all cases

One of the issue of `MCTSExtent` is that it does not have the property of exhaustiveness. `MonteCloPi` has also this issue when dealing with time series of different lengths. This is due to the fact the meet (or longest common subsequence for `MCTSExtent`) is not unique. When enumerating the search space with successive expand steps, important and interesting patterns could be missed, as we compute only one meet. In order to make the search exhaustive, we would

need to compute all meets, and to create as many expanded node as possible meets. However, the number of common patterns between two positives elements can be gigantic, as showed by Egho et al. in [45] for sequences of itemsets. Further investigations would be needed to tackle this difficult problem.

Parallelisation
Monte Carlo Tree Search offers different way of parallelisation to improve results quality for a same time budget: the leaf parallelisation, the root parallelisation and the tree parallelisation [25]. The leaf parallelisation seems to be the simplest applied to our setting, as it does not require the use of a mutex. The idea of the leaf parallelisation is to run several roll-outs in parallel after expanding a node, giving a better appreciation of its quality. One of the drawbacks of the leaf parallelisation in classical MCTS is that the time of the roll-out step will be the longest among roll-outs launched in parallel. However in our case our roll-out strategy should take approximately the same time for any possible roll-out. Moreover, considering previous improvement of making the roll-out step create a pattern closed on the positive, this leaf parallelisation step would decrease the time needed to explore the search space exhaustively.

Applying this methodology to other pattern description languages

The methodology proposed with `MCTSExtent` and `MonteCloPi` is in fact generic enough to be applied to other types of data, not only numericals or sequences of itemsets. Indeed, this idea of exploring the search space of positive extends, iteratively grouping positive objects and computing their meet in order to create interesting patterns, coupled to the use of the Monte Carlo Tree Search, could be applied to graphs or categorical data, for example. The only two steps that need to be redefine is the meet step, and the generalisation step.

7.2.2 Going further with the Rocket League use case

Improving the classification in Rocket League

One possible way of improvement for the Rocket League use case would be to better deal with the "noise" problem. Indeed, in our settings, we chose to create a class containing noise, i.e., random moves and *failed figures*. However as those failed figures are very similar to successful ones, our classifier tend to classify those objects as figures the player wanted to perform. A better approach would be to reserve the "noise" class to only random moves, and to add a new "succeeded" label for each object, denoting the success or not of the figure. The classifier we proposed in this article would then be used to detect the intention of the user. After this classifier identifies user intention, another one would detect the success or not of the figure. A possibility would be to re-use an approach similar to the first one focusing on classifying the success of a figure. Another simpler, yet probably effective approach would be to use the simple rule "if a goal is scored quick after player performed the figure, we considered it successful".

Automatic detection of new classes

One problem in a game like Rocket League is that the number of skillshots that can be

performed, i.e., the number of classes, increases over time. Indeed, the community still discovers new mechanics, and new ways of scoring goals: the "musty flick" was unknown to game developers, but it emerged and became quickly famous. Such a system of figure classification would have to deal with this issue: the number of class increases over time. To tackle this problem, one possibility would be to have an expert monitor matches to identify new skillshots. Once this is the case, new label data would need to be collected for this new class. However we could imagine to take advantage of the pattern mining approach presented in this thesis. Each figure is re-encoded to a boolean vector, each value corresponding to the presence or not of a previously mined pattern. Figures of the same class should form clusters in the feature space. As the number of class is known at a time t, it would be possible to use a clustering algorithm like `k-means`[82], for example, to assess this hypothesis. Then, we could automatically detect outliers that are far away from other centroïds, and are grouped together. This would trigger an alarm indicating that one or several new classes appeared, so that the system would need to take this into account in order to keep performing correctly. Note that this approach could be applied not only to this use case: a production system exhibiting and classifying time series, or more generally classifying users behaviour, could take advantage of such an approach to evolve over time.

Bibliography

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, page 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] M. Atzmueller. Subgroup discovery. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, 5(1):35–49, Jan. 2015.

[3] M. Atzmueller and F. Lemmerich. Fast subgroup discovery for continuous target concepts. In *Proceedings of the 18th International Symposium on Foundations of Intelligent Systems*, ISMIS '09, page 35–44, Berlin, Heidelberg, 2009. Springer-Verlag.

[4] M. Atzmueller and F. Lemmerich. Vikamine – open-source subgroup discovery, pattern mining, and analytics. In P. A. Flach, T. De Bie, and N. Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 842–845, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] M. Atzmüller and F. Puppe. Sd-map - A fast algorithm for exhaustive subgroup discovery. In *Proc. PKDD*, pages 6–17, 2006.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, May 2002.

[7] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings ACM SIGKDD 2002*, pages 429–435, 2002.

[8] T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., GBR, 1st edition, 1997.

[9] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.*, 31(3):606–660, 2017.

[10] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, Sep. 2015.

[11] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. The UEA & UCR time series classification repository, 2017. `www.timeseriesclassification.com`.

[12] I. Batal, D. Fradkin, J. H. Harrison, F. Mörchen, and M. Hauskrecht. Mining recent temporal patterns for event detection in multivariate time series data. *Proc. ACM SIGKDD*, pages 280–288, 2012.

[13] S. Bay and M. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5, 01 2001.

[14] A. Belfodil. *An Order Theoretic Point-of-view on Subgroup Discovery.* Theses, Université de Lyon, Sept. 2019.

[15] A. Belfodil, A. Belfodil, A. Bendimerad, P. Lamarre, C. Robardet, M. Kaytoue, and M. Plantevit. Fssd - a fast and efficient algorithm for subgroup set discovery. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 91–99, 2019.

[16] A. Belfodil, A. Belfodil, and M. Kaytoue. Anytime subgroup discovery in numerical domains with guarantees (best student paper). In *Proc. ECML/PKDD*, pages 500–516, 2018.

[17] F. Berlanga, M. J. del Jesus, P. González, F. Herrera, and M. Mesonero. Multiobjective evolutionary induction of subgroup discovery fuzzy rules: A case study in marketing. In P. Perner, editor, *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, pages 337–349, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[18] M. Boley. The power of saying "i don't know"—an introduction to subgroup discovery and local modeling, 2017.

[19] M. Boley, T. Horváth, A. Poigné, and S. Wrobel. Listing closed sets of strongly accessible set systems with applications to data mining. *Theoretical Computer Science*, 411(3):691 – 700, 2010.

[20] M. Boley, C. Lucchese, D. Paurat, and T. Gärtner. Direct local pattern sampling by efficient two-step random procedures. In *Proceedings ACM SIGKDD 2011*, pages 582–590, 2011.

[21] G. Bosc. *Anytime discovery of a diverse set of patterns with Monte Carlo tree search. (Découverte d'un ensemble diversifié de motifs avec la recherche arborescente de Monte Carlo).* PhD thesis, University of Lyon, France, 2017.

[22] G. Bosc, J.-F. Boulicaut, C. Raïssi, and M. Kaytoue. Anytime discovery of a diverse set of patterns with monte carlo tree search. *Data Min. Knowl. Discov.*, 32(3):604–650, May 2018.

[23] G. Bosc, P. Tan, J.-F. Boulicaut, C. Raïssi, and M. Kaytoue. A Pattern Mining Approach to Study Strategy Balance in RTS Games. *IEEE Trans. Comput. Intellig. and AI in Games*, 9(2):123–132, June 2017.

[24] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone. *Classification and regression trees.* Chapman and Hall/CRC, 1984.

[25] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(1):1–43, March 2012.

[26] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[27] M. Campbell, A. Hoane, and F. hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57 – 83, 2002.

[28] C. J. Carmona, P. Gonzalez, M. J. d. Jesus, and F. Herrera. Nmeef-sd: Non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. *IEEE Transactions on Fuzzy Systems*, 18(5):958–970, 2010.

[29] O. Cavadenti, V. Codocedo, J. Boulicaut, and M. Kaytoue. When cyberathletes conceal their game: Clustering confusion matrices to identify avatar aliases. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.

[30] O. Cavadenti, V. Codocedo, J. Boulicaut, and M. Kaytoue. What did i do wrong in my moba game? mining patterns discriminating deviant behaviours. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 662–671, 2016.

[31] C. R. Charig, D. R. Webb, S. R. Payne, and J. E. Wickham. Comparison of treatment of renal calculi by open surgery, percutaneous nephrolithotomy, and extracorporeal shockwave lithotripsy. *BMJ*, 292(6524):879–882, 1986.

[32] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[33] W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Machine Learning Proceedings 1995*, pages 115 – 123. Morgan Kaufmann, San Francisco (CA), 1995.

[34] B. Crémilleux and J.-F. Boulicaut. Simplest rules characterizing classes generated by δ-free sets. In M. Bramer, A. Preece, and F. Coenen, editors, *Research and Development in Intelligent Systems XIX*, pages 33–46, London, 2003. Springer London.

[35] M. J. del Jesus, P. Gonzalez, F. Herrera, and M. Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *IEEE Transactions on Fuzzy Systems*, 15(4):578–592, 2007.

[36] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142 – 153, 2013.

[37] L. Diop, C. T. Diop, A. Giacometti, D. Li, and A. Soulet. Sequential pattern sampling with norm-based utility. *Knowl. Inf. Syst.*, 2020.

[38] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 43–52, New York, NY, USA, 1999. Association for Computing Machinery.

[39] P. Drake and S. Uurtamo. Move ordering vs heavy playouts: Where should heuristics be applied in monte carlo go. In *3rd North America Game-On Conf*, pages 35—42, 2007.

[40] D. Dua and E. Karra Taniskidou. UCI machine learning repository, 2017.

[41] W. Duivesteijn, A. Feelders, and A. Knobbe. Different slopes for different folks: Mining for exceptional regression models with cook's distance. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, page 868–876, New York, NY, USA, 2012. Association for Computing Machinery.

[42] W. Duivesteijn, A. J. Feelders, and A. Knobbe. Exceptional model mining. *Data Min. Knowl. Discov.*, 30(1):47–98, Jan 2016.

[43] V. Dzyuba, M. Leeuwen, and L. De Raedt. Flexible constrained sampling with guarantees for pattern mining. *Data Mining and Knowledge Discovery*, 31, 10 2016.

[44] E. Egho, D. Gay, M. Boullé, N. Voisine, and F. Clérot. A user parameter-free approach for mining robust sequential classification rules. *Knowl. Inf. Syst.*, 52(1):53–81, July 2017.

[45] E. Egho, C. Raïssi, T. Calders, N. Jay, and A. Napoli. On measuring similarity for sequences of itemsets. *Data Min. Knowl. Discov.*, 29(3):732–764, May 2015.

[46] P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng. Vmsp: Efficient vertical mining of maximal sequential patterns. In M. Sokolova and P. van Beek, editors, *Advances in Artificial Intelligence*, pages 83–94, Cham, 2014. Springer International Publishing.

[47] D. Fradkin and F. Mörchen. Mining sequential patterns for classification. *Knowledge and Information Systems*, 45, 01 2015.

[48] J. Fürnkranz. From local to global patterns: Evaluation issues in rule learning algorithms. In K. Morik, J.-F. Boulicaut, and A. Siebes, editors, *Local Pattern Detection*, pages 20–38, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[49] J. Fürnkranz, D. Gamberger, and N. Lavrač. *Foundations of rule learning.* Springer, Berlin, Heidelberg, 01 2012.

[50] J. Fürnkranz, T. Kliegr, and H. Paulheim. On cognitive preferences and the plausibility of rule-based models. *Machine Learning*, 12 2019.

[51] B. Ganter, R. Wille, and C. Franzke. *Formal Concept Analysis: Mathematical Foundations.* Springer-Verlag, Berlin, Heidelberg, 1st edition, 1997.

[52] G. C. Garriga, P. Kralj, and N. Lavrač. Closed sets for labeled data. *J. Mach. Learn. Res.*, 9:559–580, 2008.

[53] R. Gaudel and M. Sebag. Feature selection as a one-player game. In *ICML*, 2010.

[54] A. Giacometti, D. H. Li, P. Marcel, and A. Soulet. 20 years of pattern mining: A bibliometric survey. *SIGKDD Explor. Newsl.*, 15(1):41–50, Mar. 2014.

[55] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* The MIT Press, 2016.

[56] S. Gsponer, B. Smyth, and G. Ifrim. Efficient sequence regression by learning linear models in all-subsequence space. In *Proceedings ECML/PKDD 2017 (2)*, pages 37–52, 2017.

[57] T. Guyet and Q. René. Negpspan: efficient extraction of negative sequential patterns with embedding constraints. *Data Mining and Knowledge Discovery*, pages 563–609, 04 2020.

[58] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[59] Z. He, S. Zhang, F. Gu, and J. Wu. Mining conditional discriminative sequential patterns. *Inf. Sci.*, 478:524–539, 2019.

[60] Z. He, S. Zhang, and J. Wu. Significance-based discriminative sequential pattern mining. *Expert Systems with Applications*, 122:54 – 64, 2019.

[61] R. Herbrich, T. Minka, and T. Graepel. Trueskill(tm): A bayesian skill rating system. In *Advances in Neural Information Processing Systems*, pages 569–576, 2007.

[62] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Min. Knowl. Discov.*, 28(4):851–881, July 2014.

[63] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, Oct. 1977.

[64] G. Ifrim, G. Bakir, and G. Weikum. Fast logistic regression for text categorization with variable-length n-grams. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 354–362, 2008.

[65] Intel. *Intel World Open*, Accessed March 28, 2020. `https://www.intelworldopen.gg/rocket-league/`.

[66] X. Ji, J. Bailey, and G. Dong. Mining minimal distinguishing subsequence patterns with gap constraints. *Knowl. Inf. Syst.*, 11(3):259–286, Apr 2007.

[67] N. Jin, P. Flach, T. Wilcox, R. Sellman, J. Thumim, and A. Knobbe. Subgroup discovery in smart electricity meter data. *IEEE Trans. on Industrial Informatics*, 10(2):1327–1336, 2014.

[68] V. Jovanoski and N. Lavrač. Classification rule learning with apriori-c. In *Proceedings of The10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-Agent Systems, Logic Programming and Constraint Solving*, EPIA '01, page 44–51, Berlin, Heidelberg, 2001. Springer-Verlag.

[69] B. Kavšek, N. Lavrač, and V. Jovanoski. Apriori-sd: Adapting association rule learning to subgroup discovery. In M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, editors, *Advances in Intelligent Data Analysis V*, pages 230–241, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[70] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Revisiting numerical pattern mining with formal concept analysis. In *Proc. IJCAI*, page 1342–1347, 2011.

[71] M. Kaytoue, A. Silva, L. Cerf, W. Meira, and C. Raïssi. Watch me playing, i am a professional: A first study on video game live streaming. In *WWW (companion volume)*, page 1181–1188, 2012.

[72] A. J. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models: The {LeGo} approach to data mining. In *From Local Patterns to Global Models: Proceedings of the ECML/PKDD-08 Workshop (LeGo-08)*, pages 1–16, 2008.

[73] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[74] S. Kuznetsov. A fast algorithm for computing all intersections of objects in a finite semi-lattice. *Automatic Documentation and Mathematical Linguistics*, 27:11–21, 01 1993.

[75] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1675–1684, New York, NY, USA, 2016. Association for Computing Machinery.

[76] N. Lavrač, P. A. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In *Proceedings ILP 1999*, pages 174–185, 1999.

[77] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with cn2-sd. *J. Mach. Learn. Res.*, 5:153–188, Dec. 2004.

[78] C.-S. Lee, M. Müller, and O. Teytaud. Special issue on monte carlo techniques and computer go. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2:225 – 228, 01 2011.

[79] M. V. Leeuwen and A. J. Knobbe. Diverse subgroup set discovery. *Data Min. Knowl. Discov.*, 25(2):208–242, 2012.

[80] F. Lemmerich and M. Becker. pysubgroup: Easy-to-use subgroup discovery in python. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 658–662, 2018.

[81] B. Letham, C. Rudin, and D. Madigan. Sequential event prediction. *Machine Learning*, 93(2):357–380, Nov 2013.

[82] S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982.

[83] D. Louapre. Le paradoxe de simpson, 2013.

[84] T. Lucas, T. C. Silva, R. Vimieiro, and T. B. Ludermir. A new evolutionary algorithm for mining top-k discriminative patterns in high dimensional data. *Applied Soft Computing*, 59:487 – 499, 2017.

[85] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[86] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1:241–258, 1997.

[87] R. Mathonat, J. Boulicaut, and M. Kaytoue. Découverte de sous-groupes à partir de données séquentielles par échantillonnage et optimisation locale. In M. Rousset and L. Boudjeloud-Assala, editors, *Extraction et Gestion des connaissances, EGC 2019, Metz, France, January 21-25, 2019*, volume E-35 of *RNTI*, pages 153–164. Éditions RNTI, 2019.

[88] R. Mathonat, J.-F. Boulicaut, and K. Mehdi. A behavioral pattern mining approach to model players skills in rocket league. In *IEEE Conference on Games*, 2020.

[89] R. Mathonat, D. Nurbakova, J. Boulicaut, and M. Kaytoue. Seqscout: Using a bandit model to discover interesting subgroups in labeled sequences. In *Proc. IEEE DSAA*, pages 81–90, 2019.

[90] A. Millot, R. Cazabet, and J. Boulicaut. Optimal subgroup discovery in purely numerical data. In H. W. Lauw, R. C. Wong, A. Ntoulas, E. Lim, S. Ng, and S. J. Pan, editors, *Advances in Knowledge Discovery and Data Mining - 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11-14, 2020, Proceedings, Part II*, volume 12085 of *Lecture Notes in Computer Science*, pages 112–124. Springer, 2020.

[91] A. Millot, R. Mathonat, R. Cazabet, and J.-F. Boulicaut. Actionable subgroup discovery and urban farm optimization. In M. R. Berthold, A. Feelders, and G. Krempl, editors, *Advances in Intelligent Data Analysis XVIII*, pages 339–351, Cham, 2020. Springer International Publishing.

[92] S. Moens and M. Boley. Instant exceptional model mining using weighted controlled pattern sampling. In *Proceedings IDA 2014*, pages 203–214, 2014.

[93] F. Mörchen and A. Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *Data Min. Knowl. Discov.*, 15(2):181–215, Oct 2007.

[94] A. Mueen and E. Keogh. Extracting optimal performance from dynamic time warping. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 2129–2130, 2016.

[95] D. Nguyen, W. Luo, T. D. Nguyen, S. Venkatesh, and D. Phung. Sqn2vec: Learning sequence representation via sequential patterns with a gap constraint. In M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 569–584, Cham, 2019. Springer International Publishing.

[96] T. Nguyen, S. Gsponer, I. Ilie, M. O'reilly, and G. Ifrim. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data Min. Knowl. Discov.*, 33(4):1183–1222, 2019.

[97] P. K. Novak, N. Lavrač, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal Machine Learning Research*, 10:377–403, June 2009.

[98] S. Nowozin, G. Bakir, and K. Tsuda. Discriminative subsequence mining for action classification. In *Proceedings IEEE ICSV 2007*, pages 1–8, Oct 2007.

[99] C. Nunes, M. De Craene, H. Langet, O. Camara, and A. Jonsson. Learning decision trees through monte carlo tree search: An empirical evaluation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1348, 02 2020.

[100] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Mach. Learn.*, 5(1):71–99, May 1990.

[101] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In C. Beeri and P. Buneman, editors, *Database Theory — ICDT'99*, pages 398–416, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[102] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009.

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

[104] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. on Knowl. and Data Eng.*, 16(11):1424–1440, Nov. 2004.

[105] J. Pei, J. Han, B. Mortazavi-asl, and H. Zhu. Mining access patterns efficiently from web logs. In T. Terano, H. Liu, and A. L. P. Chen, editors, *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 396–407, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[106] J. R. Quinlan and R. M. Cameron-Jones. Foil: A midterm report. In P. B. Brazdil, editor, *Machine Learning: ECML-93*, pages 1–20, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[107] C. Raïssi and J. Pei. Towards bounding sequential patterns. In *Proceedings ACM SIGKDD 2011*, pages 1379–1387, 2011.

[108] E. Ramirez, M. Wimmer, and M. Atzmueller. A computational framework for interpretable anomaly detection and classification of multivariate time series with application to human gait data analysis. In *AIME International Workshops KR4HC/ProHealth and TEAAM Revised Selected Papers*, pages 132–147, 2019.

[109] Reddit. *All Rocket League moves / skills with descriptions*, Accessed March 28, 2020. `https://www.reddit.com/r/RocketLeague/comments/adiu96/all_rocket_league_moves_skills_with_descriptions/`.

[110] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[111] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018.

[112] R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[113] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[114] A. Saas, A. Guitart, and A. Periáñez. Discovering playing patterns: Time series clustering of free-to-play game data. In *Proceedings IEEE Conference on Computational Intelligence and Games*, pages 1–8, Sep. 2016.

[115] P. Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.*, 29(6):1505–1530, 2015.

[116] R. She, F. Chen, K. Wang, M. Ester, J. L. Gardy, and F. S. L. Brinkman. Frequent-subsequence-based prediction of outer membrane proteins. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 436–445, New York, NY, USA, 2003. Association for Computing Machinery.

[117] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[118] E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the royal statistical society series b-methodological*, 13:238–241, 1951.

[119] E. Spyropoulou, T. De Bie, and M. Boley. Interesting pattern mining in multi-relational data. *Data Min. Knowl. Discov.*, 28(3):808–849, May 2014.

[120] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Advances in Database Technology — EDBT '96*, pages 1–17, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[121] N. Tatti and J. Vreeken. The long and the short of it: Summarising event sequences with serial episodes. *Proceedings ACM SIGKDD 2012*, pages 462–470, 08 2012.

[122] T. L. Taylor. *Raising the Stakes:E-Sports and the Professionalization of Computer Gaming.* MIT Press, 2012.

[123] T. L. Taylor. *Watch Me Play: Twitch and the Rise of Game Live Streaming.* Princeton University Press, 2018.

[124] A. Termier, M. C. Rousset, and M. Sebag. Dryade: a new approach for discovering closed frequent trees in heterogeneous tree databases. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 543–546, 2004.

[125] D. Vihanga, M. Barlow, E. Lakshika, and K. Kasmarik. Weekly seasonal player population patterns in online games: A time series clustering approach. In *IEEE Conference on Games*, pages 1–8, 2019.

[126] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[127] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *Proceedings ACM SIGKDD 2003*, pages 216–225. ACM, 2003.

[128] J. Wang, J. Han, and C. Li. Frequent closed sequence mining without candidate maintenance. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1042–1056, 2007.

[129] Wikipedia. *Rocket League*, Accessed March 28, 2020. `https://en.wikipedia.org/wiki/Rocket_League`.

[130] S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *PKDD*, volume 1263 of *LNCS*, pages 78–87. Springer, 1997.

[131] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *SIGKDD Explor. Newsl.*, 12(1):40–48, 2010.

[132] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large datasets. In *In SDM*, pages 166–177, 2003.

[133] M. Zaki, N. Lesh, and M. Ogihara. Planmine: Predicting plan failures using sequence mining. *Artif. Intell. Rev.*, 14:421–446, 12 2000.

[134] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[135] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60, Jan 2001.

[136] M. J. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, pages 457–473, 2002.

[137] C. Zhou, B. Cule, and B. Goethals. Pattern based sequence classification. *IEEE Trans. Knowl. Data Eng.*, 28:1285–1298, 2016.

[138] G. L. Zuin, Y. P. Macedo, L. Chaimowicz, and G. L. Pappa. Discovering combos in fighting games with evolutionary algorithms. In *Proceedings ACM Genetic and Evolutionary Computation Conference GECCO*, page 277–284, 2016.

Abstract

It is extremely useful to exploit labeled datasets not only to learn models and perform predictive analytics but also to improve our understanding of a domain and its available targeted classes. The subgroup discovery task has been considered for more than two decades. It concerns the discovery of rules covering sets of objects having interesting properties, e.g., they characterize a given target class. Though many subgroup discovery algorithms have been proposed for both transactional and numerical data, discovering rules within labeled sequential data has been much less studied.

In that context, exhaustive exploration strategies can not be used for real-life applications and we have to look for heuristic approaches. In this thesis, we propose to apply bandit models and Monte Carlo Tree Search to explore the search space of possible rules using an exploration-exploitation trade-off, on different data types such as sequences of itemset or time series. For a given budget, they find a collection of top-k best rules in the search space w.r.t chosen quality measure. They require a light configuration and are independent from the quality measure used for pattern scoring. To the best of our knowledge, this is the first time that the Monte Carlo Tree Search framework has been exploited in a sequential data mining setting. We have conducted thorough and comprehensive evaluations of our algorithms on several datasets to illustrate their added-value, and we discuss their qualitative and quantitative results.

To assess the added-value of one or our algorithms, we propose a use case of game analytics, more precisely Rocket League match analysis. Discovering interesting rules in sequences of actions performed by players and using them in a supervised classification model shows the efficiency and the relevance of our approach in the difficult and realistic context of high dimensional data. It supports the automatic discovery of skills and it can be used to create new game modes, to improve the ranking system, to help e-sport commentators, or to better analyse opponent teams, for example.

Keywords: Knowledge Discovery in Databases, Pattern Mining, Game analytics.

Résumé

Exploiter des jeux de données labelisés est très utile, non seulement pour entrainer des modèles et mettre en place des procédures d'analyses prédictives, mais aussi pour améliorer la compréhension d'un domaine. La découverte de sous-groupes a été l'objet de recherches depuis deux décennies. Elle consiste en la découverte de règles couvrants des ensembles d'objets ayant des propriétés intéressantes, qui caractérisent une classe cible donnée. Bien que de nombreux algorithmes de découverte de sous-groupes aient été proposés à la fois dans le cas des données transactionnelles et numériques, la découverte de règles dans des données séquentielles labelisées a été bien moins étudiée.

Dans ce contexte, les stratégies d'exploration exhaustives ne sont pas applicables à des cas d'application rééls, nous devons donc nous concentrer sur des approches heuristiques. Dans cette thèse, nous proposons d'appliquer des modèles de bandit manchot ainsi que la recherche arborescente de Monte Carlo à l'exploration de l'espace de recherche des règles possibles, en utilisant un compromis exploration-exploitation, sur différents types de données tels que les sequences d'ensembles d'éléments, ou les séries temporelles. Pour un budget temps donné, ces approches trouvent un ensemble des top-k règles decouvertes, vis-à-vis de la mesure de qualité choisie. De plus, elles ne nécessitent qu'une configuration légère, et sont indépendantes de la mesure de qualité utilisée. A notre connaissance, il s'agit de la première application de la recherche arborescente de Monte Carlo au cas de la fouille de données séquentielles labelisées. Nous avons conduit des études appronfondies sur différents jeux de données pour illustrer leurs plus-values, et discuté leur résultats quantitatifs et qualitatifs.

Afin de valider le bon fonctionnement d'un de nos algorithmes, nous proposons un cas d'utilisation d'analyse de jeux vidéos, plus précisémment de matchs de Rocket League. La decouverte de règles intéressantes dans les séquences d'actions effectuées par les joueurs et leur exploitation dans un modèle de classification supervisée montre l'efficacité et la pertinence de notre approche dans le contexte difficile et réaliste des données séquentielles de hautes dimensions. Elle permet la découverte automatique de techniques de jeu, et peut être utilisée afin de créer de nouveaux modes de jeu, d'améliorer le système de classement, d'assister les commentateurs de "e-sport", ou de mieux analyser l'équipe adverse en amont, par exemple.

Mots-clés: Découverte de connaissances, fouille de motifs, analyse de jeu

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : MATHONAT
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 29/09/20

Prénoms : Romain

TITRE : Rule Discovery in Labeled Sequential Data : Application to Game Analytics

NATURE : Doctorat

Numéro d'ordre : 2020LYSEI080

Ecole doctorale : InfoMaths (ED 512)

Spécialité : Informatique

RESUME :
Exploiter des jeux de données étiquetées est très utile, non seulement pour entrainer des modèles et mettre en place des procédures d'analysesS prédictives, mais aussi pour améliorer la compréhension d'un domaine. La découverte de sous-groupes a été l'objet de recherches depuis deux décennies. Elle consiste en la découverte de règles couvrants des ensembles d'objets ayant des propriétés intéressantes, qui caractérisent une classe cible donnée. Bien que de nombreux algorithmes de découverte de sous-groupes aient été proposés à la fois dans le cas des données transactionnelles et numériques, la découverte de règles dans des données séquentielles étiquetées a été bien moins étudiée.
Dans ce contexte, les stratégies d'exploration exhaustives ne sont pas applicables à des cas d'application réels, nous devons donc nous concentrer sur des approches heuristiques. Dans cette thèse, nous proposons d'appliquer des modèles de bandit manchot ainsi que la recherche arborescente de Monte Carlo à l'exploration de l'espace de recherche des règles possibles, en utilisant un compromis exploration-exploitation, sur différents types de données tels que les séquences d'ensemble d'éléments, ou les séries temporelles. Pour un budget temps donné, ces approches trouvent un ensemble des top-k règles découvertes, vis-à-vis de la mesure de qualité choisie. De plus, elles ne nécessitent qu'une configuration légère, et sont indépendantes de la mesure de qualité utilisée. A notre connaissance, il s'agit de la première application de la recherche arborescente de Monte Carlo au cas de la fouille de données séquentielles étiquetées. Nous avons conduit des études approfondies sur différents jeux de données pour illustrer leurs plus-values, et discuté leur résultats quantitatif et qualitatif.
Afin de valider le bon fonctionnement d'un de nos algorithmes, nous proposons un cas d'utilisation d'analyse de jeux vidéos, plus précisément de matchs de Rocket League. La découverte de règles intéressantes dans les séquences d'actions effectués par les joueurs et leur exploitation dans un modèle de classification supervisé montre l'efficacité et la pertinence de notre approche dans le contexte difficile et réaliste des données séquentielles de hautes dimensions. Elle permet la découverte automatique de techniques de jeu, et peut être utilisée afin de créer de nouveaux modes de jeu, d'améliorer le système de classement, d'assister les commentateurs de "e-sport", ou de mieux analyser l'équipe adverse en amont, par exemple.
MOTS-CLÉS : Découverte de connaissances, fouille de motifs, analyse de jeu

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique enImage et Systèmes d'information(LIRIS)

Directeur de thèse:
Jean-François Boulicaut (Professeur des Universités, INSA de Lyon)
Mehdi Kaytoue (Maître de conférences HDR, INSA de Lyon)

Président de jury :

Composition du jury :
Atzmüller, Martin Professeur (Osnabrueck University)
Termier, Alexandre (Professeur, Université Rennes)
Amer-Yahia, Sihem (Directrice de recherche, CNRS)
Forestier, Germain (Professeur, Université de Haute-Alsace)
Laurent, Anne (Professeure, Université de Montpellier)
Raïssi, Chedy (Chargé de recherche, INRIA et UBISOFT)