Optimal Interpretability-Performance Trade-off of Classification Trees with Black-Box Reinforcement Learning

Hector Kohler, Riad Akrour, Philippe Preux

RESEARCH REPORT
N° 9503
April 2023
Project-Team Scool

ISSN 0249-6399 ISRN INRIA/RR--9503--FR+ENG

Optimal Interpretability-Performance Trade-off of Classification Trees with Black-Box Reinforcement Learning

Hector Kohler [*], Riad Akrour[†], Philippe Preux[*]

Project-Team Scool

Research Report n° 9503 — April 2023 — 21 pages

Abstract: Interpretability of AI models allows for user safety checks to build trust in these models. In particular, decision trees (DTs) provide a global view on the learned model and clearly outlines the role of the features that are critical to classify a given data. However, interpretability is hindered if the DT is too large. To learn compact trees, a Reinforcement Learning (RL) framework has been recently proposed to explore the space of DTs. A given supervised classification task is modeled as a Markov decision problem (MDP) and then augmented with additional actions that gather information about the features, equivalent to building a DT. By appropriately penalizing these actions, the RL agent learns to optimally trade-off size and performance of a DT. However, to do so, this RL agent has to solve a partially observable MDP. The main contribution of this paper is to prove that it is sufficient to solve a fully observable problem to learn a DT optimizing the interpretability-performance trade-off. As such any planning or RL algorithm can be used. We demonstrate the effectiveness of this approach on a set of classical supervised classification datasets and compare our approach with other interpretability-performance optimizing methods.

Key-words: Reinforcement Learning, Supervised Learning, Interpretability

[*] Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 – CRIStAL, Lille, France
[†] Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 – CRIStAL, Lille, France

RESEARCH CENTRE
LILLE – NORD EUROPE

Parc scientifique de la Haute-Borne
40 avenue Halley - Bât A - Park Plaza
59650 Villeneuve d'Ascq

Compromis Interprétabilité-Performance Optimale des Arbres de Classification avec de l'Apprentissage par Renforcement Boîte Noire

Résumé : L'interprétabilité des modèles d'IA permet à l'utilisateur d'effectuer des contrôles de sécurité afin d'instaurer la confiance dans ces modèles. En particulier, les arbres de décision fournissent une vue d'ensemble du modèle appris et soulignent clairement le rôle des caractéristiques essentielles à la classification de ces données. Cependant, l'interprétabilité est entravée si l'arbre de décision est trop grand. Pour apprendre des arbres compacts, un cadre d'apprentissage par renforcement a été récemment proposé pour explorer l'espace des arbres de décision. Une tâche de classification supervisée donnée est modélisée comme un problème de décision de Markov auquel on ajoute des actions qui récoltent des informations sur les caractéristiques d'une donnée, ce qui équivaut à la construction d'un arbre de décision. En pénalisant ces actions de manière appropriée, l'agent RL apprend à faire un compromis optimal entre la taille et les performances de l'arbre appris. Cependant, un agent doit résoudre un problème de décision de Markov partiellement observable. La principale contribution de cet article est de prouver qu'il suffit de résoudre un problème entièrement observable pour apprendre un arbre de décision optimisant le compromis interprétabilité-performance. Ainsi, n'importe quel algorithme de planification ou d'apprentissage par renforcement peut être utilisé. Nous démontrons l'efficacité de cette approche sur un ensemble de tâches de classification supervisée et nous la comparons à d'autres méthodes d'optimisation de l'interprétabilité et de la performance.

Mots-clés : Apprentissage par Renforcement, Apprentissage Supervisé, Interprétabilité

1 Introduction

The last decade or so has seen a surge in the performance of machine learning models, whether in supervised learning [15] or RL [17]. These achievements rely on deep neural models that are often described as black-box [18, 13, 2], trading interpretability for performance. In many real world tasks, predictive models can hide undesirable biases (see e.g. Sec. 2 in [13] for a list of such occurrences) hindering trustworthiness towards AIs. Gaining trust is one of the primary goals of interpretability (see Sec. 2.4 of [2] for a literature review) along with informativeness requests, i.e. the ability for a model to provide information on why a given decision was taken. The computational complexity of such informativeness requests can be measured objectively, and [5] showed that multi-layer neural networks cannot answer these requests in polynomial time, whereas several of those are in polynomial time for linear models and DT.

In contrast to deep neural models, DTs provide a global look at the learned model and transparently reveal which features of the input are used in taking a particular decision. This is referred to as global [13] or model-based [18] interpretability, as opposed to post-hoc interpretability [18, 2]. Even though DTs are globally intepretable, they have also been used in prior work for post-hoc interpretability of deep neural models, e.g. in image classification [33] or RL [6]. The latter work provides another motivation for DTs, as their simpler nature allowed to make a stability analysis of the resulting controllers and provided theoretical guarantees of their efficacy. In the 54 papers reviewed in [13], over 25% use DTs as the interpretable model and over 50% the more general class of decision rules.

DTs are a common interpretable model and it is thus important to improve their associated learning algorithms. However, interpretability of DTs is hindered if the tree grows too large. The quantification of what is too large might vary greatly depending on the desired type of simulability, that is whether we want individual paths from root to leaf to be short or the total size of the tree to be small [16, p. 13]. In both cases, an algorithmic mechanism to control these tree metrics and to manage the inevitable trade-off between interpretability and performance is necessary. One of the main challenges for learning DTs is that it is a discrete optimization problem that cannot, a priori, be solved via gradient descent. Algorithms such as CART [10] build a DT by greedily maximizing the information gain—a performance related criteria. Interpretability can then be controlled by fixing a maximal tree depth, or by using post-processing pruning algorithms [9, 22]. Unfortunately, this two-step process provides no guaranty that the resulting DT is achieving an optimal interpretability-performance trade-off.

An alternative way to learn DTs, that inherently takes into account the interpretability-performance trade-off, is the recently proposed framework of Iterative Bounding Markov Decision Processes (IBMDPs) [31]. An IBMDP extends a base MDP state space with *feature bounds* that encode the current knowledge about the input, and the action space with *information gathering actions* that refine the feature bounds by performing the same test a DT would do: comparing a feature value to a threshold. The reward function is also augmented with a penalty to take the cost of information gathering action into account. The IBMDP reward function encodes an interpretability-performance trade-off: an agent learns when to add decision nodes or when to make a prediction.

In this work we study the IBMDP setting when the base MDPs encode supervised classification tasks. By doing so, we are able to analyse the optimality of policies learned with RL with respect to the IBDMP reward function. Thus our work studies RL frameworks to learn DTs that trade-off between interpretability (depth of the DT) and performance (accuracy of the DT). After a literature review and the introduction of our notations in Section 2 and 3, our work continues as follows:

Section 4: we present a simple toy task to benchmark RL algorithms solving IBMDPs and

analyse causes of failure of existing RL algorithms solving IBMDPs.
Section 5: we present a new RL framework with optimality guarantees w.r.t the IBDMP objective.
Section 6: we apply this framework to UCI [11] supervised classification datasets.

All proofs are provided in Appendix. All learned DTs and the code to reproduce all experiments can be found on an anonymous github[1].

2 Decision Trees for Supervised Learning

2.1 Greedy Approaches

Early research on the induction of DTs focused on the ID3 algorithm, which uses a greedy approach to select the best attribute at each node based on information gain [24]. This approach was later extended by the C4.5 algorithm, which introduced techniques such as pruning and handling missing data [25].

However, these methods may lead to large trees that a human cannot interpret. An other very well-known DT induction algorithm is CART [10] which performs equivalently to C4.5 and has the same troubles with the induction of large trees.

2.2 Optimal Decision Trees

The algorithms discussed earlier are greedy heuristics and may produce poorly performing trees [14]. As a result, there is an increasing interest in developing algorithms that can train DTs to achieve optimal accuracy.

Training optimal DTs for classification can done with dynamic programming [19] or with Mixed-Integer Linear Programming based formulations [7, 32]. However there is no guarantee that the optimal DT will not grow very large.

To provide regularization and encourage interpretability, the size of the tree is typically limited, and then a solver is used to find the DT that maximizes accuracy within the predetermined size constraints. This cannot be considered as optimizing an interpretability-performance trade-off as the size of the resulting DT is given by the user and not learned.

2.3 Online Learning

Another approach to learn DTs for classification is to model the classification task as an MDP. When doing so, states correspond to training samples, and actions build the DT (either add a decision node or a leaf node by making a prediction). The reward function of the MDP encodes a trade-off between the depth of the tree and the performance of the tree. Indeed, there is a penalty for querying information about a training sample feature, and rewards for predictions.

The proposed algorithms of [8] and [31] are RL agents solving Partially Observable MDPs (POMDPs) [28, chapter 3]. In [12], an other RL agent is proposed, this time acting in a fully observable MDP and is a special case of Iterative Bounding MDPs [31] where training samples have categorical features.

None of these works study the optimality of their proposed method with respect to the interpretability-performance trade-off.

[1] https://github.com/KohlerHECTOR/Interpretability-Performance-official-implem

3 Preliminaries

3.1 Supervised Classification Tasks

In this work, we aim to learn DTs for supervised classification tasks. We consider classification tasks made of a set of training examples $\mathcal{X} = \{x_1, ..., x_N\}$ (a dataset), and a set of labels $\mathcal{Y} \in \{C_1, ..., C_K\}^N$ (one of K classes for each training example). Each of the N training example $x_i \in \mathcal{X}$ has d features $x_{i1}, ..., x_{id}$. The goal of the task is to find a classifier $g : \mathbb{R}^d \to \mathcal{Y}$, $g : x_i \mapsto \hat{y}_i$. Classifiers are computed by algorithms optimizing a loss function of $\hat{y}_i = g(x_i)$ and the true label y_i. In this paper we present algorithms returning classifiers that are DTs optimizing a function of both the performance and the interpretability defined in the following sections.

3.2 Markov Decision Problems

We consider an infinite horizon MDP [23] defined by the tuple $\langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the discrete action space, $R : \mathcal{S} \times \mathcal{A} \mapsto [R_{\min}, R_{\max}] \subset \mathbb{R}$ is the reward function, T is the transition function, and $\gamma < 1$ is the discount factor. The agent interacts with the environment according to its policy π. At time t, the agent takes action $a_t \sim \pi(.|s_t)$, $a_t \in \mathcal{A}$, after which it observes the reward r_t and the next state s_{t+1} with probability $T(s_t, a_t, s_{t+1})$. Let $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a]$ be the Q-function, $V^\pi(s) = \mathbb{E}_\pi[Q(s, a)]$ be the value function, $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ be the advantage function, and $J(\pi) = \mathbb{E}[V(s_0)]$ be the policy return for some initial state distribution. In this work we study RL algorithms that find a policy π^* that maximizes J.

3.3 Classification Markov Decision Problems

Any supervised classification task can be cast into a classification MDP $\langle \mathcal{X}, \{C_1, ..., C_K\}, R, T, \gamma \rangle$. If the dataset to be classified is $\mathcal{X} \subsetneq \mathbb{R}^{N \times d}$ then the state space of the MDP is $\mathcal{X}$, that is the set of training examples. If the set of labels is $\mathcal{Y} \in \{C_1, ..., C_K\}^N$, then the action space is $\{C_1, ..., C_K\}$. The transition function is stochastic, we simply transit to a new state (draw a new data point to classify) whatever the action is: $T(x_i, C_h, x_j) = \frac{1}{N}$. The reward function depends on the current state and action: $R(x_i, C_h) = 1$ if $y_i = C_h$ in the supervised classification task; $R(x_i, C_h) = -1$ otherwise. A policy $\pi : x_i \mapsto C_h$ is a classifier, and a policy π that maximizes the expected discounted cumulative reward, also maximizes the classification accuracy.

3.4 Iterative Bounding Markov Decision Problems

3.4.1 Definition

Following [31], we introduce the notion of an Iterative Bounding MDP (IBMDP). IBMDPs are MDPs. Let us consider a Classification MDP $\langle \mathcal{X}, \{C_1, ..., C_K\}, R, T, \gamma \rangle$. We assume $\mathcal{X} = [0, 1]^{N \times d}$. An Iterative Bounding MDP $\langle \mathcal{S}', \mathcal{A}', R', T', \zeta, p, \gamma \rangle$ is defined on top of it with the following properties.

State space $\mathcal{S}' = \mathcal{X} \times \Omega$, with $\Omega \subsetneq [0, 1]^{2d}$. A state $s \in \mathcal{S}'$ has two parts. A training sample $x_i = (x_{i1}, ..., x_{id}) \in \mathcal{X}$, and feature bounds $o = (L_1, ..., L_d, U_1, ..., U_d) \in \Omega$. (L_k, U_k). For each feature of the training sample x_{ik}, (L_k, U_k) represents the current known range of its value.

Initially, $(L_k, U_k) = (0, 1)$ for all k, which are iteratively refined by taking Information-Gathering Actions (IGAs) defined below.

Action space $\mathcal{A}' = \{C_1, ..., C_K\} \cup \mathcal{A}_I$. An agent in an IBMDP can either take a base action $a \in \{C_1, ..., C_K\}$, or an IGA in $\mathcal{A}_I = \{1, \ldots, d\} \times \{\frac{1}{p+1}, ..., \frac{p}{p+1}\}$, with parameter $p \in \mathbb{N}$.

Transition function. If $a \in \{C_1, ..., C_K\}$, a new training sample is drawn at random from the state space $\mathcal{X}$, while feature bounds are reset to $(0, 1)$. If $a \in \mathcal{A}_I$, the base state is left unchanged, but the feature bounds are refined. Given a training sample x_i with feature bounds $o = (L_1, ..., L_d, U_1, ..., U_d)$ The information gathering action $a = (k, v)$ will compare x_{ik} to $v' = v \times (U_k - L_k) + L_k$, and will set the lower bound U_k to v' if $x_{ik} > v'$, otherwise L_k is set to v'.

Reward function. The reward for a base action in $\{C_1, ..., C_K\}$ is defined by the base classification MDP reward function R. For an IGA in $\mathcal{A}_I$ the reward is a fixed value $\zeta \in (-\inf, R_{\max}) \equiv \zeta < 1$ (the maximum value of the base reward function). We impose $\zeta < 1$, as otherwise a policy never taking any base action would always be optimal, though this restriction is not enough to prevent this degenerate case for RL algorithms.

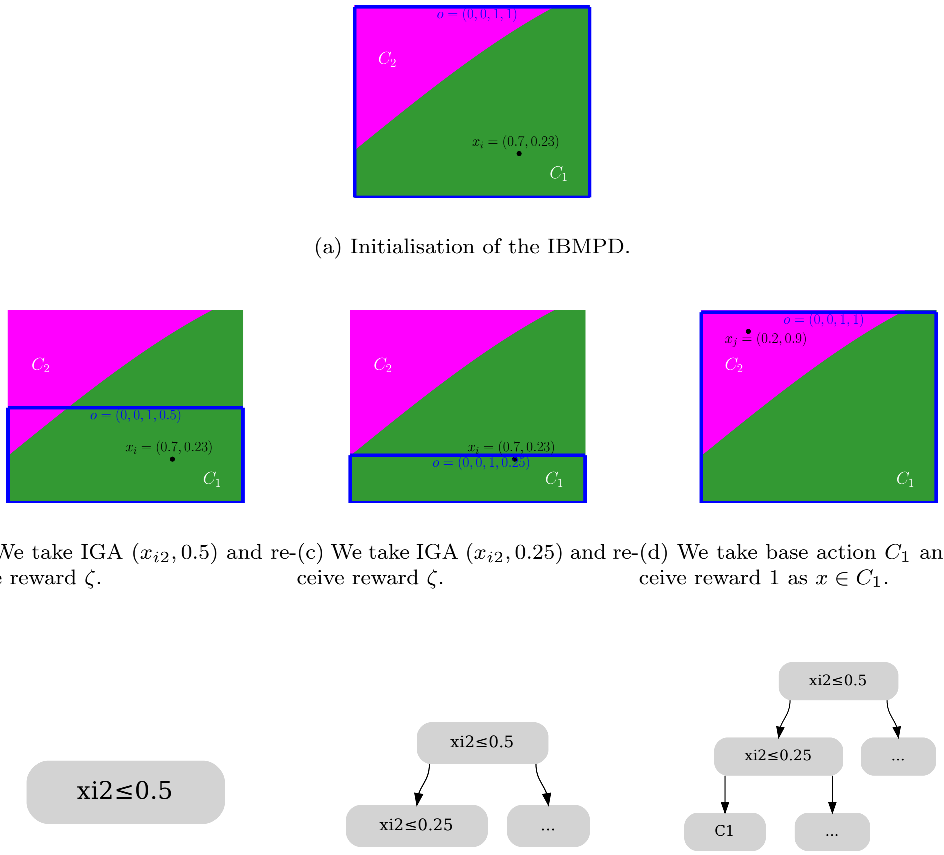
Objective function. Solving an IBDMP is finding a polciy $\pi^* \in \Pi : \mathcal{S}' \to \mathcal{A}'$ such that $\pi^* = \underset{\pi \in \Pi}{\text{argmax}} J(\pi)$ where $J()$ is the expectation of cumulative discounted rewards given by R' (the MDP objective function of Section 3.2).

3.4.2 Learning a DT using Partially Observable RL

As stated in [31], a RL algorithm for an IBMDP should return a policy depending on feature bounds only in order to be able to extract a DT. So an agent learns a DT optimizing an interpretability-performance trade-off encoded by an IBMDP reward function by finding a policy $\pi^* \in \Pi_{DT} : \Omega \to \mathcal{A}'$ such that $\pi^* = \underset{\pi \in \Pi_{DT}}{\text{argmax}} J(\pi)$. We illustrate how an agent learning such a policy is equivalent to learning a DT in Figure 1. To learn a policy depending on feature bounds only, [31] proposes CUSTARD, a partially observable RL algorithm learning a policy depending only on the feature bounds of the IBMDP state and value functions depending on the full IBMDP state. We connect CUSTARD to the class of asymmetric RL algorithms first studied empirically in [21] and more recently theoretically in [3, 4].

Asymmetric Q-Learning. In asymmetric Q-Learning methods, like the DQN [17] version of CUSTARD [31], an oracle state-action function depending on the full state of the IBMDP is learned with TD-learning [30]. This oracle Q-function is used as target for the TD-learning of an other state-action value function, this time, that depends only on feature bounds.

Asymmetric actor-critic. In asymmetric actor-critic methods, like the PPO [27] version of CUSTARD [31], a value function depending on the full state of the IBMDP is learned and a policy depending only on feature bounds is learned. Note that the policy gradient theorem [29] still holds in the asymmetric setting. We show in the next section that CUSTARD fails to learn DTs for simple tasks.



(a) Initialisation of the IBMPD.

(b) We take IGA $(x_{i2}, 0.5)$ and receive reward ζ.

(c) We take IGA $(x_{i2}, 0.25)$ and receive reward ζ.

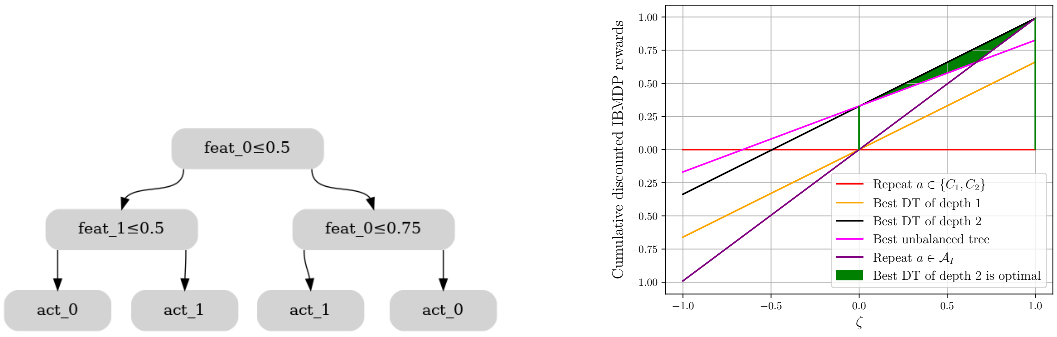
(d) We take base action C_1 and receive reward 1 as $x \in C_1$.

(e) Taking an IGA in the IBMDP adds a decision node to a DT.

(f) Taking an IGA in the IBMDP adds a decision node to a DT.

(g) Taking a base action add a decision node to the DT.

Figure 1: Example trajectory of an IBMDP$\langle p = 1 \rangle$. The state space is divided in two; green states are training samples with label C_1, magenta states are training samples with label C_2. (1a): the IBMDP is initialised: the base state x_i is drawn at random from the base MDP and the feature bounds o are set to $(0, 0, 1, 1)$. (1b): the agent takes the IGA $(x_{i2}, 0.5)$; the observation part is updated to $o = (0, 0, 1, 0.5)$ because $x_{i2} = 0.23 \leq 0.5$. Another IGA is taken in (1c). (1d): the agent takes a base action, so a new base state x_j is drawn from the base transition function and the feature bounds are reset: $o = (0, 0, 1, 1)$.



(a) (b)

Figure 2: A depth 2 binary DT that is optimal w.r.t to the IBMDP objective when $\zeta = 0.5$ (2a) and graphs of the IBMDP reward of different DTs as a function of ζ (2b).

4 Partially Observable RL for simple Classification Tasks

4.1 A Binary Classification Benchmark

In this subsection we address the question: can CUSTARD [31] find the optimal policy in Π_{DT} with respect to the IBMDP objective of Section 3.4.2. To that end, we design toy experiments that are amenable to an analysis thanks to their very small size.

The base tasks are binary supervised classification tasks with 16 different data points and two numerical features in $[0, 1]$. Each data can be perfectly classified using a depth-2 balanced binary tree (see for example Figure 2a). We generate 5 such classification tasks (hence, we will benchmark CUSTARD to retrieve 5 different DTs). Choosing $\zeta = 0.5$, $\gamma = 0.99$ and $p = 1$, induces 5 IBMDPs for which balanced binary DTs of depth 2 are optimal (see Figure 2b).

4.2 CUSTARD to retrieve IBMDP-optimal DTs

To benchmark CUSTARD, we use `stable-baselines3` [26] implementations of PPO and DQN and modify them following the definitions of asymmetric Q-learning and asymmetric actor-critic (see Section 3.4.2). The actor network in PPO is modified to only take feature bounds as inputs while the critic network uses the full state. An additional Q-function depending on the full IBMDP state is learned and used as the target network. We use 5 independent runs for each of the 5 different IBDMPs and normalize returns on each IBMDP so that results can be aggregated. Fig. 3a shows that none of the agents were able to consistently retrieve the best DT despite the extreme simplicity of the task.

4.3 Deriving an exact version of CUSTARD

To better understand how theoretically sound asymmetric actor-critic algorithms [3] like CUSTARD PPO fail to retrieve optimal DTs for simple supervised classification tasks, we start from an exact version of CUSTARD where Q^π and the policy gradient are computed exactly, which is possible in the tabular setup presented next. Note that there do not exist theoretical guarantees for CUSTARD DQN [4, Section 4.4.2] equivalent to the one for CUSTARD PPO which is why we focus on the latter.

We introduce a variant of an IBMDP that enforces a maximum depth of the resulting DTs—and ensures that the DT extraction algorithm always terminates. Let this maximum depth be $M + 1$. M is the maximum number of consecutive time-steps during which a policy can select an IGA. We implement this by forcing the policy to take a base action each time it has performed M consecutive IGAs. Interestingly, if $p + 1$ is prime (where p is the parameter controlling splitting thresholds in IBMDPs), the state space already provides such information to the policy:

Proposition 1 *For an IBMDP, if $p+1$ is prime then there is a mapping $\Omega \mapsto \mathbb{N}$ that maps any feature bound to the number of consecutive IGAs taken since the last base action.*

In other words, the number of consecutive IGAs since the last base action is directly encoded in the feature bounds (please see Appendix for proofs of this and all future statements). Thus we can benchmark, on the same IBMDP, algorithms that enforce a maximum tree depth and algorithms such as CUSTARD [31] that do not.

Having fixed a maximum tree depth $M + 1$, the number of unique feature bounds, i.e. the cardinality of the observation space $|\Omega|$, becomes finite and is at most $(2pd)^M$. Here $pd = |\mathcal{A}_I|$ is the number of IGAs available at any time (if available at all) and the factor of 2 stems from the two possible state transitions following an IGA. Since the state-action space of an IBMDP becomes finite, and its transition and reward functions are known, one can compute the policy gradient exactly. This will let us investigate whether the sub-optimal performance of CUSTARD is due to approximation errors—e.g. introduced by the learned value function—or if it is a limitation of the gradient descent approach in itself.

Because Ω is finite, we can additionally implement policy gradient on tabular policies which would eliminate any representation error of the policy. With a slight abuse of notation, we let in this case $\theta(o, a)$ be the logit of observation-action pair (o, a), i.e. $\pi(a|o) \propto \exp(\theta(o, a))$. By a straightforward application of the chain rule on Lemma C.1 of [1] we obtain:

Proposition 2 *Let $\theta \in \mathbb{R}^{\Omega \times \mathcal{A}'}$ be the logits of a tabular reactive policy of the IBMDP, then:*

$$\frac{\partial J(\pi_\theta)}{\partial \theta(o,a)} = \sum_{s \in \mathcal{S}'} 1_{O(s)=o} \frac{p^{\pi_\theta}(s)}{1-\gamma} \pi_\theta(a|o) A^{\pi_\theta}(s,a). \tag{1}$$

Here $1_{O(s)=o} = 1$ if the feature bound part of s is o, 0 otherwise.

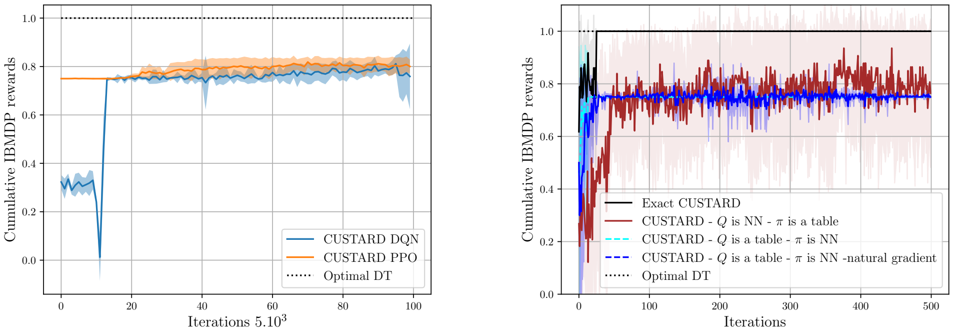
4.4 Ablation study

Starting from the exact CUSTARD algorithm defined above, we perform an ablation study to get to a CUSTARD algorithm similar to [31]. Algorithms are tested on the same IBMDPs as in Section 4.1. The main features ablated from the exact CUSTARD are:

Using an approximated $\hat{Q}^\pi$-function instead of a Q^π-table updated exactly. In that case, $\hat{Q}^\pi$ is a neural network similar to the one in CUSTARD PPO.

Using neural network for the policy π instead of a table. In that case, the policy network is similar to the one in CUSTARD PPO.

The results of the ablation are presented on Figure 3b. The exact CUSTARD algorithm consistently finds the optimal policy. This is important as it means that the partially observable framework of CUSTARD is not the reason for the poor results, at least when implemented exactly. In practice however, we find that both the approximation errors of the neural Q-function and the policy representation error hinder performance. Indeed, when the policy is encoded by a neural network in place of a table, the aggregated cumulative IBMDP reward converges to sub-optimal values after just a few iterations. When the Q^π function is encoded by a neural network, some instances of the associated CUSTARD algorithm converged to the optimal policy—reflected by the higher standard deviations on Fig. 3b, but many did not.



(a) (b)

Figure 3: Study of CUSTARD algorithms ability to retrieve DTs for simple supervised classification tasks by solving IBMDPs. In Figure 3a, we plot the IBMDP cumulative reward during training of CUSTARD as well as the IBMDP cumulative reward of the IBMDP-optimal DT. On Figure 3b, we plot the IBMDP cumulative reward during training of an exact version of CUSTARD, the IBMDP cumulative reward of the IBMDP-optimal DT, as well as the cumulative reward of different approximated versions of the exact CUSTARD.

5 IBMDP-optimal policies by solving fully observable MDPs

The main result of our work is to show that when using the IBMDP framework to learn a DT for a supervised classification task, there is no need to use partially obesrvable RL and that it is sufficient to use classical RL. We first define a new MDP and then show that a policy maximizing the expected cumulative reward of this MDP also maximizes the IBMDP reward.

5.1 Observation-IBMDP

Let us consider a base Classification MDP $\langle \mathcal{X}, \{C_1, ..., C_K\}, R, T, \gamma \rangle$. and an associated IBMDP $\langle \mathcal{S}', \mathcal{A}', R', T', \zeta, p, \gamma \rangle$. An Observation-IBMDP (OIBMDP) $\langle \Omega, \mathcal{A}', R'', T'', \zeta, p, \gamma \rangle$ is defined as follows:

State space The state space is the space of possible feature bounds $\Omega \subsetneq [0,1]^{2d}$.

Action space The action space is $\mathcal{A}_I$, the same as in the given IBMDP.

Reward function Assume the current state of the MDP is $o = (L_1, ..., L_d, U_1, ..., U_d)$.

- $a \in \mathcal{A}_I$: The reward for taking an IGA is still ζ.
- $a = C_h \in \{C_1, ..., C_K\}$: We denote $\mathcal{X}_o^{C_h}$ the set of all x_i such that $y_i = C_h$ and $L_k \leq x_{ik} \leq U_k$ for all k. Similarly, We denote $\mathcal{X}_o^{\bar{C_h}}$ the set of all x_i such that $y_i \neq C_h$ and $L_k \leq x_{ik} \leq U_k$ for all k. So $R''(o, C_h) = \frac{|\mathcal{X}_o^{C_h}| - |\mathcal{X}_o^{\bar{C_h}}|}{|\mathcal{X}_o^{C_h}| + |\mathcal{X}_o^{\bar{C_h}}|}$

Transition function Assume the current state of is $o = (L_1, ..., L_d, U_1, ..., U_d)$.

- $a = C_h \in \{C_1, ..., C_K\}$: $T(o, C_h, (0, ..., 0, 1, ..., 1)) = 1$

- $a = (k, \frac{u}{p+1}) \in \mathcal{A}_I$: We denote $v = \frac{u}{p+1}(U_k - L_k) + L_k$. The MDP will transit to $o_{inf} = (L_1, ..., v, ..., L_d, U_1, ..., U_d)$ (resp. $o_{sup} = (L_1, ..., L_d, U_1, ..., v, ..., U_d)$) with probability $\frac{|\mathcal{X}_{o_{inf}}|}{|\mathcal{X}_{o_{inf}}|+|\mathcal{X}_{o_{sup}}|}$ (resp. $\frac{|\mathcal{X}_{o_{sup}}|}{|\mathcal{X}_{o_{inf}}|+|\mathcal{X}_{o_{sup}}|}$)

Theorem 1 *Any optimal policy of the OIBMDP has the same policy return as $J(\pi^*)$ in the IBMDP. As such, any policy optimal w.r.t the OIBMDP reward is optimal w.r.t a certain interpretability-performance trade-off.*

5.2 Avoiding to learn large trees

We will observe from experimental results that CUSTARD [31] tends to learn large trees (more than 10 decision nodes). To avoid this problem, we learn an OIBMDPs with a maximum tree depth M_{max} and with p a prime number to leverage Proposition 1. Thus given the current observation $o = (L_1, ..., L_d, U_1, ..., U_d)$, the current depth is $M = \sum_{i=1}^{d} \log_2(\frac{1}{U_i - L_i})$. When an agent learning in an OIBMDP observes o such that $M \geq M_{max}$, it takes an action in $\{C_1, ..., C_K\}$ equivalent to adding a leaf node to the learned DT.

5.3 DQN Decision Tree

We now present a DQN [17] variant to learn DTs for OIBMDPs with a given max depth. As guaranteed by Theorem 1, any optimal policy of the OIBMDP is equivalent to the DT with the optimal interpretability-performance trade-off encoded by an IBMDP reward function. However existing algorithms returning optimal policies for MDPs [30] require the full state space to be stored in memory and the state space of OIBMDPs is of size $(2pd)^{M_{max}}$ with elements in $\mathbb{R}^d$. Experimentally, we used Policy Iteration [30] to consistently retrieve the optimal DT for the benchmarks of Section 4.1. However, when the number of features d and M_{max} grow, Policy Iteration becomes intractable. We have also tried Q-learning and SARSA [30] but did not manage to solve the benchmark of Section 4.1. To overcome this challenge we go for the Deep RL algorithm DQN [17] that we modify so that $\pi(o) = \underset{a\in\{C_1,...,C_K\}}{\text{argmax}} \; Q(o, a)$ when $M \geq M_{max}$. It is clear from Figure 4 that DQNDT outperforms CUSTARD on the simple benchmark of Section 4.1

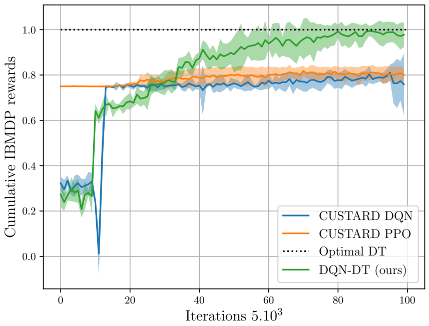


Figure 4: Comparison of CUSTARD and DQNDT on the simple binary classification benchmark of Section 4.1.

Table 1: UCI datasets: number of training samples, number of features per sample, and number of classes.

	$\lvert\mathcal{X}\rvert$	d	K
Wine	178	13	3
Diabete	520	16	2
Banknote	1372	4	2

6 Experiments on Supervised Classification Datasets

In this section, we apply DQNDT on UCI [11] datasets. We compare the learned DTs of DQNDT with trees learned by CUSTARD [31] with respect to the interpretability-performance trade-off.

6.1 Reproducibility statement

All the code to reproduce the experiments is given in the anonymous github (footnote 1). All experiments were run multiple times on independent seeds. All the versions of the necessary `python` libraries are given in a `requirements.txt` file.

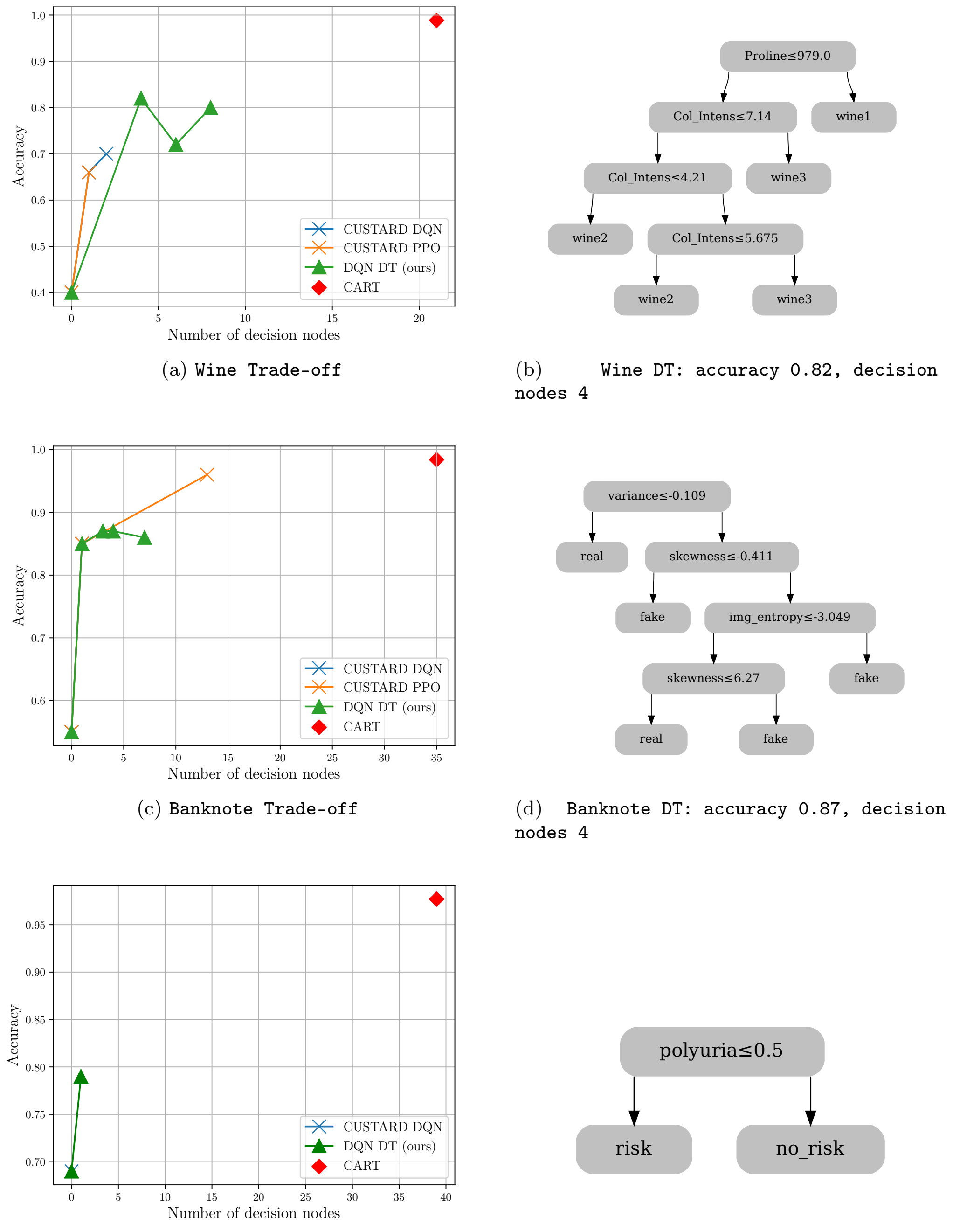
All implementations of CUSTARD and DQNDT are available in the anonymous github (footnote 1). We modify the source code of `stable-baselines3` [26] implementations of PPO and DQN with the modification of Sections 3.4.2 and 5.3. All hyperparameters of CUSTARD and DQNDT are the default hyperparameters of PPO and DQN from `stable-baselines3`.

6.2 Experimental setup

For each UCI dataset, we try 6 different values for the cost of building decision nodes: $\zeta \in \{-1, -0.6, -0.2, 0.2, 0.6, 1\}$. The splitting parameter p is always 1. Hence we solve 6 different IBMDPs (resp. OIBMDPs) with CUSTARD (resp. DQNDT). Each DT learning agent is run 5 times with 2 million timesteps. For each UCI dataset and each ζ, we analyse the best DTs obtained with each agent. We report the accuracy and the number of nodes of the best DTs and plot the interpretability-performance trade-off. For DQNDT, we fix the maximum tree depth M_{max} to 4, 5, and 5 for `Wine, Diabete, Banknote` respectively. We also compute DTs using CART (we use the implementation of [20], and fix the `max_depth` parameter to 4 or 5).

6.3 Interpretability-Performance Trade-Offs

From Figure 5, it is clear that CUSTARD learns either very small trees (2 decision nodes at most for the `Wine` dataset) or very large trees (up to 13 decision nodes for the `Banknote` dataset). And in general, for $\zeta \geq 0.6$, CUSTARD learns to repeat IGAs indefinitely. DQNDT is able to return DTs with a wider range of interpretability-performance trade-offs (DTs with 2, 4, 6, 7, 8 decision nodes). The trees returned by DQNDT are always limited in depth. As expected, CART follows a greedy approach and simply maximises the accuracy of DT resulting in non-interpretable trees with many decision nodes. All computed trees are available in the anonymous github (footnote 1).



(a) `Wine Trade-off`

(b) `Wine DT: accuracy 0.82, decision nodes 4`

(c) `Banknote Trade-off`

(d) `Banknote DT: accuracy 0.87, decision nodes 4`

(e) `Diabete Trade-off`

(f) `Diabete DT: accuracy 0.8, decision nodes 1`

Figure 5: Interpretability-performance trade-offs of DTs learned using CUSTARD, DQNDT, or CART; DTs learned with DQNDT.

7 Future work

7.1 Beyond DTs

Scaling to larger IGA spaces would allow for finer splittings at decision nodes, but could also ease the generalization of IBMDPs to more general classes of interpretable models. At the very least one could consider test nodes that also test if a feature value is within closed intervals $[v_1, v_2]$ or for example could consider combinations of more than one feature. As long as one can write the associated transition function, it is very likely that our results would extend to such settings too.

7.2 Beyond supervised learning

Finally, we stress out that provably convergent algorithms for finding an optimal IBMDP policy when the base task is a sequential decision making problem remains an open question. Perhaps one intermediate step is to study the learning of DT policies by imitation learning as in [6], which can be reduced to a sequence of supervised learning problems.

8 Conclusion

In this work we analysed and evaluated the recently proposed IBMDP framework [31] that tackles the interesting problem of learning compact DTs with RL. We showed that CUSTARD [31] can be seen as asymmetric RL [21, 3, 4]. We showed experimentally that CUSTARD fails to retrieve the optimal DT for IBMDPs of simple supervised classification base tasks. One of the main contribution of the paper is to show that this problem can be reformulated into a fully observable MDP. To scale to supervised classification tasks with large feature dimension and to avoid the learning of arbitrarily large DTs, we proposed a variant of DQN [17]: DQNDT. We showed experimentally that DQNDT is able to learn DTs with more interpretability-performances trade-offs than CUSTARD on UCI [11] datasets. Our work opens up a large avenue for future research to go beyond greedy search algorithms and explore the full space of DTs and similar models of discrete nature.

9 Ethical statement

The ethical implications of this work are the same as the DQN algorithm [17]. Furthermore since DQNDT learns an interpretable DT, we believe our work offers a new way to solve classification tasks in an ethical manner.

10 Acknowledgements

Hector Kohler acknowledges the funding from ANR AI_PhD@Lille for his PhD. Philippe Preux acknowledges the support of the Métropole Européenne de Lille (MEL), ANR, Inria, Université de Lille, through the AI chair Apprenf number R-PILOTE-19-004-APPRENF. We also acknowledge the outstanding working environment provided by Inria in the Scool research group [2].

[2] https://team.inria.fr/scool/

References

[1] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In *Conference on Learning Theory*, pages 64–66. PMLR, 2020.

[2] Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[3] Andrea Baisero and Christopher Amato. Unbiased asymmetric actor-critic for partially observable reinforcement learning. *CoRR*, 2021.

[4] Andrea Baisero, Brett Daley, and Christopher Amato. Asymmetric DQN for partially observable reinforcement learning. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 107–117. PMLR, 01–05 Aug 2022.

[5] Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux. Model interpretability through the lens of computational complexity. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[6] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *CoRR*, 2018.

[7] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106:1039–1082, 2017.

[8] Blai Bonet and Héctor Geffner. Learning sorting and decision trees with pomdps. In *ICML*, pages 73–81. Citeseer, 1998.

[9] Jeffrey P Bradford, Clayton Kunz, Ron Kohavi, Cliff Brunk, and Carla E Brodley. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, pages 131–136. Springer, 1998.

[10] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984.

[11] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[12] Abhinav Garlapati, Aditi Raghunathan, Vaishnavh Nagarajan, and Balaraman Ravindran. A reinforcement learning approach to online learning of decision trees, 2015.

[13] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 2018.

[14] Michael Kearns. Boosting theory towards practice: Recent developments in decision tree induction and the weak learning framework. In *Proceedings of the national conference on artificial intelligence*, pages 1337–1339, 1996.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[16] Zachary C. Lipton. The mythos of model interpretability, 2016.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[18] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, oct 2019.

[19] Siegfried Nijssen and Elisa Fromont. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 530–539, 2007.

[20] 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. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018.

[22] Andreas L Prodromidis and Salvatore J Stolfo. Cost complexity-based pruning of ensemble classifiers. *Knowledge and Information Systems*, 3(4):449–469, 2001.

[23] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[24] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[25] J. Ross Quinlan. C4.5: Programs for machine learning. 1992.

[26] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, 2017.

[28] Olivier Sigaud and Olivier Buffet. *Markov decision processes in artificial intelligence.* John Wiley & Sons, 2013.

[29] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[31] Nicholay Topin, Stephanie Milani, Fei Fang, and Manuela Veloso. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[32] Sicco Verwer and Yingqian Zhang. Learning decision trees with flexible constraints and objectives using integer optimization. In *Integration of AI and OR Techniques in Constraint Programming: 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings 14*, pages 94–103. Springer, 2017.

[33] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. Interpreting cnns via decision trees. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

A Proofs

A.1 Proposition 1: tree depth information in feature bounds

We want to show that in an IBMDP, if $p+1$ is prime, then the number of information-gathering actions performed since the last base action is encoded in the feature bounds part of the state. Without loss of generality we only study the case of a single feature, showing that $l = U - L$, the difference between the upper and lower bound of the feature, fully determines the number of information-gathering actions taken since the last base action. The extension to multiple features is trivial by addition of each feature's inferred number of information-gathering actions.

Let n be the number of information-gathering actions taken since the last base action. Let l_n be the difference between upper and lower bounds of the feature after these n information-gathering actions. Clearly $n = 0 \Leftrightarrow l_n = 1$. When $n > 0$, l_n is given by $l_n = \prod\limits_{k=1}^{n} \frac{h_k}{p+1}$, where $h_k \in \{1, ..., p\}$ for each k. We want to show that if $p+1$ is prime $\implies \nexists (i,j) : i < j$ and $l_i = l_j$.

$$\text{Suppose } \exists (i,j) : 0 < i < j \text{ and } l_i = l_j,$$
$$\implies \frac{\prod\limits_{k=1}^{i} h_k}{(p+1)^i} = \frac{\prod\limits_{k=1}^{j} h'_k}{(p+1)^j},$$
$$\implies \prod_{k=1}^{i} h_k = (p+1)^{j-i} \prod_{k=1}^{j} h'_k,$$
$$\implies \prod_{k=1}^{i} h_k = (p+1)(p+1)^{j-i-1} \prod_{k=1}^{j} h'_k,$$

But that is impossible since the left-hand side is the product of non-zero natural numbers $< p+1$ and the right-hand side is the product of non-zero natural numbers containing the prime number $p+1$. □

A.2 Proposition 2: gradient of soft-max reactive policy

For an MDP $\langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$ with finite state-action spaces, Lemma C.1 of [1] showed that for tabular soft-max policies

$$\frac{\partial J(\pi_\theta)}{\partial \theta(s,a)} = \frac{1}{1-\gamma} p^{\pi_\theta}(s) \pi_\theta(a|s) A^{\pi_\theta}(s,a), \tag{2}$$

where $\theta(s,a)$ is the logit parameter of the policy. Now extending this MDP into an IBMDP $\langle \mathcal{S}', \mathcal{A}', R', T', \zeta, p, \gamma \rangle$ with a fixed maximum depth as defined in Sec. 4.3, the state space remains finite. Let $s = (\phi, o) \in \mathcal{S}'$ be a state of the IBMDP, with $o \in \Omega$ where Ω is the finite set of reachable feature bounds. We are interested in tabular policies parameterized by logits $\theta' \in \mathbb{R}^{\Omega \times \mathcal{A}'}$ such that $\pi_{\theta'}(a|s) \propto \exp(\theta'(o,a))$. That is, the main difference with the setting of Eq. (2) is that a given logit $\theta'(o,a)$ is shared between several states and provides the unormalized log-probability of taking action a in all states $s \in \mathcal{S}'$ such that their feature bounds is o, i.e. such that $O(s) = o$. Informally, we can decompose this map $\theta' \mapsto J(\pi_{\theta'})$ going from logits in $\mathbb{R}^{\Omega \times \mathcal{A}'}$ to a policy return into the composition $\theta' \mapsto \theta \mapsto J(\pi_\theta)$, where the first map maps logits in $\mathbb{R}^{\Omega \times \mathcal{A}'}$ into logits in $\mathbb{R}^{\mathcal{S}' \times \mathcal{A}'}$ according to $\theta(s,a) = \theta'(O(s),a)$. By the chain rule, we have

$\nabla_{\theta'} J(\pi_{\theta'}) = H(\theta, \theta')^T \nabla_\theta J(\pi_\theta)$ where H is the Jacobian of the map $\theta' \mapsto \theta$. This Jacobian will have a value of 1 at row (s, a) and column $(O(s), a)$ for all s and a and is 0 otherwise. Thus the product simply becomes

$$\frac{\partial J(\pi_\theta)}{\partial \theta'(o, a)} = \sum_{s \in \mathcal{S}'} 1_{O(s)=o} \frac{\partial J(\pi_\theta)}{\partial \theta(s, a)}\bigg|_{\theta(s,a)=\theta'(o,a)} \tag{3}$$

Combining Eq. (2) and Eq. (3) completes the proof of Eq. (1).

A.3 Theorem 1: problem equivalence with the OIBMDP

A stochastic reactive policy of an IBMDP $\pi : \Omega \mapsto \Delta\mathcal{A}'$, where $\Delta\mathcal{A}'$ is the set of all probability distributions over $\mathcal{A}'$, can also act on the OIBMDP (Sec. 5.1) since the state and action spaces of the OIBMDP are respectively Ω and $\mathcal{A}'$. We will show in this section that any such policy has the same policy return in the IBMDP and the OIBMDP. Indeed, the construction of the OIBMDP's transition function is such that feature bounds are visited with the same frequency in the IBMDP and the OIBMDP, while the reward at state o of the OIBMDP is the average over the state rewards of the IBMDP that 'fall' within the feature bounds of o.

The proof only holds when the base MDP of the IBMDP is a supervised task because we have that for any reactive policy acting in the IBMDP, $Pr(s_t = s|o_t = o) = p(s|o)$. That is, for a policy π acting in the IBMDP, if at time-step t the observation part of the state is o, then the distribution of the random variable s_t follows the fixed distribution $p(s|o)$. This is trivial to see because on one hand, information gathering actions in an IBMDP only influence the observation part of the state, and on the other hand, all base actions induce the same next state distribution since in the supervised setting, the next data point is sampled randomly from the dataset independently from the last prediction (base action). Thus one can easily show by induction that for any policy π, $Pr(s_t = s|o_t = o)$ is independent of π, since the base state is independent of the actions of the policy.

Following Sec. 5.1, one can see that the reward and transition functions of the OIBMDP can be written as

$$R''(o, a) = \sum_{s \in \mathcal{S}'} p(s|o) R'(s, a),$$

and

$$T''(o_t, a_t, o_{t+1}) = \sum_{s, s' \in \mathcal{S}'} 1_{O(s')=o_{t+1}} p(s|o_t) T'(s, a_t, s'),$$

where $p(s|o) = \frac{1}{|\mathcal{X}_o|}$ if $O(s) = o$ and the base state of s is in $\mathcal{X}_o$ and is 0 otherwise.

Let o_t and v_t, $t \geq 0$ be the feature bounds random variables as π acts on the IBMDP and the OIBMDP respectively. We will show that for all $o \in \Omega$ and $t \geq 0, Pr(o_t = o) = Pr(v_t = o)$. By induction, it is true for $t = 0$ since the feature bounds are all initialized to $(0, 1)$ for both the

IBMDP and OIBMDP. Assume it is true for t then,

$$\begin{aligned}
Pr(o_{t+1} = o') &= \sum_{a\in\mathcal{A}'} \sum_{s,s'\in\mathcal{S}'} Pr(s_t = s)\pi(a|O(s))T'(s,a,s')1_{O(s')=o'} && (4)\\
&= \sum_{o\in\Omega} \sum_{a\in\mathcal{A}'} \sum_{s,s'\in\mathcal{S}'} Pr(s_t = s|o_t = o)Pr(o_t = o)\pi(a|O(s))T'(s,a,s')1_{O(s')=o'} && (5)\\
&= \sum_{o\in\Omega} Pr(o_t = o) \sum_{a\in\mathcal{A}'} \pi(a|o) \sum_{s,s'\in\mathcal{S}'} Pr(s_t = s|o_t = o)T'(s,a,s')1_{O(s')=o'} && (6)\\
&= \sum_{o\in\Omega} Pr(o_t = o) \sum_{a\in\mathcal{A}'} \pi(a|o) \sum_{s,s'\in\mathcal{S}'} p(s|o)T'(s,a,s')1_{O(s')=o'} && (7)\\
&= \sum_{o\in\Omega} Pr(v_t = o) \sum_{a\in\mathcal{A}'} \pi(a|o) \sum_{s,s'\in\mathcal{S}'} T''(o,a,o') && (8)\\
&= Pr(v_{t+1} = o') && (9)
\end{aligned}$$

Now let $J_O(\pi)$ be the policy return of π when acting on the OIBMDP. We have

$$\begin{aligned}
J(\pi) &= \sum_{t\geq 0} \gamma^t \sum_{s\in\mathcal{S}'} \sum_{a\in\mathcal{A}'} Pr(s_t = s)\pi(a|O(s))R(s,a) && (10)\\
&= \sum_{t\geq 0} \gamma^t \sum_{s\in\mathcal{S}'} \sum_{o\in\Omega} \sum_{a\in\mathcal{A}'} Pr(s_t = s|o_t = o)Pr(o_t = o)\pi(a|O(s))R'(s,a) && (11)\\
&= \sum_{t\geq 0} \gamma^t \sum_{o\in\Omega} \sum_{a\in\mathcal{A}'} Pr(v_t = o)\pi(a|o) \sum_{s\in\mathcal{S}'} p(s|o)R'(s,a) && (12)\\
&= \sum_{t\geq 0} \gamma^t \sum_{o\in\Omega} \sum_{a\in\mathcal{A}'} Pr(v_t = o)\pi(a|o)R''(o,a) && (13)\\
&= J_O(\pi) && (14)
\end{aligned}$$

Thus reactive policies have the same policy return in the IBMDP and the OIBMDP. Since we know that an MDP always admits a deterministic policy as a solution, an optimal policy of the OIBMDP has a return $J(\pi^*)$, the return of the best deterministic reactive policy.

Contents

RESEARCH CENTRE
LILLE – NORD EUROPE

Parc scientifique de la Haute-Borne
40 avenue Halley - Bât A - Park Plaza
59650 Villeneuve d'Ascq

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399