# Limits of reinforcement learning for decision trees in Markov decision processes

**Hector Kohler, Riad Akrour, Philippe Preux,**

**Keywords:** Reinforcement learning, decision trees, MDPs, POMDPs

## Summary

For applications like medicine, machine learning models ought to be interpretable. In that case, decision tree models are preferred over neural networks because humans can read their predictions from the root to the leaves. Training such decision trees for sequential decision making problems is a relatively new research direction and most of the existing literature focuses on imitating neural networks. In contrast, we study reinforcement learning (RL) algorithms that train decision trees *directly* optimizing some trade-off of cumulative rewards and interpretability in a Markov decision process (MDP). We show that such algorithms can be seen as training policies for partially observable Markov decision processes (POMDPs). This helps us understand why in practice it is often hard to train decision tree policies from scratch for MDPs with RL.

## Contribution(s)

1. We bridge the gap between RL for decision tree policies and POMDPs.
   **Context:** Topin et al. (2021) proposed the first RL algorithms that train decision tree policies to maximize the rewards in MDPs without having to first compute an expert policy, e.g. a neural network. Contemporarily, Pinto et al. (2017); Baisero & Amato (2022); Baisero et al. (2022) developed asymmetric RL algorithms that are specifically tailored to train policies in POMDPs. We unify those two approaches.
2. We show that RL for decision tree policies is hard because of partial observability.
   **Context:** We perform comprehensive empirical benchmarking of asymmetric and standard RL on simple decision tree policy learning tasks for which we know the optimal solutions and show that RL fails. We relate our findings to hardness results from the POMDP literature Littman (1994). Furthermore, we show that RL can consistently learn optimal decision tree policies for task that are fully observable which further corroborates our findings.

# Limits of reinforcement learning for decision trees in Markov decision processes

**Hector Kohler**[1,†], **Riad Akrour**[2], **Philippe Preux**[3],
hector.kohler@polytechnique.edu,
{riad.akrour,philippe.preux}@inria.fr

† Corresponding author
[1]**LIX, Ecole Polytechnique, CNRS, Institut Polytechnique de Paris, F-91477 Palaiseau, France**
[2]**Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 – CRIStAL, F-59000 Lille, France**
[3]**Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 – CRIStAL, F-59000 Lille, France**

## Abstract

For applications like medicine, machine learning models ought to be interpretable. In that case, decision tree models are preferred over neural networks because humans can read their predictions from the root to the leaves. Training such decision trees for sequential decision making problems is a relatively new research direction and most of the existing literature focuses on imitating neural networks. In contrast, we study reinforcement learning (RL) algorithms that train decision trees *directly* optimizing some trade-off of cumulative rewards and interpretability in a Markov decision process (MDP). We show that such algorithms can be seen as training policies for partially observable Markov decision processes (POMDPs). This helps us understand why in practice it is often hard to train decision tree policies from scratch for MDPs with RL.

## 1 Introduction

Interpretability in machine learning is often divided into local and global approaches (Glanois et al., 2024). Local methods—also referred to as explainability or post-hoc methods (Lipton, 2018)—provide explanations for individual predictions using e.g. local linear approximations (Ribeiro et al., 2016), saliency maps (Puri et al., 2020), feature attributions (Lundberg & Lee, 2017), or attention mechanisms (Shi et al., 2022). These methods approximate the behavior of an underlying black-box model and may therefore be unfaithful to its true computations (Atrey et al., 2020).

Global interpretability approaches instead restrict the model class so that the trained model is transparent by construction. Decision trees (Breiman et al., 1984) are a canonical example, as their predictions can be inspected and formally verified (Lipton, 2018). This makes them attractive for safety-critical applications and has motivated extensive research in supervised learning (Murthy & Salzberg, 1995; Demirovic et al., 2022; van der Linden et al., 2023).

Extending global interpretability to sequential decision making, however, remains challenging. Existing approaches largely rely on *indirect* methods (Milani et al., 2024): a high-performing but opaque policy (typically a neural network) is first trained using RL, and an interpretable model is then trained to imitate its behavior. A prominent example is VIPER (Bastani et al., 2018), which distills neural network policies into decision trees using imitation learning (Ross et al., 2010). Such methods have demonstrated strong empirical performance but they optimize a surrogate objective—policy imitation—rather than the MDP rewards. As a result, the best decision tree policy for the task may differ substantially from the tree that best approximates a neural expert (appendix 7).

This limitation motivates the study of approaches that train interpretable policies that *directly* maximize the MDP rewards rather than the expert's behavior likelihood (Ross et al., 2010). While direct decision tree training is well understood in supervised settings, it is far less developed for sequential decision making. Understanding why direct optimization is hard—and when it can succeed—is the central focus of this work.

In this article, we show that using RL algorithms to train decision tree policies that directly maximize the MDP rewards, without relying on a black-box expert, is often very hard. We construct very simple MDPs for which we know optimal decision tree policies and show that RL consistently fails to train those policies. We identify partial observability as a key reason for those failures.

In section 2, we present the related work on RL algorithms that train decision tree policies for MDPs. In section 3, we present key concepts for decision trees, MDPs, and the formalism from Topin et al. (2021) for training decision tree policies for MDPs with RL. In section 4, we show that this direct approach is equivalent to training a *deterministic memoryless* policy for POMDPs (Sondik, 1978). In section 5, we present our methodology to benchmark RL algorithms that train decision tree policies for MDPs. In section 5.3, we show that when RL fails to train optimal decision tree policies for MDPs it is most likely because partial observability is involved.

## 2 Related work

There exist reinforcement learning algorithms that directly train decision tree policies optimizing the cumulative rewards in a given MDP. These approaches can be divided into methods based on *parametric* and *non-parametric* trees.

Parametric decision trees fix the tree structure *a priori*—the depth and node arrangements—and only learn the node labels. This formulation enables differentiability and allows direct optimization of the MDP cumulative rewards using policy gradient methods (Sutton et al., 1999). Several works employ PPO to train such differentiable trees (Schulman et al., 2017; Silva et al., 2020; Vos & Verwer, 2024; Marton et al., 2025). While these methods can achieve strong performance, they require the tree structure to be specified in advance, making it difficult to adaptively trade off interpretability and performance. An overly complex structure may require post-hoc pruning, whereas an insufficiently expressive structure may fail to represent good policies. Moreover, Marton et al. (2025) reports that additional stabilization techniques, such as adaptive batch sizes, are often necessary for direct reinforcement learning to match indirect imitation methods performances.

Non-parametric decision trees, by contrast, are the standard model in supervised learning, where algorithms efficiently construct trees that balance predictive performance and interpretability (Breiman et al., 1984; Bertsimas & Dunn, 2017). This trade-off is optimized by using a regularized training objective. However, training non-parametric trees to trade-off MDP cumulative rewards and interpretability is largely unexplored (Milani et al., 2024). To the best of our knowledge, the only work that studies this setting is Topin et al. (2021). Topin et al. (2021) introduced *iterative bounding MDPs* (IBMDPs), which augment the downstream MDP with additional state features, actions, rewards, and transitions. They show that certain policies in the IBMDP correspond to decision tree policies for the downstream MDP. Hence, RL algorithms such as Deep Q-Networks (Mnih et al., 2015) can be used to learn such policies in IBMDPs. While the setting of direct RL is well motivated, see e.g. a toy example in section 7, the work of Topin et al. (2021) is often excluded from benchmarks because it is misunderstood. For instance, both Vos & Verwer (2024) and Laforest et al. (2025) claim that the full knowledge of the IBMDP model is required to train decision tree policies for MDPs which is not true. In fact, to the best of our knowledge, we are the first work to properly benchmark the approach of Topin et al. (2021) since its introduction. In appendix 8, we reproduce experiments from Topin et al. (2021) that already shows limits of their approach compared to indirect methods like VIPER (Bastani et al., 2018) for the CartPole MDP but our goal remains to *understand* those limits.

Finally, a few specialized methods exist for restricted problem classes. For maze-like MDPs, Mansour et al. (2022) proves the existence of optimal decision tree policies and provides a constructive algorithm. In settings where the MDP model is fully known, Vos & Verwer (2023) use planning to compute shallow parametric decision tree policies. Next, we recall useful technical material.

## 3 Technical preliminaries

### 3.1 Markov decision processes

Informally, an MDP models how an agent acts over time to achieve a goal (Bellman, 1957; Puterman, 1994). At every time step, the agent observes the environment current state and takes an action. The agent receives a reward that helps evaluate the quality of the action with respect to the goal. Finally, the agent transitions to a new state and repeats this process over time:

**Definition 1** (Markov decision process)**.** An MDP is a tuple $\mathcal{M} := \langle S, A, R, T, T_0 \rangle$. $S$ is a finite set of states representing all possible configurations of the environment. $A$ is a finite set of actions available to the agent. $R : S \times A \to \mathbb{R}$ is a deterministic reward function that assigns a real-valued reward to each state-action pair. While in general reward functions are often stochastic, in this manuscript we focus deterministic ones without loss of generality. $T : S \times A \to \Delta(S)$ is the transition function that maps state-action pairs to probability distributions over next states $\Delta(S)$. $T_0 \in \Delta(S)$ is the initial distribution over states.

Informally, we would like to act in an MDP so that we obtain as much reward as possible over time. We can formally define this objective, that we call the reinforcement learning objective, as follows:

**Definition 2** (Reinforcement learning objective)**.** Given an MDP $\mathcal{M}$, the objective is to train a policy, $\pi : S \to A$ that maximizes the expected discounted sum of rewards:

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 \sim T_0, s_{t+1} \sim T(s_t, \pi(s_t))\right]$$

Where $0 < \gamma \leq 1$ is the discount factor that trades off immediate and future rewards.

Algorithms presented in this article aim to find an optimal policy $\pi^\star \in \underset{\pi}{\operatorname{argmax}}\, J(\pi)$. In particular, RL algorithms (Sutton & Barto, 1998; Watkins & Dayan, 1992; Mnih et al., 2015) train such optimal policies using data of MDP interactions without prior knowledge of the reward and transition models. Useful quantities for such algorithms include *value* of states and actions.

**Definition 3** (Value of a state)**.** In an MDP $\mathcal{M}$, the value of a state $s \in S$ under policy $\pi$ is the expected discounted sum of rewards starting from state $s$ and following policy $\pi$:

$$V^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s, s_{t+1} \sim T(s_t, \pi(s_t))\right]$$

Applying the Markov property gives a recursive definition of the value of $s$ under policy $\pi$: $V^\pi(s) = R(s, \pi(s)) + \gamma \mathbb{E}\left[V^\pi(s') | s' \sim T(s, \pi(s))\right]$. The optimal value of a state $s \in S$, $V^\star(s)$, is the value of state $s$ when following the optimal policy $\pi^\star$ (the policy that maximizes the RL objective (definition 2)): $V^\star(s) = V^{\pi^\star}(s)$. Similarly, the optimal value of a state–action pair $(s, a) \in S \times A$, $Q^\star(s, a)$, is the value when taking action $a$ in state $s$ and then following the optimal policy: $Q^\star(s, a) = R(s, a) + \gamma \mathbb{E}\left[V^\star(s') | s' \sim T(s, a)\right]$.

### 3.2 Decision tree policies

While other interpretable policy classes exist (Kohler et al., 2025b), one conjecture from Glanois et al. (2024) is that interpretable policies are all equally hard to train because they are non-differentiable in nature. Hence, we focus on decision tree policies but our results can be extended to other interpretable policies like oblique decision trees or short programs (Verma et al., 2018).

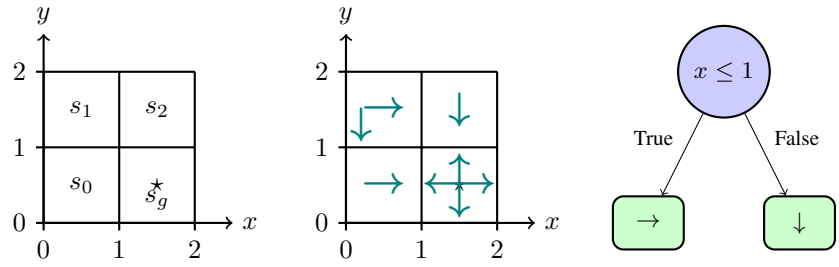



Figure 1: A grid world MDP with, four states, four directional actions that moves an agent, and rewards of 0 for every transitions except when taking an action in the bottom right goal state ($\star$). The states have discrete labels that represent their coordinates in the $[0, 2] \times [0, 2]$ square, e.g. for $s_0$, $x = 0.5$, $y = 0.5$. In the centre, optimal actions w.r.t the RL objective (definition 2). On the right, an optimal depth-1 decision tree policy w.r.t the RL objective that takes a state label as input, performs some tests on the state coordinates, and returns a directional action.

**Definition 4** (Decision tree policy)**.** A decision tree policy is a rooted tree $\pi_{\mathcal{T}} := (\mathcal{N}, E)$. Each internal node $\nu \in \mathcal{N}$ is associated with a test that maps an MDP state feature to a Boolean, e.g. $s_i \leq v$ for $v \in \mathbb{R}$. Each edge $e \in E$ from an internal node corresponds to an outcome of the associated test function. Each leaf node $l \in \mathcal{N}$ is associated with an MDP action $a_l \in A$. For any input $s \in S$, the tree defines a unique path from root to leaf, determining the prediction $\pi_{\mathcal{T}}(s) = a_l$ where $l$ is the reached leaf. The depth of a tree is the maximum path length from root to any leaf.

In figure 1, we present an example MDP, the actions that maximize the RL objective, and a decision tree policy that also maximizes the RL objective. Next, we present the class of MDPs introduced in Topin et al. (2021) which are useful to understand reinforcement learning algorithms that directly train such decision tree policies that maximize the RL objective.

### 3.3 Iterative bounding Markov decision processes

The key thing to know about iterative bounding Markov decision processes is that they are, as their name suggests, MDPs. Hence, IBMDPs admit an optimal deterministic Markovian policy–a policy that maps MDPs states to actions–that maximizes the RL objective (definition 2, Bellman (1957); Puterman (1994)). From now on, we will assume that all the MDPs we consider are MDPs with continuous state spaces and a finite set of actions, and we use bold fonts for states and observations that are vector-valued. However all our results generalize to discrete states MDPs since states can be one-hot encoded. Given an MDP for which we want to learn a decision tree policy, *the downstream MDP*, IBMDP states are concatenations of the downstream MDP state features and some observations. Those observations are information about the downstream state features that are refined–"iteratively bounded"–at each step. Those observations essentially represent some knowledge about where downstream state features lie in the state space. Actions available in an IBMDP are: (i) the actions of the downstream MDP, that change downstream state features, and (ii) *information-gathering* actions (IGAs) that change the aforementioned observations. Now, downstream actions in an IBMDP are rewarded like in the downstream MDP, this ensures that the RL objective w.r.t. the downstream MDP is encoded in the IBMDP reward. When taking an information-gathering action, the reward is an arbitrary value such that maximizing the RL objective in the IBMDP is equivalent to optimizing some trade-off between interpretability and the rewards in the downstream MDP. Before showing how to get decision tree policies from IBMDP policies, we give a formal definition of IBMDPs following Topin et al. (2021).

**Definition 5** (Iterative bounding Markov decision process (Topin et al., 2021))**.** Given a downstream MDP $\mathcal{M}$, an associated iterative bounding Markov decision process $\mathcal{M}_{IB}$ is a tuple:

$$\langle \overbrace{S \times O}^{\text{State space}}, \overbrace{A \cup A_{info}}^{\text{Action space}}, \overbrace{(R, \zeta)}^{\text{Reward}}, \overbrace{(T_{info}, T, T_0)}^{\text{Transitions}} \rangle$$

$S$ are the downstream MDP state features. Downstream state features $\boldsymbol{s} = (s_1, \ldots, s_p) \in S$ are bounded: $s_j \in [L_j, U_j]$ where $-\infty < L_j \leq U_j < \infty \; \forall 1 \leq j \leq p$. $O$ are observations. They represent bounds on the downstream state features: $O \subsetneq S^2 = [L_1, U_1] \times \cdots \times [L_p, U_p] \times [L_1, U_1] \times \cdots \times [L_p, U_p]$. So the complete IBMDP state space is $S \times O$: the concatenations of downstream state features and observations. Given some downstream state features $\boldsymbol{s} = (s_1, \ldots, s_p) \in S$ and some observation $\boldsymbol{o} = (L_1, U_1, \ldots, L_p, U_p)$, an IBMDP state is $\boldsymbol{s}_{IB} = (\overbrace{s_1, \ldots, s_p}^{\text{downstream state features}}, \overbrace{L_1, U_1, \ldots, L_p, U_p}^{\text{observation}})$. $A$ are the downstream MDP actions. $A_{info}$ are information-gathering actions of the form $\langle j, v \rangle$ where $j$ is a state feature index $1 \leq j \leq p$ and $v$ is a real number between $L_j$ and $U_j$. So the complete action space of an IBMDP is the set of downstream MDP actions and information-gathering actions $A \cup A_{info}$. $R : S \times A \to \mathbb{R}$ is the downstream MDP reward function. $\zeta$ is a reward signal for taking an information-gathering action. Hence, the IBMDP reward function gives a reward from the downstream MDP if the action is a downstream MDP action or $\zeta$ if the action is an IGA action. $T_{info} : S \times O \times (A_{info} \cup A) \to \Delta(S \times O)$ is the transition function of IBMDPs: given some observation $\boldsymbol{o}_t = (L'_1, U'_1, \ldots, L'_p, U'_p) \in O$ and downstream state features $\boldsymbol{s}_t = (s'_1, s'_2, \ldots, s'_p)$ if an IGA $\langle j, v \rangle$ is taken, the new observation $\boldsymbol{o}_{t+1}$ is $(L'_1, U'_1, \ldots, L'_j, \min\{v, U'_j\}, \ldots, L'_p, U'_p)$ if $s_j \leq v$ or $(L'_1, U'_1, \ldots, \max\{v, L'_j\}, U'_j, \ldots, L'_p, U'_p)$ if $s_j > v$. If a downstream action is taken, the observation is reset to the default downstream state feature bounds $(L_1, U_1, \ldots, L_p, U_p)$ and the downstream state features change according to the downstream MDP transition function: $\boldsymbol{s}_{t+1} \sim T(\boldsymbol{s}_t, a_t)$. At initialization, the downstream state features are drawn from the downstream MDP initial distribution $T_0$ and the observation is always set to the default downstream state features bounds $\boldsymbol{o}_0 = (L_1, U_1, \ldots, L_p, U_p)$.

We present an IBMDP for the grid-world MDP in figure 2. Now we need a way to extract a decision tree policy for the downstream MDP given a policy for an associated IBMDP.

### 3.4 From policies to trees

Information-gathering actions in IBMDPs resemble the tests that make up internal decision tree nodes (figure 2). Indeed, a policy taking actions in an IBMDP essentially builds a tree by taking sequences of IGAs (internal nodes) and then a downstream action (leaf node) and repeats this process over time. In particular, the IGA rewards $\zeta$ can be seen as a regularization or a penalty for interpretability: if $\zeta$ is very small compared to downstream rewards, a policy will try to take downstream actions as often as possible, i.e. build shallow trees with short paths between root and leaves.

Topin et al. (2021) show that not all IBMDP policies are decision tree policies for the downstream MDP. In particular, their algorithm that converts IBMDP policies into decision trees (appendix, algorithm 2) takes as input deterministic policies depending solely on the observations of the IBMDP. If the policies were not deterministic, different subtrees could stem from similar IBMDP observations: this means that the corresponding policy would be a stochastic decision tree (Blanc et al., 2021). While there is nothing wrong with learning stochastic decision tree policies from a performance standpoint, interpreting a stochastic policy is an open problem that is not our focus. If the policies were depending on the current full IBMDP state rather than solely on the current IBMDP observation, then a learning agent has 0 incentive to take information-gathering actions to build a decision tree policy as all the state information required to optimally control the downstream MDP as well as downstream actions are available to the agent. The connections between partially observable MDPs and extracting decision tree policies from IBMDPs might seem obvious but they are absent from the original IBMDP paper and from subsequent work. In the next section we bridge this gap.

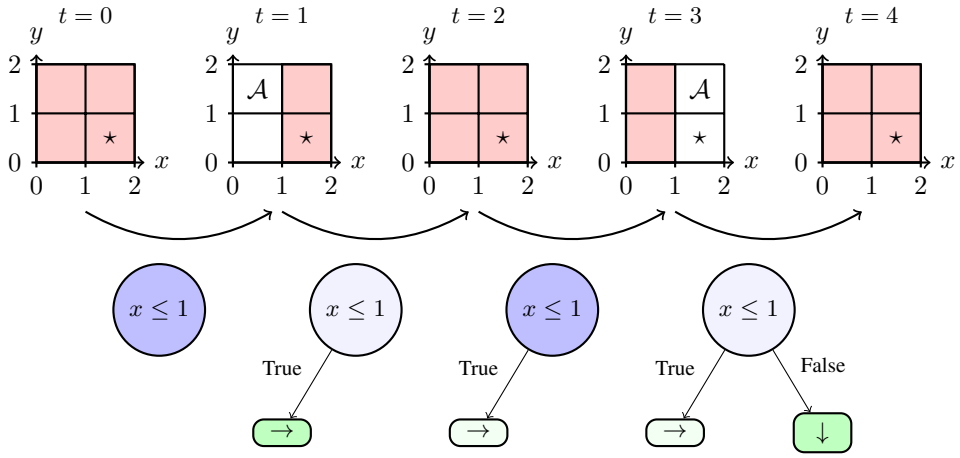



Figure 2: An IBMDP trajectory (definition 5) when the downstream MDP is the grid world from figure 1. In the top row, we represent the IBMDP state at a given time $t$. In the bottom row, we present the corresponding decision tree policy traversal. When the pink covers the whole grid, the observation $\boldsymbol{o}_t$ could be interpreted as 'the current state features could be anywhere in the grid". The more information-gathering actions are taken, the more refined the bounds on the current downstream state features get. At $t = 0$, the downstream state features are $\boldsymbol{s}_0 = (0.5, 1.5)$ and the initial observation is always: $\boldsymbol{o}_0 = (0, 2, 0, 2)$ because the downstream state features are in $[0, 2] \times [0, 2]$. This means that the overall IBMDP state is $\boldsymbol{s}_{IB} = (0.5, 1.5, 0, 2, 0, 2)$. The first action is an IGA $\langle x, 1\rangle$ that tests the feature $x$ of the downstream state against the value 1 and the reward is $\zeta$. This transition corresponds to going through an internal node $x \leq 1$ in a decision tree policy as illustrated in the figure. At $t = 1$, after gathering the information that the $x$-value of the current downstream state is below 1, the observation is updated with the refined bounds $\boldsymbol{o}_1 = (0, 1, 0, 2)$, i.e. more information has been gathered and the obstructed pink area shrinks. The downstream state features remain unchanged. The agent then takes a downstream action to move right. This gives a reward 0, resets the observation to the downstream state feature bounds, and changes the downstream state features to $\boldsymbol{s}_2 = (1.5, 1.5)$. The trajectory continues like this until the absorbing downstream state $\boldsymbol{s}_4 = (1.5, 0.5)$ is reached. By masking the current downstream state features to an agent, we force it to take information-gathering actions otherwise it could not act optimally.

## 4 Bridging the gap with the partially observable MDPs literature

To better understand RL of decision tree policies for MDPs, we formulate the problem of training a deterministic policy depending on current observations in an IBMDP as training a deterministic policy depending only on current observations–also known as a deterministic *memoryless* policy–in a partially observable Markov decision process (POMDP, Sondik (1978); Sigaud & Buffet (2013)). By doing so, we can leverage results from the POMDP literature that is richer than interpretable RL literature.

### 4.1 An adequate formalism

**Definition 6** (Partially observable Markov decision process)**.** A partially observable Markov decision process is a tuple $\langle X, A, O, R, T, T_0, \Omega\rangle$. $X$ is the hidden state space. $A$ is a finite set of actions. $O$ is a set of observations. $T : X \times A \to \Delta(X)$ is the transition function, where $T(\boldsymbol{x}_t, a, \boldsymbol{x}_{t+1}) = P(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, a)$ is the probability of transitioning to state $\boldsymbol{x}_t$ when taking action $a$ in state $\boldsymbol{x}$. $T_0$: is the initial distribution over states. $\Omega : X \to \Delta(O)$ is the observation function, where $\Omega(\boldsymbol{o}, a, \boldsymbol{x}) = P(\boldsymbol{o}|\boldsymbol{x}, a)$ is the probability of observing $\boldsymbol{o}$ in state $\boldsymbol{x}$. $R : X \times A \to \mathbb{R}$ is the reward function, where $R(\boldsymbol{x}, a)$ is the immediate reward for taking action $a$ in state $\boldsymbol{x}$. Note that $\langle X, A, R, T, T_0\rangle$ defines an MDP (definition 1).

Next, we define partially observable iterative bounding Markov decision processes (POIBMDPs): IBMDPs for which we explicitly define an observation space and an observation function.

**Definition 7** (Partially observable iterative bounding Markov decision process)**.** a partially observable iterative bounding Markov decision process $\mathcal{M}_{POIB}$ is a tuple:

$$\langle \overbrace{S \times O}^{\text{States}}, \overbrace{A \cup A_{info}}^{\text{Action space}}, \overbrace{O}^{\text{Observations}}, \overbrace{(R, \zeta)}^{\text{Rewards}}, \overbrace{(T_{info}, T, T_0)}^{\text{Transitions}}, \Omega \rangle$$

Where $\langle S \times O, A \cup A_{info}, (R, \zeta), (T, T_0, T_{info}) \rangle$ is an IBMDP (definition 5). The transition function $\Omega$ maps concatenation of state features and observations–IBMDP states–to observations, $\Omega : S \times O \to O$, with $P(\boldsymbol{o}|(\boldsymbol{s}, \boldsymbol{o})) = 1$

POIBMDPs are particular instances of POMDPs where the observation function simply applies a mask over some features of the hidden state. This setting has other names in the literature. For example, POIBMDPs are mixed observability MDPs (Araya-López et al., 2010) with downstream MDP state features as the *hidden variables* and feature bounds as *visible* variables. POIBMDPs can also be seen as a special case of non-stationary MDPs (N-MDPs) (Singh et al., 1994) in which there is one unique transition function per downstream MDP state: these are sometimes called hidden-mode MDPs (Choi et al., 2001). Following Singh et al. (1994), we can write the value of a deterministic memoryless policy $\pi : O \to A \cup A_{info}$ in observation $\boldsymbol{o}$.

**Definition 8** (Value of an observation)**.** In a POIBMDP, the expected cumulative discounted reward of a deterministic memoryless policy $\pi : O \to A \cup A_{info}$ starting from observation $o$ is $V^\pi(\boldsymbol{o})$: $V^\pi(\boldsymbol{o}) = \sum_{(s, \boldsymbol{o}') \in S \times O} P^\pi((\boldsymbol{s}, \boldsymbol{o}')|\boldsymbol{o}) V^\pi((\boldsymbol{s}, \boldsymbol{o}'))$ With $P^\pi((\boldsymbol{s}, \boldsymbol{o}')|\boldsymbol{o})$ the asymptotic occupancy distribution (see section 4 from Singh et al. (1994) for the full definition) of the hidden POIBMDP state $(\boldsymbol{s}, \boldsymbol{o}')$ given the partial observation $o$ and $V^\pi((s, o'))$ the classical state-value function (definition 3). We abuse notation and denote both values of observations and values of states by $V$ since the function input is not ambiguous.

### 4.2 Reinforcement learning in POMDPs

In general, the policy that maximizes the RL objective in a POMDP maps "belief states" or observation histories to actions (Sigaud & Buffet, 2013). Hence, those policies do not correspond to decision trees since we require that policies depend only on the current observation (algorithm 2). If we did not have this constraint, we could apply any standard RL algorithm to solve POIBMDPs by seeking policies depending on belief states or observations histories because those are sufficient to optimally control any POMDP (Sigaud & Buffet, 2013; Lambrechts et al., 2025a).

In particular, the problem of finding the optimal deterministic memoryless policies for POMDPs is NP-HARD, even with full knowledge of transitions and rewards (section 3.2 from Littman (1994)). It means that it is impractical to enumerate all possible policies and take the best one. For even moderate-sized POMDPs, a brute-force approach would take a very long time since there are $|A|^{|O|}$ deterministic memoryless policies. Hence it is interesting to study RL for training deterministic memoryless policies since it would not enumerate the whole solution space.

In Singh et al. (1994), authors show that the optimal memoryless policy can be stochastic. Hence, policy gradient algorithms (Sutton et al., 1999)–that return stochastic policies–should be avoided since we seek the best *deterministic* policy. Furthermore, the optimal deterministic memoryless policy, unlike for standard MDPs, might not maximize the values of all observations simultaneously (Singh et al., 1994) which makes it difficult to use temporal difference (TD) learning algorithms like Q-learning (Watkins & Dayan, 1992). Indeed, doing a TD update of an observation's value can change the value of *all* other observations in an uncontrollable manner because of the dependency in $P^\pi((s, \boldsymbol{o}')|\boldsymbol{o})$ from definition 8.

Despite those hardness results, applying RL to POMDPs, by naively replacing the processes (hidden) states $\boldsymbol{x}$ by the observation $\boldsymbol{o}$ in Q-learning or Sarsa (Sutton & Barto, 1998), has already demonstrated successful in practice (Loch & Singh, 1998). More recently, the framework of asymmetric

RL (Pinto et al., 2017; Baisero et al., 2022; Baisero & Amato, 2022) has shown promising results to train policies for POMDPs. Asymmetric RL algorithms train a model–a policy or a value function–depending on hidden state (only available at train time) and a history dependent (or observation dependent) model to be deployed at test time. The history or observation dependent model serves as target or critic to train the hidden state dependent model. The history dependent (or observation dependent) model can thus be deployed in the POMDP after training since it does not require access to the hidden state to output actions. In appendix 11, we present asymmetric variants of standard tabular RL algorithms. For example, algorithm 1, is a variant of Q-learning that returns a deterministic memoryless policy. Given a POMDP, asymmetric Q-learning trains a partially observable Q-function $Q : O \times A \to \mathbb{R}$ and a Q-function $U : X \times A \to \mathbb{R}$. The hidden-state-dependent Q-function $U$ serves as a target in the TD learning update. It is the tabular version of the modified DQN algorithm used in Topin et al. (2021). Indeed, when training deterministic memoryless policies for IBMDPs, Topin et al. (2021) used algorithms equivalent to asymmetric DQN before those were formally introduced in Baisero et al. (2022); Baisero & Amato (2022).

**Algorithm 1:** Asymmetric Q-Learning (Baisero et al., 2022). We highlight in green the differences with the standard Q-learning (Watkins & Dayan, 1992).

**Data:** A POMDP, learning rates $\alpha_u,\ \alpha_q$, exploration prob. $\epsilon$
**Result:** $\pi : O \to A$
Initialize $U(\boldsymbol{x}, a) = 0$ for all $x \in X, a \in A$
Initialize $Q(\boldsymbol{o}, a) = 0$ for all $o \in O, a \in A$
**for** *each episode* **do**
  Initialize state $x_0 \sim T_0$
  Initialize observation $\boldsymbol{o}_0 \sim \Omega(\boldsymbol{x}_0)$
  **for** *each step* $t$ **do**
    Choose action $a_t$ using $\epsilon$-greedy: $a_t = \operatorname{argmax}_a Q(\boldsymbol{o}_t, a)$ with prob. $1 - \epsilon$
    Take action $a_t$, observe $r_t = R(\boldsymbol{x}_t, a_t)$, $x_{t+1} \sim T(x_t, a_t)$, and $\boldsymbol{o}_{t+1} \sim \Omega(\boldsymbol{x}_{t+1})$
    $y \leftarrow r + \gamma U(\boldsymbol{x}_{t+1}, \operatorname{argmax}_{a'} Q(\boldsymbol{o}_{t+1}, a'))$
    $U(\boldsymbol{x}_t, a_t) \leftarrow (1 - \alpha_u) U(\boldsymbol{x}_t, a_t) + \alpha_u y$
    $Q(\boldsymbol{o}_t, a_t) \leftarrow (1 - \alpha_q) Q(\boldsymbol{o}_t, a_t) + \alpha_q y$
    $x_t \leftarrow \boldsymbol{x}_{t+1}$
    $\boldsymbol{o}_t \leftarrow \boldsymbol{o}_{t+1}$
  **end**
**end**
$\pi(o) = \operatorname{argmax}_a Q(\boldsymbol{o}, a)$

Until recently, the benefits of asymmetric RL over standard RL was only shown empirically and only for history-dependent models. The work of Lambrechts et al. (2025b) proves that some asymmetric RL algorithms should in theory learn better history-dependent **or** memoryless policies for POMDPs which is exactly what we wish for. However, those algorithms are impractical because they require estimations of the asymptotic occupancy distribution $P^\pi((s, \boldsymbol{o}')|\boldsymbol{o})$ (definition 8) for candidate policies which in turn would require to perform costly Monte-Carlo estimations. We leave it to future work to use those algorithms that combine asymmetric RL and estimation of future visitation frequencies since those results are contemporary to the writing of this manuscript.

Bridging the gap between, training decision tree policies for MDPs with RL through the formalism of IBMDPs (sections 3.3 and 3.4), and, training deterministic memoryless policies for POMDPs with RL gave us insights on the hardness of the former. More importantly, it gave us insights on what RL algorithms can be used to train decision tree policies. In the next section, we benchmark those algorithms for training decision tree policies.

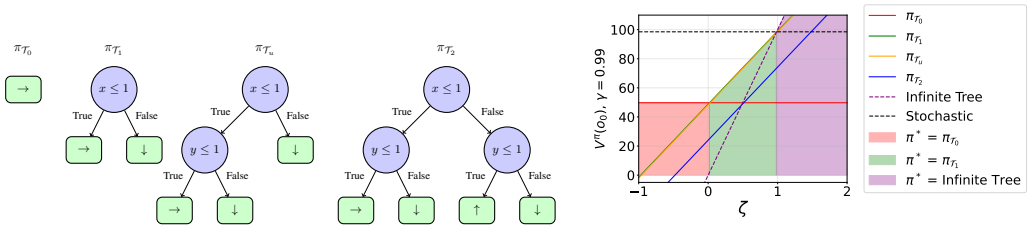



Figure 3: On the left, decision tree policies for the grid world MDP from figure 1. On the right, their RL objective values in associated POIBMDPs exemplified in figure 2 as functions of the reward term $\zeta$. Shaded areas on the right plot indicate which deterministic memoryless policy is optimal depending on $\zeta$. Recall that $\zeta$ can be seen as a reward that encourages the deterministic memoryless policy to take information-gathering actions, i.e. the reward that trades off interpretability of the corresponding decision tree and its cumulative reward (definitions 5 and 7).

# 5 Methodology

The goal of this section is to observe if RL can consistently train optimal decision tree policies for a simple MDP. In particular, we train decision tree policies that maximize the RL objective in the MDP by training deterministic memoryless policies in POIBMDPs.

## 5.1 Computing some decision tree policies

To assess the performances of the RL training, we manually compute the values of optimal deterministic memoryless policies in POIBMDPs. Each of those policies correspond to one of the decision tree policies illustrated in figure 3: (i) a depth-0 tree $\pi_{\mathcal{T}_0}$ that always take the same downstream action, (ii) a depth-1 tree $\pi_{\mathcal{T}_1}$ that alternates between an IGA and a downstream action, (iii) an unbalanced depth-2 tree $\pi_{\mathcal{T}_u}$ that sometimes takes two IGAs then a downstream action and sometimes one IGA followed by a downstream action, (iv) a depth-2 tree $\pi_{\mathcal{T}_2}$ that alternates between taking two IGAs and a downstream action, or (v) an infinite 'tree" that only takes IGAs. We will particularly focus on trying to train a depth-1 decision tree that is the most interpretable–smallest number of nodes and shallowest–tree that takes optimal actions (figure 1). Because we know from Singh et al. (1994) that stochastic memoryless policies can sometimes get better expected discounted rewards than deterministic memoryless policies in POMDPs, we also compute the value of the stochastic policy that randomly alternates between two downstream actions: $\rightarrow$ and $\downarrow$. Taking those two downstream actions always lead to the goal state in expectation. After doing the calculations of the RL objective values using the known model of the IBMDP from figure 2, we plot in figure 3 the RL objective values of the decision tree policies in POIBMDPs as functions of $\zeta$ when we fix $\gamma = 0.99$ (standard choice of discount in practice Sutton & Barto (1998)). Despite objective values being very similar for the depth-1 and unbalanced depth-2 tree, we now know from the green shaded area that a depth-1 tree is optimal when $0 < \zeta < 1$ in the POIBMDP. Interestingly, two challenges of learning in POMDPs described in Singh et al. (1994) are visible in figure 3. First, there is a whole range of $\zeta$ values for which the optimal memoryless policy is stochastic. Second, for e.g. $\zeta = 0.5$, while the optimal deterministic memoryless policy corresponds to a depth-1 tree, the value of the POIBMDP state $(\boldsymbol{s}_2, \boldsymbol{o}_0) = (1.5, 1.5, 0, 2, 0, 2)$–the agent is in the upper right cell but it does not know it–is not maximized by the optimal memoryless policy that will take one information-gathering action in between each downstream action, but by the sub-optimal policy that always goes down. Next, we present the specific experimental setup that we use in the remaining of the paper.

### 5.2 Experimental setup

All our code is open source[1]. The downstream MDP for which we want to train decision tree policies with RL is the grid world MDP from figure 1. We then define 100 associated POIBMDPs using the model of the IBMDP from figure 2 with $\zeta$ chosen uniformly among 100 different in $]0, 1[$ (figure 3).

First we use standard tabular RL algorithms, namely Q-learning, Sarsa, and vanilla policy gradient with a softmax policy (Watkins & Dayan, 1992; Sutton & Barto, 1998; Jaakkola et al., 1994), to learn deterministic memoryless policies in POIBMDPs by simply replacing the current state in the algorithm descriptions by the current observation (Loch & Singh, 1998). In theory the policy gradient algorithm should not be a good candidate for our problem since it searches for stochastic policies that we showed to be better than our sought depth-1 decision tree policy on figure 3, but for completeness we use trees obtained after "greedyfication" of the stochastic policies. Second, we use the more specialized asymmetric Q-learning, asymmetric Sarsa, and asymmetric policy gradient (section 4.2). Each algorithm is trained until convergence on the 100 POIBMDPs uniquely defined by their trade-off reward $\zeta$. Each of those training run is repeated 100 times for a total of 10 000 training runs. For all baselines we use, when applicable, exploration rates $\epsilon = 0.3$ and learning rates $\alpha = 0.1$. The detailed training curves are presented in appendix 10.

To evaluate the performances of the RL training, we consider, for each POIBMDP, the distribution of the trained trees over the 100 training seeds. Indeed, since for each POIBMDP we run each RL algorithm 100 times, after convergence we get 100 deterministic memoryless policies from which we can extract the equivalent 100 decision tree policies using algorithm 2. Then we can count the occurrences of each tree policy from figure 3. This helps us understand which trees RL algorithms tend to train as a function of the trade-off reward $\zeta$. Next, we present our results.

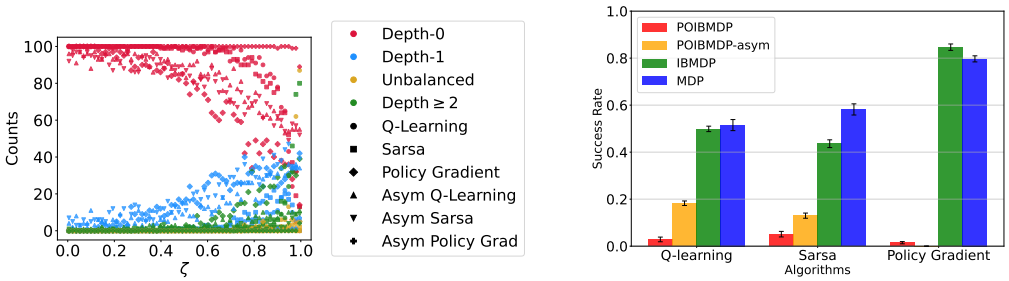
### 5.3 Can RL train optimal deterministic memoryless policies in POIBMDPs?

In figure 4a, we plot the distributions of the final trained trees as functions of $\zeta$. For example, in figure 4a, when training 100 times in a POIBMDP with $\zeta = 0.5$, Q-learning returned almost 100 times a depth-0 tree. For none of the tested baselines do we see a high rate of trained depth-1 trees for $\zeta \in ]0, 1[$. It is alarming that the most frequent trained trees are the depth-0 trees for $\zeta \in ]0, 1[$ because such trees are way more sub-optimal than e.g. the depth-2 unbalanced trees according to our computations on figure 3. One interpretation of this phenomenon is that the training in POIBMDPs is very hard and so RL tend to converge to trivial policies, e.g., repeating the same downstream action. Furthermore, in appendix 10 we show that for POIBMDPs with $\zeta \in [-1, 0]$ and $\zeta \in [1, 2]$, baselines consistently learn the optimal policies–a depth-0 tree and an infinite tree respectively–which is concerning as it means that RL seem to find only trivial policies. On the positive side, we observe that asymmetric versions of Q-learning and Sarsa have found the optimal deterministic memoryless policy–the depth-1 decision tree–more frequently throughout the optimality range $]0, 1[$, than their symmetric counter-parts for $\zeta \in ]0, 1[$. Next, we quantify how hard it is to train memoryless policies in POIBMDPs as opposed to training standard Markovian policies in MDPs.

### 5.4 How hard is it train optimal deterministic memoryless policies in POIBMDPs with RL?

In this section we run the same (asymmetric) RL algorithms to train deterministic Markovian policies–the standard policy class for most sequential decision making (Sutton & Barto, 1998)–in MDPs or IBMDPs, or deterministic memoryless policies in POIBMDPs. In order to see how hard each of these tasks is, we can run a *great* number of training runs for each and compare solving rates. To make solving rates comparable we consider a unique instance for each of those tasks. Task 1 is training a Markovian deterministic policy ($\pi : S \to A$) that maximizes the RL objective in the grid world MDP with $\gamma = 0.99$. Task 2 is training a Markovian deterministic policy ($\pi : S \times O \to A \cup A_{info}$) that maximizes the RL objective for the IBMDP from figure 2 with $\gamma = 0.99$ and $\zeta = 0.5$. Task 3 is what has been done in the previous section to train deterministic

[1] https://github.com/KohlerHECTOR/poibmdps




(a) (b)

Figure 4: (4a) Distribution of the trained trees over 100 random seeds for different values of $\zeta$ and different RL algorithms. (4b) Success rates of different (asymmetric) RL algorithms over thousands of runs when applied to training deterministic memoryless policies in a POIBMDP or training deterministic policies in associated MDP and IBMDP.

memoryless policies ($\pi : O \to A \cup A_{info}$) that maximizes the RL objective in POIBMDPs where in addition of fixing $\gamma = 0.99$ we also fix $\zeta = 0.5$.

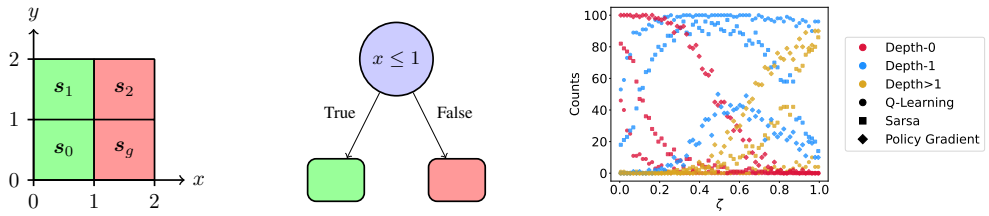
We try a wide set of hyperparameters and additional training tricks (optimistic Q-function, eligibility traces, entropy regularization and $\epsilon$-decay, all are described in Sutton & Barto (1998)). The complete detailed lists of hyperparameters are given in the appendix 11. Furthermore, the careful reader might notice that there is no point running asymmetric RL on MDPs or IBMDPs when the task does not require training a memoryless policy. Hence, we only run asymmetric RL for POIBMDPs. Each unique hyperparameter combination for a given algorithm on a given task is run 10 times on 1 million training steps to get standard errors. We have for example 768 hyperparamters combinations for asymmetric Sarsa. Hence, we train a total of $10 \times 768 = 7680$ deterministic memoryless policies for the POIBMDP. To get a success rate, we can simply divide the number of trained policy corresponding to an optimal depth-1 decision tree policy by 768 (recall that for $\gamma = 0.99$ and $\zeta = 0.5$, the optimal policy is a depth-1 tree, c.f. figure 3).

The key observations from figure 4b is that training a deterministic memoryless policy in a POIBMDP with RL, is way harder than training a Markovian policy. For example, Q-learning converges to the optimal deterministic memoryless policy in the POIBMDP only 3% of the time while the same algorithm in an MDP or IBMDP converges to the optimal policy 50% of the time. Even though asymmetry seems to increase performances; training a decision tree policy for a simple grid world directly with RL using the framework of POIBMDP originally developed in Topin et al. (2021) seems way too hard and costly as successes might require a million steps for such a seemingly simple problem. Next, we further support that partial observability is the main limitation for using RL to train decision tree policies that maximize the RL objective in MDPs.

### 5.5 Are there downstream MDPs for which partial observability is not a limitation?

In this section, we show that for a special class of POIBMDP, RL can consistently train deterministic memoryless policies maximizing the RL objective. It means that we can train decision tree policies to directly maximize the RL objective in MDPs without imitating experts. This class of POIBMDPs are those for which downstream MDPs have uniform transitions, i.e. $T(\boldsymbol{s}, a, \boldsymbol{s}') = \frac{1}{|S|}$. Such downstream MDPs include classification tasks formulated as MDPs. This implies that training deterministic memoryless policies in POIBMDPs where the downstream MDP encodes a classification task is equivalent to classical decision tree induction with supervised learning (Breiman et al., 1984). This is exactly what is done in e.g. Kohler et al. (2025a). In figure 5a, we give an example of such downstream MDPs for a binary classification task with 4 data in the training set.

In appendix 12, we show that one can rewrite POIBMDPs associated to downstream MDPs with uniform transitions as standard MDPs (definition 1). This means that in principle, standard RL




(a) (b)

Figure 5: (5a) In this MDP, there are four data points to classify with either a green or red label. On the right, there is the unique optimal depth-1 tree for this particular MDP which also maximizes the accuracy on the corresponding classification task. (5b) We reproduce the same plot as in figure 4a for POIBMDPs associated to the downstream MDP from figure 5a.

algorithms like Q-learning, should work as well as for any MDP (Watkins & Dayan, 1992; Sutton & Barto, 1998). If RL does work for such fully observable POIBMDPs, this would mean that the hardness of training decision tree policies for *any* MDP using POIBMDPs, is most likely due to the partial observability. This is exactly what we show next.

We define POIBMDPs for the classification task from figure 5a, with $\gamma = 0.99$, 200 values of $\zeta \in [0, 1]$ and IGAs $x \leq 1$ and $y \leq 1$. Since those POIBMDPs are MDPs, we do not need to study asymmetric RL. We see on figure 5b that, compared to results for general POIBMDPs from figure 4a, RL can consistently train optimal deterministic memoryless policies $O :\to A \cup A_{info}$ which are equivalent to decision tree classifiers. This supports partial observability being the main limitation for training decision tree policies that directly maximize the RL objective in MDPs.

## 6 Conclusion

In this paper, we studied RL algorithms that train decision tree policies to directly optimize some trade-off of interpretability and performance in MDPs rather than to match an expert's behavior. Starting from the IBMDP formulation from Topin et al. (2021), we showed that such algorithms essentially train deterministic memoryless policies in POMDPs that we called POIBMDPs. By bridging the gap with the POMDP literature, we gave strong insights on the hardness of this task. We supported those insights by benchmarking those RL algorithms on carefully crafted problems for which we knew the optimal decision tree policies. Across our experiments, we found that only when partial observability is absent from the task can good decision tree policies be trained. Attempting to overcome the partial observability challenges seems like a bad research direction. Indeed, while algorithms tailored specifically for the problem of training deterministic memoryless policies for POIBMDPs might exist, imitation learning works well in practice and has been the state-of-the-art for interpretable sequential decision making for a while. Furthermore there are other limitations to IBMDPs that we did not cover such as how to choose good candidates information-gathering actions when building the decision tree policy or simply how to choose $\zeta$ for a target trade-off between tree interpretablility and downstream performance. To conclude, we recommend the community to focus on imitation learning or parametric trees approaches for future research.

### Acknowledgments

The authors would like to acknowledge the Scool team for its great working environment. This work has also been supported by the French Ministry of Higher Education and Research, the Hauts-de-France region, Inria, the MEL, and the CornelIA project that partially funds Grid'5000. The results of this paper were produced while Hector Kohler was still a Ph.D. student in Inria Lille. The paper was written while Hector Kohler was a postdoctoral researcher in LIX. Hector Kohler acknowledges the AI_PhD@Lille ANR funding and ANR-24-CE23-2874-01. Experiments presented in this paper were carried out using the Grid'5000 testbed (https://www.grid5000.fr).

## References


Mauricio Araya-López, Vincent Thomas, Olivier Buffet, and François Charpillet. A Closer Look at MOMDPs. In *Proceedings of the 22nd International Conference on Tools with Artificial Intelligence*, Proceedings of the 22nd International Conference on Tools with Artificial Intelligence, Arras, France, October 2010. IEEE. URL https://inria.hal.science/inria-00535559.

Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl3m1BFDB.

Andrea Baisero and Christopher Amato. Unbiased asymmetric reinforcement learning under partial observability. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 44–52, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Andrea Baisero, Brett Daley, and Christopher Amato. Asymmetric DQN for partially observable reinforcement learning. In James Cussens and Kun Zhang (eds.), *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 107–117. PMLR, 01–05 Aug 2022. URL https://proceedings.mlr.press/v180/baisero22a.html.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. DOI: 10.1109/TSMC.1983.6313077.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. 2018.

Richard Bellman. *Dynamic Programming*. 1957.

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

Guy Blanc, Jane Lange, and Li-Yang Tan. Learning stochastic decision trees. *CoRR*, abs/2105.03594, 2021. URL https://arxiv.org/abs/2105.03594.

L Breiman, JH Friedman, R Olshen, and CJ Stone. *Classification and Regression Trees*. Wadsworth, 1984.

Samuel Ping-Man Choi, Nevin Lianwen Zhang, and Dit-Yan Yeung. Solving hidden-mode markov decision problems. In Thomas S. Richardson and Tommi S. Jaakkola (eds.), *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, volume R3 of *Proceedings of Machine Learning Research*, pp. 49–56. PMLR, 04–07 Jan 2001. URL https://proceedings.mlr.press/r3/choi01a.html. Reissued by PMLR on 31 March 2021.

Emir Demirovic, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J. Stuckey. Murtree: Optimal decision trees via dynamic programming and search. *Journal of Machine Learning Research*, 23(26):1–47, 2022. URL http://jmlr.org/papers/v23/20-520.html.

Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *Machine Learning*, pp. 1–44, 2024.

Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable markov decision problems. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'94, pp. 345–352, Cambridge, MA, USA, 1994. MIT Press.

Hector Kohler, Riad Akrour, and Philippe Preux. Breiman meets bellman: Non-greedy decision trees with mdps. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 1207–1218, New York, NY, USA, 2025a. Association for Computing Machinery. ISBN 9798400714542. DOI: 10.1145/3711896.3736868. URL https://doi.org/10.1145/3711896.3736868.

Hector Kohler, Waris Radji, Quentin Delfosse, Riad Akrour, and Philippe Preux. Evaluating interpretable reinforcement learning by distilling policies into programs. In *RLC 2025 Workshop on Programmatic Reinforcement Learning*, 2025b. URL https://openreview.net/forum?id=n1CfzixauT.

Geoffrey Laforest, Olivier Buffet, Alexandre Niveau, and Bruno Zanuttini. Post-Hoc Interpretation of POMDP Policies. In I. Lynce and A. Murano (eds.), *Proc. 28th European Conference on Artificial Intelligence (ECAI 2025)*, Bologna (ITALY), Italy, October 2025. IOS Press. URL https://hal.science/hal-05197856.

Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Informed POMDP: Leveraging additional information in model-based RL. *Reinforcement Learning Journal*, 2:763–784, 2025a.

Gaspard Lambrechts, Damien Ernst, and Aditya Mahajan. A theoretical justification for asymmetric actor-critic algorithms. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=F1yANMCnAn.

Zachary C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Michael L. Littman. Memoryless policies: theoretical limitations and practical results. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3: From Animals to Animats 3*, SAB94, pp. 238–245, Cambridge, MA, USA, 1994. MIT Press. ISBN 0262531224.

John Loch and Satinder P. Singh. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pp. 323–331, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Yishay Mansour, Michal Moshkovitz, and Cynthia Rudin. There is no accuracy-interpretability tradeoff in reinforcement learning for mazes, 2022. URL https://arxiv.org/abs/2206.04266.

Sascha Marton, Tim Grams, Florian Vogt, Stefan Lüdtke, Christian Bartelt, and Heiner Stuckenschmidt. Mitigating information loss in tree-based reinforcement learning via direct optimization. 2025. URL https://openreview.net/forum?id=qpXctF2aLZ.

Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Comput. Surv.*, 56(7), April 2024. ISSN 0360-0300. DOI: 10.1145/3616864. URL https://doi.org/10.1145/3616864.

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.

Sreerama Murthy and Steven Salzberg. Lookahead and pathology in decision tree induction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pp. 1025–1031, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603638.

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.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning, 2017. URL https://arxiv.org/abs/1710.06542.

Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgzLkBKPB.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. pp. 1135–1144, 2016. DOI: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. 2010.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Wenjie Shi, Gao Huang, Shiji Song, Zhuoyuan Wang, Tingyu Lin, and Cheng Wu. Self-supervised discovering of interpretable features for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2712–2724, 2022. DOI: 10.1109/TPAMI.2020.3037898.

Olivier Sigaud and Olivier Buffet. *Partially Observable Markov Decision Processes*, chapter 7, pp. 185–228. John Wiley & Sons, Ltd, 2013. ISBN 9781118557426. DOI: https://doi.org/10.1002/9781118557426.ch7. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118557426.ch7.

Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1855–1865. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/silva20a.html.

Satinder P. Singh, Tommi S. Jaakkola, and Michael I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, pp. 284–292, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1558603352.

Edward J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/169635.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

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*, 35:9923–9931, 2021.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Jacobus van der Linden, Mathijs de Weerdt, and Emir Demirović. Necessary and sufficient conditions for optimal decision trees using dynamic programming. *Advances in Neural Information Processing Systems*, 36:9173–9212, 2023.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. pp. 5045–5054, 2018.

Daniël Vos and Sicco Verwer. Optimal decision tree policies for markov decision processes. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023. ISBN 978-1-956792-03-4. DOI: 10.24963/ijcai.2023/606. URL https://doi.org/10.24963/ijcai.2023/606.

Daniël Vos and Sicco Verwer. Optimizing interpretable decision tree policies for reinforcement learning. 2024. URL https://arxiv.org/abs/2408.11632.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

# Supplementary Materials

*The following content was not necessarily subject to peer review.*

## 7 Imitation learning: a baseline for indirect decision tree policy learning

In this section we present decision tree policies of this manuscript obtained using Dagger or VIPER (Ross et al., 2010; Bastani et al., 2018) after learning an expert Q-function for the grid world MDP. Recall the optimal policies for the grid world, taking the green actions in each state in figure 1. Among the optimal policies, the ones that go left or up in the goal state can be problematic for imitation learning algorithms. Indeed, we know that for this grid world MDP there exists decision tree policies with a very good interpretability-performance trade-off: depth-1 decision trees that are optimal w.r.t. the RL objective. One could even say that those trees have the *optimal* interpretability-performance trade-off because they are the shortest trees that are optimal w.r.t. the RL objective.

In figure 6, we present a depth-1 decision tree policy that is optimal w.r.t. the RL objective and a depth-1 tree that is sub-optimal. The other optimal depth-1 tree is to go right when $y \leq 1$ and down otherwise.

Now a fair question is: can Dagger or VIPER learn such an optimal depth-1 tree given access to an expert optimal policy from figure 1?

We start by running the standard Q-learning algorithm with $\epsilon = 0.3$, $\alpha = 0.1$ over 10,000 time steps. While for Q-learning, Sutton and Barto break ties by index value in their book (Sutton & Barto, 1998) (the greedy action is the $\mathrm{argmax}$ action with smallest index), we show that the choice of tie-breaking greatly influences the performance of subsequent imitation learning algorithms. Indeed, depending on how actions are ordered in practice, Q-learning may be biased toward some optimal policies rather than others. While this does not matter for one who just wants to find an optimal policy, in our example of finding the optimal depth-1 decision tree policy, it matters *a lot*.

In the left plot of figure 7, we see that Q-learning, independently of how ties are broken, consistently converges to an optimal policy over 100 runs (random seeds). However, in the right plot of figure 7, where we plot the proportion over 100 runs of optimal decision trees returned by Dagger or VIPER at different stages of Q-learning, we observe that imitating the optimal policy obtained by breaking ties at random consistently yields more optimal trees than breaking ties by indices. What actually happens is that the most likely output of Q-learning when ties are broken by indices is the optimal policy that goes left in the goal state, which cannot be perfectly represented by a depth-1 decision tree, because there are three different actions taken and a binary tree of depth $D = 1$ can only map to $2^D = 2$ labels.

This short experiment shows that imitation learning approaches can sometimes be very bad at learning decision tree policies with good interpretability-performance trade-offs for very simple MDPs. Despite VIPER almost always finding the optimal depth-1 decision tree policy in terms of the RL objective when ties are broken at random, we have shed light on the sub-optimality of indirect approaches such as imitation learning. This motivates the study of direct approaches to directly search for policies with good interpretability-performance trade-offs with respect to the original RL objective.

## 8 Reproducing "Iterative Bounding MDPs: Learning Interpretable Policies via Non-Interpretable Methods"

We attempt to reproduce the results from Topin et al. (2021) in which authors compare direct and indirect learning of decision tree policies of depth at most 2 for the CartPole MDP (Barto et al., 1983). In the original paper, the authors find that both direct and indirect learning yields decision tree

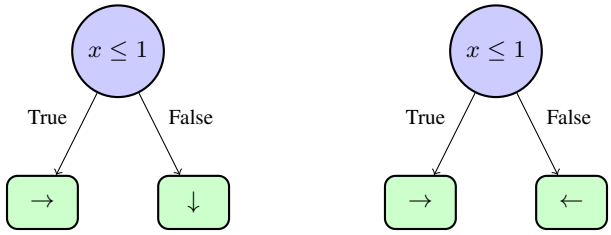



Figure 6: Left, an optimal depth-1 decision tree policy for the grid world MDP from figure 1. On the right, a sub-optimal depth-1 decision tree policy.

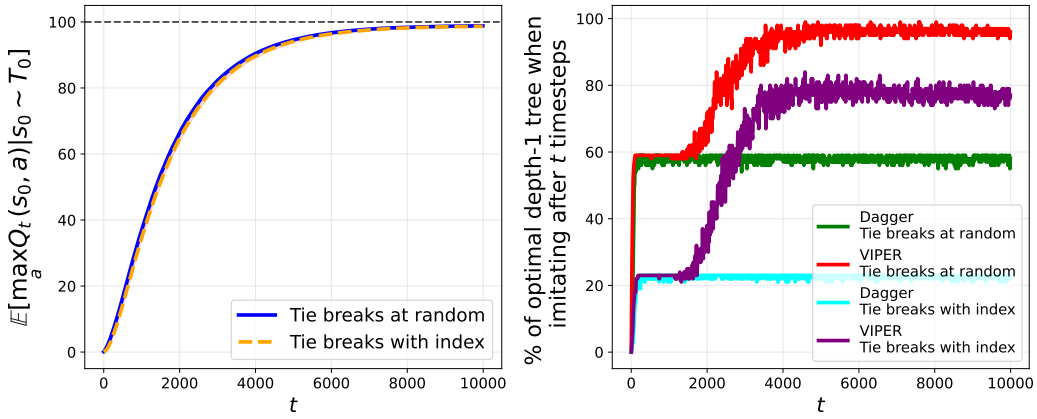



Figure 7: Left, sample complexity curve of Q-learning with default hyperparameters on the $2 \times 2$ grid world MDP over 100 random seeds. Right, performance of indirect interpretable methods when imitating the greedy policy with a tree at different Q-learning stages.

**Algorithm 2:** Extract a decision tree policy: algorithm 1 from Topin et al. (2021)

**Data:** Deterministic partially observable policy $\pi_{po}$ for IBMDP
$\mathcal{M}_{IB} = \langle S \times O, A \cup A_{info}, (R, \zeta), (T_{info}, T, T_0) \rangle$ and IBMDP observation
$\boldsymbol{o} = (L'_1, U'_1, \ldots, L'_p, U'_p)$
**Result:** Decision tree policy $\pi_{\mathcal{T}}$ for MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$
**Function** `Subtree_From_Policy(`$\boldsymbol{o}, \pi_{po}$`)`**:**
  $a \leftarrow \pi_{po}(\boldsymbol{o})$
  **if** *$a$ is a downstream action* **then**
    **return** Leaf_Node(action: $a$)
  **end**
  **else**
    $\langle i, v \rangle \leftarrow a$;  $\boldsymbol{o}_L \leftarrow \boldsymbol{o}$;  $\boldsymbol{o}_R \leftarrow \boldsymbol{o}$
    $\boldsymbol{o}_L \leftarrow (L'_1, U'_1, \ldots, L'_j, v, \ldots, L'_p, U'_p)$;  $\boldsymbol{o}_R \leftarrow (L'_1, U'_1, \ldots, v, U'_j, \ldots, L'_p, U'_p)$
    $child_L \leftarrow$ Subtree_From_Policy$(\boldsymbol{o}_L, \pi_{po})$; $child_R \leftarrow$ Subtree_From_Policy$(\boldsymbol{o}_R, \pi_{po})$
    **return** Internal_Node(feature: $i$, value: $v$, children: $(child_L, child_R)$)
  **end**

policies with similar RL objective values (definition 2) for the CartPole. On the other hand, we find that, imitation learning, despite not directly optimizing the RL objective for CartPole, outperforms deep RL which directly optimizes a trade-off of the standard RL objective and interpretability.

Authors of Topin et al. (2021) use two deep reinforcement learning baselines to which they apply some modifications in order to learn memoryless policies. Authors modify the standard DQN (Mnih et al., 2015) to return a memoryless policy. The trained $Q$-function is approximated with a neural network $O \to \mathbb{R}^{|A \cup A_{info}|}$ rather than $S \times O \to \mathbb{R}^{|A \cup A_{info}|}$. In this modified DQN, the temporal difference error target for the $Q$-function $O \to A \cup A_{info}$ is approximated by a neural network $S \times O \to A \cup A_{info}$ that is in turn trained by bootstrapping the temporal difference error with itself. We present the modifications in algorithm 3. Similar modifications are applied to the standard PPO (Schulman et al., 2017) that we present in the appendix (algorithm 4). In the modified PPO, neural network policy $O \to A \cup A_{info}$ is trained using a neural network value function $S \times O \to A \cup A_{info}$ as a critic.

Those two variants of DQN and PPO have first been introduced in (Pinto et al., 2017) for robotic tasks with partially observable components, under the name "asymmetric" actor-critic. Asymmetric RL algorithms that have policy and value estimates using different information from a POMDP (Sondik, 1978; Sigaud & Buffet, 2013) were later studied theoretically to solve POMDPs in Baisero's work (Baisero et al., 2022; Baisero & Amato, 2022). The connections from Deep RL in IBMDPs for objective is absent from (Topin et al., 2021) and we defer their connections to direct interpretable reinforcement learning to the next chapter as our primary goal is to reproduce (Topin et al., 2021) *as is*. Next, we present the precise experimental setup we use to reproduce (Topin et al., 2021) in order to study direct deep reinforcement learning of decision tree policies for the CartPole MDP.

### 8.1 IBMDP formulation

Given a base MDP $\mathcal{M} \equiv \langle S, A, R, T, T_0 \rangle$ (cf. definition 1), in order to define an IBMDP $\mathcal{M}_{IB}\langle S \times O, A \cup A_{info}, (R, \zeta), (T, T_0, T_{info}) \rangle$ (cf. definition 5), the user needs to provide the set of information gathering actions $A_{info}$ and the reward $\zeta$ for taking those. Authors of (Topin et al., 2021) propose to parametrize the set of IGAs with $j \times q$ actions $\langle j, v_k \rangle$ with $v_k$ depending on the current observation $\boldsymbol{o}_t = (L'_1, U'_1, \ldots, L'_j, U'_j, \ldots, L'_p, U'_p)$: $v_k = \frac{k(U'_j - L'_j)}{q+1}$. This parametric IGAs space keeps the discrete IBMDP action space at a reasonable size while providing a learning algorithm with varied IGAs to try.

For example, if we define an IBMDP with $q = 3$ for the grid world from figure 1, the grid world action space is augmented with six IGAs. At $t = 0$, recall that $\boldsymbol{o}_0 = (0, 2, 0, 2)$, so if an IGA is taken, e.g. $\langle 2, v_2 \rangle$, the effective IGA is $\langle j, v_2 = \frac{k(2-0)}{3+1} \rangle = \langle 1, 2 \rangle$ which in turn effectively corresponds to an internal decision tree node $y \leq 1$. If the current state $y$-feature value is $0.5$, then the next observation at $t = 1$ is $\boldsymbol{o}_1 = (0, 2, 0, 1)$. At $t = 2$ if $a_t = \langle 2, v_2 \rangle$ again, it would be effectively $\langle j, v_2 = \frac{k(1-0)}{3+1} \rangle = \langle 2, 0.5 \rangle$. This would give the next observation at $t = 2$ $\boldsymbol{o}_2 = (0, 2, 0, 0.5)$ and so on.

Furthermore, author propose to regularize the learned decision tree policy with a maximum depth parameter $D$. Unfortunately, the authors did not describe how they implemented the depth control in their work, hence we have to try different approaches to reproduce their results.

To control the tree depth during learning in the IBMDP, we can either give negative reward for taking $D$ IGAs in a row, or terminate the trajectory. In practice, we could also have a state-dependent action space such that taking an IGA is not allowed after taking $D$ IGAs in a row. The latter approach–sometimes called action masking–is not compatible with the definition of an MDP (cf. definition 1) in which all actions are available in all states. To apply the penalization approaches, one can extend the MDP states to keep track of the current tree depth. Similarly, the termination approach requires a transition function that depends on the current tree depth.

Table 1: IBMDP hyperparameters. We try 12 different IBMDPs. In green we highlight the hyperparameters from the original paper and in red we highlight the hyperparameter names for which author do not give information.

| **Hyperparameter** | **Values** |
|---|---|
| Discount factor $\gamma$ | 1 |
| Information gathering actions parameter $q$ | 2, 3 |
| Information gathering actions rewards $\zeta$ | -0.01, 0.01 |
| Depth control | Done signal, negative reward, none |

We actually find that when $q + 1$, the parameter that defines threshold values in decision tree policy nodes (cf. definition 5), is a prime number, then as a direct consequence of the *Chinese Remainder Theorem*[2], the current tree depth is directly encoded in the current observation $\boldsymbol{o}_t$. Hence, when $q + 1$ is prime, we can control the depth through either transitions or rewards without tracking the tree depth. We use the exact same downstream MDP and associated IBMDPs for our experiments as (Topin et al., 2021) except when mentioned otherwise.

**Downstream MDP** The task at hand is to optimize the RL objective (definition 2) with a decision tree policy for the CartPole MDP (Barto et al., 1983). At each time step a learning algorithm observes the cart's position and velocity and the pole's angle and angular velocity, and can take action to push the CartPole left or right. While the CartPole is roughly balanced, i.e., while the cart's angle remains in some fixed range, the agent gets a positive reward. If the CartPole is out of balance, the MDP transitions to an absorbing terminal state and gets 0 reward forever. Like in (Topin et al., 2021), we use the gymnasium `CartPole-v0` implementation (Towers et al., 2024) of the CartPole MDP in which trajectories are truncated after 200 timesteps making the maximum cumulative reward, i.e. the optimal value of the RL objective when $\gamma = 1$, to be 200. The state features of the CartPole MDP are in $[-2, 2] \times [-2, 2] \times [-0.14, 0.14] \times [-1.4, 1.4]$.

**IBMDP** Authors define the associated IBMDP (definition 5) with $\zeta = -0.01$ and 4 information gathering actions. In addition to the original IBMDP paper, we also try $\zeta = 0.01$ and 3 information gathering actions. We use the same discount factor as the authors: $\gamma = 1$. We try two different approaches to limit the depth of decision tree policies to be at most 2: terminating trajectories if the agent takes too many information gathering actions in a row or simply giving a reward of $-1$ to the agent every time it takes an information gathering action past the depth limit. In practice, we could have tried an action masking approach, i.e. having a state dependent-action set, but we want to abide to the MDP formalism in order to properly understand direct interpretable approaches. We will also try IBMDPs where we do not limit the maximum depth for completeness.

In tables 4 and 5 we report the top-5 hyperparameters for Modified RL baselines when learning partially observable IBMDP policies in terms of extracted decision tree policies performances in the CartPole MDP.

### 8.2 Experimental setup

**Modified DQN and Modified PPO** as mentioned above, the authors use the modified version of DQN from algorithm 3. We use the exact same hyperparameters for modified DQN as the authors when possible. We use the same layers width (128) and number of hidden layers (2), the same exploration strategy ($\epsilon$-greedy with linearly decreasing value $\epsilon$ between 0.5 and 0.05 during the first 10% of the training), the same replay buffer size ($10^6$) and the same number of transitions to be collected randomly before doing value updates ($10^5$). We also try to use more exploration during training (change the initial $\epsilon$ value to 0.9). We use the same optimizer (RMSprop with hyperparameter 0.95 and learning rate $2.5 \times 10^{-4}$) to update the $Q$-networks. Authors did not share

[2] https://en.wikipedia.org/wiki/Chinese_remainder_theorem

Table 2: (Modified) DQN trained on $10^6$ timesteps. This gives four different instantiation of (modified) DQN. Hyperparameters not mentioned are stable-baselines3 default. In green we highlight the hyperparameters from the original paper and in red we highlight the hyperparameter names for which author do not give information.

| **Hyperparameter** | **Values** |
|---|---|
| Buffer size | $10^6$ |
| Random transitions before learning | $10^5$ |
| Epsilon start | 0.9, 0.5 |
| Epsilon end | 0.05 |
| Exploration fraction | 0.1 |
| Optimizer | RMSprop ($\alpha = 0.95$) |
| Learning rate | $2.5 \times 10^{-4}$ |
| Networks architectures | [128, 128] |
| Networks activations | tanh(), relu() |

Table 3: (Modified) PPO trained on $4 \times 10^6$ timesteps. This gives two different instantiation of (modified) PPO. Hyperparameters not mentioned are stable-baselines3 default. In green we highlight the hyperparameters from the original paper and in red we highlight the hyperparameter names for which author do not give information.

| **Hyperparameter** | **Values** |
|---|---|
| Steps between each policy gradient steps | 512 |
| Number of minibatch for policy gradient updates | 4 |
| Networks architectures | [64, 64] |
| Networks activations | tanh(), relu() |

Table 4: Top 5 hyperparameter configurations for modified DQN + IBMDP, bold font represent the original paper hyperparameters.

| Rank | $q$ | Depth control | Activation | Exploration | $\zeta$ | Mean Final Performance |
|---|---|---|---|---|---|---|
| 1 | 3 | termination | tanh | 0.9 | 0.01 | 53 |
| 2 | 2 | termination | tanh | 0.5 | -0.01 | 24 |
| **3** | **3** | **termination** | tanh | **0.5** | **-0.01** | **24** |
| 4 | 2 | termination | tanh | 0.5 | 0.01 | 23 |
| 5 | 2 | termination | tanh | 0.9 | -0.01 | 22 |

Table 5: Top 5 hyperparameter configurations for modified PPO + IBMDP, bold font represent the original paper hyperparameters.

| Rank | $q$ | Depth Control | Activation | $\zeta$ | Mean Final Performance |
|---|---|---|---|---|---|
| 1 | 3 | reward | relu | 0.01 | 139 |
| 2 | 3 | termination | relu | 0.01 | 132 |
| **3** | **3** | **reward** | tanh | **-0.01** | **119** |
| 4 | 3 | reward | relu | -0.01 | 117 |
| 5 | 3 | reward | tanh | 0.01 | 116 |

**Algorithm 3:** Modified Deep Q-Network. We highlight in green the changes to the standard DQN (Mnih et al., 2015).

**Data:** IBMDP $\mathcal{M}_{IB}\langle S \times O, A \cup A_{info}, (R, \zeta), (T, T_0, T_{info})\rangle$, learning rate $\alpha$, exploration rate $\epsilon$, partially observable Q-network parameters $\theta$, Q-network parameters $\phi$, replay buffer $\mathcal{B}$, update frequency $C$
**Result:** deterministic memoryless policy $\pi_{po}$
Initialize partially observable Q-network parameters $\theta$
Initialize Q-network parameters $\phi$ and target network parameters $\phi^- = \phi$
Initialize replay buffer $\mathcal{B} = \emptyset$
**for** *each episode* **do**
  Initialize downstream state features $\boldsymbol{s}_0 \sim T_0$
  Initialize observation $\boldsymbol{o}_0 = (L_1, U_1, \ldots, L_p, U_p)$
  **for** *each step* $t$ **do**
    Choose action $a_t$ using $\epsilon$-greedy: $a_t = \text{argmax}_a\, Q_\theta(\boldsymbol{o}_t, a)$ with prob. $1 - \epsilon$
    Take action $a_t$, observe $r_t$
    Store transition $(\boldsymbol{s}_t, \boldsymbol{o}_t, a_t, r_t, \boldsymbol{s}_{t+1})$ in $\mathcal{B}$
    Sample random batch $(\boldsymbol{s}_i, \boldsymbol{o}_i, a_i, r_i, \boldsymbol{s}_{i+1}) \sim \mathcal{B}$
    $a' = \underset{a}{\text{argmax}} Q_\theta(\boldsymbol{o}_i, a)$
    $y_i = r_i + \gamma Q_{\phi^-}(\boldsymbol{s}_{i+1}, a')$ // Compute target
    $\phi \leftarrow \phi - \alpha \nabla_\phi (Q_\phi(\boldsymbol{s}_i, a_i) - y_i)^2$ // Update Q-network
    $\theta \leftarrow \theta - \alpha \nabla_\theta (Q_\theta(\boldsymbol{o}_i, a_i) - y_i)^2$ // Update partially observable Q-network
    **if** $t \bmod C = 0$ **then**
      $\theta^- \leftarrow \theta$ // Update target network
    **end**
    $\boldsymbol{s}_t \leftarrow \boldsymbol{s}_{t+1}$
    $\boldsymbol{o}_t \leftarrow \boldsymbol{o}_{t+1}$
  **end**
**end**
$\pi_{po}(\boldsymbol{o}) = \text{argmax}_a\, Q_\theta(\boldsymbol{o}, a)$ // Extract greedy policy

**Algorithm 4:** Modified Proximal Policy Optimization

**Data:** IBMDP $\mathcal{M}_{IB}\langle S \times O, A \cup A_{info}, (R, \zeta), (T, T_0, T_{info})\rangle$, learning rate $\alpha$, policy parameters $\theta$, clipping parameter $\epsilon$, value function parameters $\phi$
**Result:** Memoryless stochastic policy $\pi_{po_\theta}$
Initialize policy parameters $\theta$ and value function parameters $\phi$
**for** *each episode* **do**
  Generate trajectory $\tau = (\boldsymbol{s}_0, \boldsymbol{o}_0, a_0, r_0, \boldsymbol{s}_1, \boldsymbol{o}_1, a_1, r_1, \ldots)$ following $\pi_\theta$
  **for** *each timestep* $t$ *in trajectory* **do**
    $G_t \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r_k$ // Compute return
    $A_t \leftarrow G_t - V_\phi(\boldsymbol{s}_t)$ // Compute advantage
    $r_t(\theta) \leftarrow \frac{\pi_{po_\theta}(a_t|\boldsymbol{o}_t)}{\pi_{po_{\theta\,old}}(a_t|\boldsymbol{o}_t)}$ // Compute probability ratio
    $L_t^{CLIP} \leftarrow \min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)$ // Clipped objective
    $\theta \leftarrow \theta + \alpha \nabla_\theta L_t^{CLIP}$ // Policy update
    $\phi \leftarrow \phi + \alpha \nabla_\phi (G_t - V_\phi(\boldsymbol{s}_t))^2$ // Value function update
  **end**
  $\theta_{old} \leftarrow \theta$ // Update old policy
**end**

which DQN implementation they used so we use the stable-baselines3 one (Raffin et al., 2021)[3]. Authors did not share which activation functions they used so we try both $\tanh$ and $\mathrm{relu}$. For the modified PPO algorithm (algorithm 4), we can exactly match the authors hyperparameters since they use the open source stable-baselines3 implementation of PPO. We match training budgets: we train modified DQN on 1 million timesteps and modified PPO on 4 million timesteps.

**DQN and PPO** We also benchmark the standard DQN and PPO when learning standard Markovian IBMDP policies $\pi : S \times O \to A \cup A_{info}$ and when learning standard $\pi : S \to A$ policies directly in the CartPole MDP. We summarize hyperparameters for the IBMDP and for the learning algorithms in appendices 1, 2 and 3.

**Indirect methods** We also compare modified RL algorithm to imitation learning. To do so, we use VIPER or Dagger to imitate greedy neural network policies obtained with standard DQN learning directly on CartPole. We use Dagger to imitate neural network policies obtained with the standard PPO learning directly on CartPole. For each indirect method, we imitate the neural network experts by fitting decision trees on 10000 expert transitions using the CART (Breiman et al., 1984) implementation from scikit-learn (Pedregosa et al., 2011) with default hyperparameters and maximum depth of 2 like in (Topin et al., 2021).

### 8.2.1 Metrics

The key metric of this section is performance when controlling the CartPole, i.e, the average *undiscounted* cumulative reward of a policy on 100 trajectories (RL objective with $\gamma = 1$). For modified RL algorithms that learn a memoryless policy (or $Q$-function) in an IBMDP, we periodically extract the policy (or $Q$-function) and use algorithm 2 to extract a decision tree for the CartPole MDP. We then evaluate the tree on 100 independent trajectories in the MDP and report the mean undiscounted cumulative reward. For RL applied to IBMDPs, since we can't deploy learned policies directly to the downstream MDP as the state dimensions mismatch–such policies are $S \times O \to A \cup A_{info}$ but the MDP states are in $S$–we periodically evaluate those IBMDP policies in a copy of the IBMDP in which we fix $\zeta = 0$ ensuring that the copied IBMDP undiscounted cumulative rewards only account rewards from the CartPole MDP (non-zero rewards in the IBMDP only occur when a reward from the downstream MDP is given, i.e. when $a_t \in A$ in the IBMDP (definition 5)). Similarly, we do 100 trajectories of the extracted policies in the copied IBMDP and report the average undiscounted cumulative reward. For RL applied directly to the downstream MDP we can just periodically extract the learned policies and evaluate them on 100 CartPole trajectories.

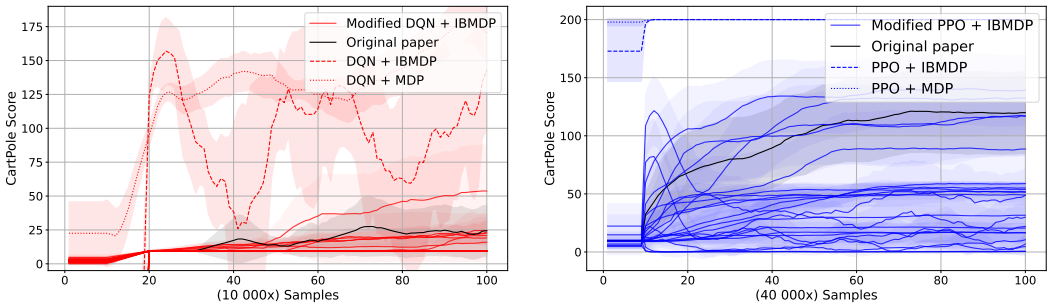
Since imitation learning baselines train offline, i.e, on a fixed dataset, their performances cannot directly be reported on the same axis as RL baselines. For that reason, during the training of a standard RL baseline, we periodically extract the trained neural policy/$Q$-function that we consider as the expert to imitate. Those experts are then imitated with VIPER or Dagger using 10 000 newly generated transitions and then fitsted decision tree policies are then evaluated on 100 CartPole trajectories. We do not report the imitation learning objective values during VIPER or Dagger training. Every single combination of IBMDP and Modified RL hyperparameters is run 20 times. For standard RL on either an IBMDP or an MDP, we use the paper original hyperparameters when they were specified, with depth control using negative rewards, $\tanh()$ activations. We use 20 individual random seeds for every experiment in this chapter. Next, we present our results when reproducing (Topin et al., 2021).

## 8.3 Results

### 8.3.1 How well do modified deep RL baselines learn in IBMDPs?

On figure 8a, we observe that modified DQN can learn in IBMDPs–the curves have an increasing trend–but we also observe that modified DQN finds poor decision tree policies for the CartPole MDP

[3] We are cleaning our source code and will open source it as soon as possible




(a) DQN variants

(b) PPO variants

Figure 8: Comparison of modified reinforcement learning algorithms on different CartPole IBMDPs. (a) Shows variations of modified DQN and DQN (table 2), while (b) shows variations of modified PPO and PPO (table 3). For both algorithms, we give different line-styles for the learning curves when applied directly on the CartPole MDP versus when applied on the IBMDP to learn standard Markovian policies. We color the modified RL algorithm variant from the original paper in black. Shaded areas represent the confidence interval at 95% at each measure on the y-axis.

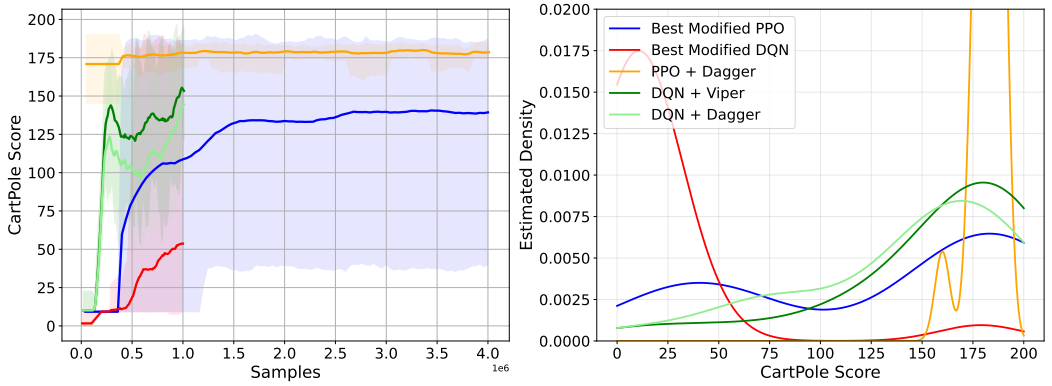



Figure 9: (left) Mean performance of the best–w.r.t. the RL objective for CartPole–modified RL + IBMDP combination. Shaded areas represent the min and max performance over the 20 seeds during training. (right) Corresponding score distribution of the final decision tree policies w.r.t. the RL objective for CartPole.

in average–the curves flatten at the end of the x-axis and have low y-values. In particular, the highest final y-value, among all the learning curves that could possibly correspond to the original paper modified DQN, correspond to poor performances on the CartPole MDP. On figure 8b, we observe that modified PPO finds decision tree policies with almost 150 cumulative rewards towards the end of training. The performance difference with modified DQN could be because we trained modified PPO longer, like in the original paper. However it could also be because DQN-like algorithms with those hyperparameters struggle to learn in CartPole (IB)MDPs. Indeed, we notice that for DQN-like baselines, learning seems difficult in general independently of the setting. On figures 8a and 8b, we observe that standard RL baselines (RL + IBMDP and RL + MDP), learn better CartPole policies in average than their modified counterparts that learn memoryless policies. On figure 8b, it is clear that for the standard PPO baselines, learning is super efficient and algorithms learn optimal policies with reward 200 in few thousands steps.

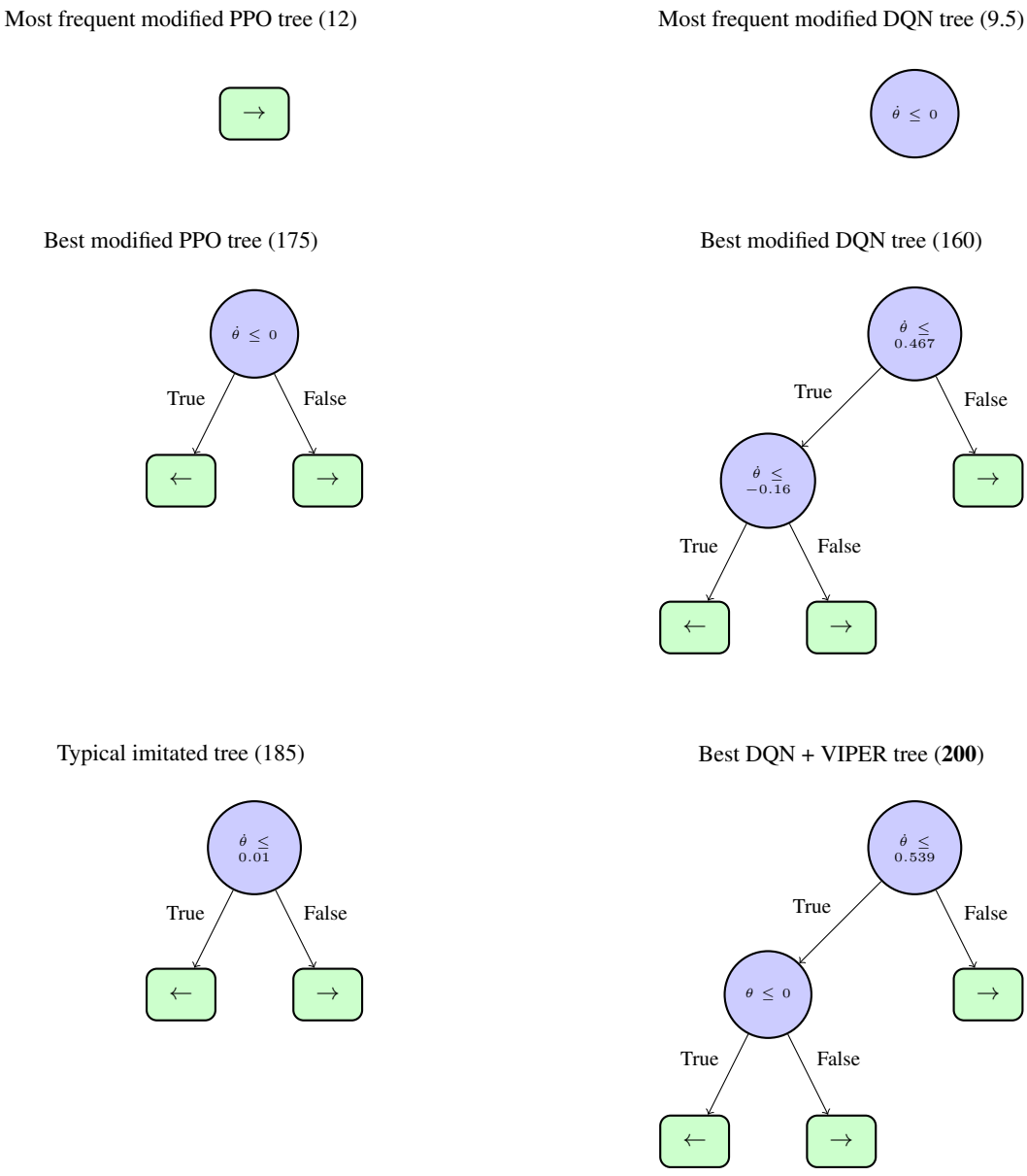



Figure 10: Trees obtained by modified deep RL in IBMDPs against trees obtained with imitation (RL objective value). $\theta$ and $\dot{\theta}$ are respectively the angle and the angular velocity of the pole

### 8.3.2 Which decision tree policies does direct reinforcement learning return for the CartPole MDP?

On figure 9, we isolate the best performing algorithms instantiations that learn decision tree policies for the CartPole MDP. We compare the best modified DQN and modified PPO to imitation learning baselines that use the surrogate imitation objective to find CartPole decision tree policies. We find that despite having poor performances in *average*, the modified deep reinforcement learning baselines can find very good decision tree policies as shown by the min-max shaded areas on the left of figure 9 and the corresponding estimated density of learned trees performances. However this is not desirable, a user typically wants an algorithm that can consistently find good decision tree policies. As shown by the estimated densities, indirect methods consistently find good decision tree policies (the higher modes of distributions are on the right of the plot). On the other hand, the decision tree policies returned by direct RL methods seem equally distributed on both extremes of the scores.

On figure 10, we present the best decision tree policies for CartPole returned by modified DQN and modified PPO. We used algorithm 2 to extract 20 trees from the 20 memoryless policies returned by the modified deep reinforcement learning algorithms over the 20 training seeds. We then plot the best tree for each baseline. Those trees get an average RL objective of roughly 175. Similarly, we plot a representative tree for imitation learning baseline as well as a tree that is optimal for CartPole w.r.t. the RL objective obtained with VIPER. Unlike for direct methods, the trees returned by imitation learning are extremely similar across seeds. In particular they often only vary in the scalar value used in the root node but in general have the same structure and test the angular velocity. On the other hand the most frequent trees across seeds returned by modified RL baselines are "trivial" decision tree policies that either repeat the same downstream action forever or repeat the same IGA (definition 5) forever.

## 9 RL objective values calculations

**Optimal depth-1 decision tree policy** $\pi_{\mathcal{T}_1}$ has one root node that tests $x \leq 1$ (respectively $y \leq 1$) and two leaf nodes $\rightarrow$ and $\downarrow$. To compute $V^{\pi}_{\mathcal{T}_1}(\boldsymbol{o}_0)$, we compute the values of $\pi_{\mathcal{T}_1}$ in each of the possible starting states $(\boldsymbol{s}_0, \boldsymbol{o}_0), (\boldsymbol{s}_1, \boldsymbol{o}_0), (\boldsymbol{s}_2, \boldsymbol{o}_0), (\boldsymbol{s}_g, \boldsymbol{o}_0)$ and compute the expectation over those. At initialization, when the downstream state is $\boldsymbol{s}_g = (1.5, 0.5)$, the depth-1 decision tree policy cycles between taking an information gathering action $x \leq 1$ and moving down to get a positive reward for which it gets the returns:

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_g, \boldsymbol{o}_0) &= \zeta + \gamma + \gamma^2\zeta + \gamma^3 \dots \\ &= \sum_{t=0}^{\infty} \gamma^{2t}\zeta + \sum_{t=0}^{\infty} \gamma^{2t+1} \\ &= \frac{\zeta + \gamma}{1 - \gamma^2} \end{aligned}$$

At initialization, in either of the downstream states $\boldsymbol{s}_0 = (0.5, 0.5)$ and $\boldsymbol{s}_2 = (1.5, 1.5)$, the value of the depth-1 decision tree policy is the return when taking one information gathering action $x \leq 1$, then moving right or down, then following the policy from the goal state $\boldsymbol{s}_g$:

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_0, o_0) &= \zeta + \gamma 0 + \gamma^2 V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_g, o_0) \\ &= \zeta + \gamma^2 V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_g, o_0) \\ &= V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_2, o_0) \end{aligned}$$

Similarly, the value of the best depth-1 decision tree policy in state $\boldsymbol{s}_1 = (0.5, 1.5)$ is the value of taking one information gathering action then moving right to $\boldsymbol{s}_2$ then following the policy in $\boldsymbol{s}_2$:

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_1, \boldsymbol{o}_0) &= \zeta + \gamma 0 + \gamma^2 V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_2, \boldsymbol{o}_0) \\ &= \zeta + \gamma^2 V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_2, o_0) \\ &= \zeta + \gamma^2(\zeta + \gamma^2 V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_g, o_0)) \\ &= \zeta + \gamma^2\zeta + \gamma^4 V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_g, o_0) \end{aligned}$$

Since the probability of being in any downstream states at initialization given that the observation is $\boldsymbol{o}_0$ is simply the probability of being in any downstream states at initialization, we can write:

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(\boldsymbol{o}_0) &= \frac{1}{4}V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_g, \boldsymbol{o}_0) + \frac{2}{4}V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_2, \boldsymbol{o}_0) + \frac{1}{4}V^{\pi_{\mathcal{T}_1}}(\boldsymbol{s}_1, \boldsymbol{o}_0) \\ &= \frac{1}{4}\frac{\zeta+\gamma}{1-\gamma^2} + \frac{2}{4}(\zeta + \gamma^2\frac{\zeta+\gamma}{1-\gamma^2}) + \frac{1}{4}(\zeta + \gamma^2\zeta + \gamma^4\frac{\zeta+\gamma}{1-\gamma^2}) \\ &= \frac{1}{4}\frac{\zeta+\gamma}{1-\gamma^2} + \frac{2}{4}(\frac{\zeta+\gamma^3}{1-\gamma^2}) + \frac{1}{4}(\frac{\zeta+\gamma^5}{1-\gamma^2}) \\ &= \frac{4\zeta+\gamma+2\gamma^3+\gamma^5}{4(1-\gamma^2)} \end{aligned}$$

**Depth-0 decision tree:** has only one leaf node that takes a single downstream action indefinitely. For this type of tree the best reward achievable is to take actions that maximize the probability of reaching the objective $\rightarrow$ or $\downarrow$. In that case the objective value of such tree is: In the goal state $G = (1, 0)$, the value of the depth-0 tree $\mathcal{T}_0$ is:

$$\begin{aligned} V_G^{\mathcal{T}_0} &= 1 + \gamma + \gamma^2 + \dots \\ &= \sum_{t=0}^{\infty}\gamma^t \\ &= \frac{1}{1-\gamma} \end{aligned}$$

In the state $(0, 0)$ when the policy repeats going right respectively in the state $(0, 1)$ when the policy repeats going down, the value is:

$$\begin{aligned} V_{S_0}^{\mathcal{T}_0} &= 0 + \gamma V_g^{\mathcal{T}_0} \\ &= \gamma V_G^{\mathcal{T}_0} \end{aligned}$$

In the other states the policy never gets positive rewards; $V_{S_1}^{\mathcal{T}_0} = V_{S_2}^{\mathcal{T}_0} = 0$. Hence:

$$\begin{aligned} J(\mathcal{T}_0) &= \frac{1}{4}V_G^{\mathcal{T}_0} + \frac{1}{4}V_{S_0}^{\mathcal{T}_0} + \frac{1}{4}V_{S_1}^{\mathcal{T}_0} + \frac{1}{4}V_{S_2}^{\mathcal{T}_0} \\ &= \frac{1}{4}V_G^{\mathcal{T}_0} + \frac{1}{4}\gamma V_G^{\mathcal{T}_0} + 0 + 0 \\ &= \frac{1}{4}\frac{1}{1-\gamma} + \frac{1}{4}\gamma\frac{1}{1-\gamma} \\ &= \frac{1+\gamma}{4(1-\gamma)} \end{aligned}$$

**Unbalanced depth-2 decision tree:** the unbalanced depth-2 decision tree takes an information gathering action $x \leq 0.5$ then either takes the $\downarrow$ action or takes a second information $y \leq 0.5$ followed by $\rightarrow$ or $\downarrow$. In states $G$ and $S_2$, the value of the unbalanced tree is the same as for the

depth-1 tree. In states $S_0$ and $S_1$, the policy takes two information gathering actions before taking a downstream action and so on:

$$V_{S_0}^{\mathcal{T}_u} = \zeta + \gamma\zeta + \gamma^2 0 + \gamma^3 V_G^{\mathcal{T}_1}$$

$$\begin{aligned} V_{S_1}^{\mathcal{T}_u} &= \zeta + \gamma\zeta + \gamma^2 0 + \gamma^3 V_{S_0}^{\mathcal{T}_u} \\ &= \zeta + \gamma\zeta + \gamma^2 0 + \gamma^3(\zeta + \gamma\zeta + \gamma^2 0 + \gamma^3 V_G^{\mathcal{T}_1}) \\ &= \zeta + \gamma\zeta + \gamma^3\zeta + \gamma^4\zeta + \gamma^6 V_G^{\mathcal{T}_1} \end{aligned}$$

We get:

$$\begin{aligned} J(\mathcal{T}_u) &= \frac{1}{4}V_G^{\mathcal{T}_u} + \frac{1}{4}V_{S_0}^{\mathcal{T}_u} + \frac{1}{4}V_{S_1}^{\mathcal{T}_u} + \frac{1}{4}V_{S_2}^{\mathcal{T}_u} \\ &= \frac{1}{4}V_G^{\mathcal{T}_1} + \frac{1}{4}(\zeta + \gamma\zeta + \gamma^3 V_G^{\mathcal{T}_1}) + \frac{1}{4}(\zeta + \gamma\zeta + \gamma^3\zeta + \gamma^4\zeta + \gamma^6 V_G^{\mathcal{T}_1}) + \frac{1}{4}V_{S_2}^{\mathcal{T}_1} \\ &= \frac{1}{4}(\frac{\zeta+\gamma}{1-\gamma^2}) + \frac{1}{4}(\frac{\gamma\zeta+\gamma^4+\zeta-\gamma^2\zeta}{1-\gamma^2}) + \frac{1}{4}(\zeta + \gamma\zeta + \gamma^3\zeta + \gamma^4\zeta + \gamma^6 V_G^{\mathcal{T}_1}) + \frac{1}{4}V_{S_2}^{\mathcal{T}_1} \\ &= \frac{1}{4}(\frac{\zeta+\gamma}{1-\gamma^2}) + \frac{1}{4}(\frac{\gamma\zeta+\gamma^4+\zeta-\gamma^2\zeta}{1-\gamma^2}) + \frac{1}{4}(\frac{\zeta+\gamma\zeta-\gamma^2\zeta-\gamma^5\zeta+\gamma^6\zeta+\gamma^7}{1-\gamma^2}) + \frac{1}{4}V_{S_2}^{\mathcal{T}_1} \\ &= \frac{1}{4}(\frac{\zeta+\gamma}{1-\gamma^2}) + \frac{1}{4}(\frac{\gamma\zeta+\gamma^4+\zeta-\gamma^2\zeta}{1-\gamma^2}) + \frac{1}{4}(\frac{\zeta+\gamma\zeta-\gamma^2\zeta-\gamma^5\zeta+\gamma^6\zeta+\gamma^7}{1-\gamma^2}) + \frac{1}{4}(\frac{\zeta+\gamma^3}{1-\gamma^2}) \\ &= \frac{\zeta(4+2\gamma-2\gamma^2-\gamma^5+\gamma^6)+\gamma+\gamma^3+\gamma^4+\gamma^7}{4(1-\gamma^2)} \end{aligned}$$

**The balanced depth-2 decision tree:** alternates in every state between taking the two available information gathering actions and then a downstream action. The value of the policy in the goal state is:

$$\begin{aligned} V_G^{\mathcal{T}_2} &= \zeta + \gamma\zeta + \gamma^2 + \gamma^3\zeta + \gamma^4\zeta + \dots \\ &= \sum_{t=0}^{\infty}\gamma^{3t}\zeta + \sum_{t=0}^{\infty}\gamma^{3t+1}\zeta + \sum_{t=0}^{\infty}\gamma^{3t+2} \\ &= \frac{\zeta}{1-\gamma^3} + \frac{\gamma\zeta}{1-\gamma^3} + \frac{\gamma^2}{1-\gamma^3} \end{aligned}$$

Following the same reasoning for other states we find the objective value for the depth-2 decision tree policy to be:

$$\begin{aligned} J(\mathcal{T}_2) &= \frac{1}{4}V_G^{\mathcal{T}_2} + \frac{2}{4}V_{S_2}^{\mathcal{T}_2} + \frac{1}{4}V_{S_1}^{\mathcal{T}_2} \\ &= \frac{1}{4}V_G^{\mathcal{T}_2} + \frac{2}{4}(\zeta + \gamma\zeta + \gamma^2 0 + \gamma^3 V_G^{\mathcal{T}_2}) + \frac{1}{4}(\zeta + \gamma\zeta + \gamma^2 0 + \gamma^3\zeta + \gamma^4\zeta + \gamma^5 0 + \gamma^6 V_G^{\mathcal{T}_2}) \\ &= \frac{\zeta(3+3\gamma)+\gamma^2+\gamma^5+\gamma^8}{4(1-\gamma^3)} \end{aligned}$$

**Infinite tree:** we also consider the infinite tree policy that repeats an information gathering action forever and has objective: $J(\mathcal{T}_{\text{inf}}) = \frac{\zeta}{1-\gamma}$

**Stochastic policy:** the other non-trivial policy that can be learned by solving a partially observable IBMDP is the stochastic policy that guarantees to reach $G$ after some time: fifty percent chance to

do $\rightarrow$ and fifty percent chance to do $\downarrow$. This stochastic policy has objective value:

$$
\begin{aligned}
V_G^{\text{stoch}} &= \frac{1}{1-\gamma} \\
V_{S_0}^{\text{stoch}} &= 0 + \frac{1}{2}\gamma V_G^{\text{stoch}} + \frac{1}{2}\gamma V_{S_1}^{\text{stoch}} \\
V_{S_2}^{\text{stoch}} &= 0 + \frac{1}{2}\gamma V_G^{\text{stoch}} + \frac{1}{2}\gamma V_{S_1}^{\text{stoch}} = V_{S_0}^{\text{stoch}} \\
V_{S_1}^{\text{stoch}} &= 0 + \frac{1}{2}\gamma V_{S_2}^{\text{stoch}} + \frac{1}{2}\gamma V_G^{\text{stoch}} = \frac{1}{2}\gamma V_{S_0}^{\text{stoch}} + \frac{1}{2}\gamma V_G^{\text{stoch}}
\end{aligned}
$$

Solving these equations:

$$
\begin{aligned}
V_{S_1}^{\text{stoch}} &= \frac{1}{2}\gamma V_{S_0}^{\text{stoch}} + \frac{1}{2}\gamma V_G^{\text{stoch}} \\
&= \frac{1}{2}\gamma(\frac{1}{2}\gamma V_G^{\text{stoch}} + \frac{1}{2}\gamma V_{S_1}^{\text{stoch}}) + \frac{1}{2}\gamma V_G^{\text{stoch}} \\
&= \frac{1}{4}\gamma^2 V_G^{\text{stoch}} + \frac{1}{4}\gamma^2 V_{S_1}^{\text{stoch}} + \frac{1}{2}\gamma V_G^{\text{stoch}} \\
V_{S_1}^{\text{stoch}} - \frac{1}{4}\gamma^2 V_{S_1}^{\text{stoch}} &= \frac{1}{4}\gamma^2 V_G^{\text{stoch}} + \frac{1}{2}\gamma V_G^{\text{stoch}} \\
V_{S_1}^{\text{stoch}}(1 - \frac{1}{4}\gamma^2) &= (\frac{1}{4}\gamma^2 + \frac{1}{2}\gamma) V_G^{\text{stoch}} \\
V_{S_1}^{\text{stoch}} &= \frac{\frac{1}{4}\gamma^2 + \frac{1}{2}\gamma}{1 - \frac{1}{4}\gamma^2} V_G^{\text{stoch}} \\
&= \frac{\gamma(\frac{1}{4}\gamma + \frac{1}{2})}{1 - \frac{1}{4}\gamma^2} \cdot \frac{1}{1-\gamma} \\
&= \frac{\gamma(\frac{1}{4}\gamma + \frac{1}{2})}{(1 - \frac{1}{4}\gamma^2)(1-\gamma)}
\end{aligned}
$$

$$
\begin{aligned}
V_{S_0}^{\text{stoch}} &= \frac{1}{2}\gamma V_G^{\text{stoch}} + \frac{1}{2}\gamma V_{S_1}^{\text{stoch}} \\
&= \frac{1}{2}\gamma \cdot \frac{1}{1-\gamma} + \frac{1}{2}\gamma \cdot \frac{\gamma(\frac{1}{4}\gamma + \frac{1}{2})}{(1 - \frac{1}{4}\gamma^2)(1-\gamma)} \\
&= \frac{\frac{1}{2}\gamma}{1-\gamma} + \frac{\frac{1}{2}\gamma^2(\frac{1}{4}\gamma + \frac{1}{2})}{(1 - \frac{1}{4}\gamma^2)(1-\gamma)} \\
&= \frac{\frac{1}{2}\gamma(1 - \frac{1}{4}\gamma^2) + \frac{1}{2}\gamma^2(\frac{1}{4}\gamma + \frac{1}{2})}{(1 - \frac{1}{4}\gamma^2)(1-\gamma)} \\
&= \frac{\frac{1}{2}\gamma - \frac{1}{8}\gamma^3 + \frac{1}{8}\gamma^3 + \frac{1}{4}\gamma^2}{(1 - \frac{1}{4}\gamma^2)(1-\gamma)} \\
&= \frac{\frac{1}{2}\gamma + \frac{1}{4}\gamma^2}{(1 - \frac{1}{4}\gamma^2)(1-\gamma)} \\
&= \frac{\gamma(\frac{1}{2} + \frac{1}{4}\gamma)}{(1 - \frac{1}{4}\gamma^2)(1-\gamma)}
\end{aligned}
$$

$$
\begin{aligned}
J(\mathcal{T}_{\text{stoch}}) &= \frac{1}{4}(V_G^{\text{stoch}} + V_{S_0}^{\text{stoch}} + V_{S_1}^{\text{stoch}} + V_{S_2}^{\text{stoch}}) \\
&= \frac{1}{4}\left(\frac{1}{1-\gamma} + 2 \cdot \frac{\gamma(\frac{1}{2} + \frac{1}{4}\gamma)}{(1-\frac{1}{4}\gamma^2)(1-\gamma)} + \frac{\gamma(\frac{1}{4}\gamma + \frac{1}{2})}{(1-\frac{1}{4}\gamma^2)(1-\gamma)}\right) \\
&= \frac{1}{4}\left(\frac{1}{1-\gamma} + \frac{2\gamma(\frac{1}{2} + \frac{1}{4}\gamma) + \gamma(\frac{1}{4}\gamma + \frac{1}{2})}{(1-\frac{1}{4}\gamma^2)(1-\gamma)}\right) \\
&= \frac{1}{4}\left(\frac{1}{1-\gamma} + \frac{\gamma + \frac{1}{2}\gamma^2 + \frac{1}{4}\gamma^2 + \frac{1}{2}\gamma}{(1-\frac{1}{4}\gamma^2)(1-\gamma)}\right) \\
&= \frac{1}{4}\left(\frac{1}{1-\gamma} + \frac{\frac{3}{2}\gamma + \frac{3}{4}\gamma^2}{(1-\frac{1}{4}\gamma^2)(1-\gamma)}\right) \\
&= \frac{1}{4}\left(\frac{1 - \frac{1}{4}\gamma^2 + \frac{3}{2}\gamma + \frac{3}{4}\gamma^2}{(1-\frac{1}{4}\gamma^2)(1-\gamma)}\right) \\
&= \frac{1}{4}\left(\frac{1 + \frac{3}{2}\gamma + \frac{1}{2}\gamma^2}{(1-\frac{1}{4}\gamma^2)(1-\gamma)}\right) \\
&= \frac{1 + \frac{3}{2}\gamma + \frac{1}{2}\gamma^2}{4(1-\frac{1}{4}\gamma^2)(1-\gamma)}
\end{aligned}
$$

## 10 Training curves

We plot the sub-optimality gaps during training, w.r.t. the RL objective (definition 2), between the learned memoryless policy and the optimal deterministic memoryless policy: $|\mathbb{E}\left[V^{\pi^\star}(\boldsymbol{s}_0, \boldsymbol{o}_0)|\boldsymbol{s}_0 \sim T_0\right] - \mathbb{E}\left[V^{\pi}(\boldsymbol{s}_0, \boldsymbol{o}_0)|\boldsymbol{s}_0 \sim T_0\right]|$. Because we know the whole POIBMDP model that we can represent exactly as tables, and because we know for each $\zeta$ the RL objective value of the optimal deterministic memoryless policy (figure 3), we can report the *exact* sub-optimality gaps. In figure 11, we plot the sub-optimality gaps–averaged over 100 seeds–of learned policies during training. We do so for 200 different POIBMDPs where we change the reward for information gathering actions: we sample 200 $\zeta$ values uniformly in $[-1, 2]$. In figure 11, a different color represents a different POIBMDP.

Recall from figure 3 that for: (i) $\zeta \in [-1, 0]$, the optimal deterministic memoryless policy is a depth-0 tree, (ii) $\zeta \in ]0, 1[$, the optimal deterministic memoryless policy is a depth-1 tree, and (iii) $\zeta \in [1, 2]$, the optimal deterministic memoryless policy is a "infinite" tree that contains infinite number of internal nodes. We observe that, despite all sub-optimality gaps converging independently of the $\zeta$ values, not all algorithms in all POIBMDPs fully minimize the sub-optimality gap. In particular, all algorithms seem to consistently minimize the gap, i.e. learn the optimal policy or Q-function, only for $\zeta \in [1, 2]$ (all the yellow lines go to 0). However, we are interested in the range $\zeta \in ]0, 1[$ where the optimal decision tree policy is non-trivial, i.e. not taking the same action forever. In that range, no baseline consistently minimizes the sub-optimality gap.

## 11 Tabular RL algorithmic details for POIBMDPs

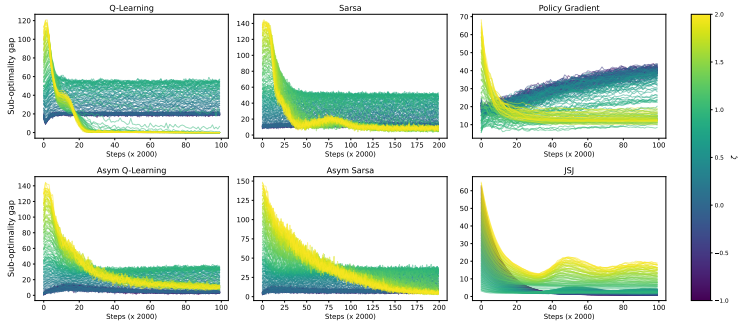



Figure 11: (Asymmetric) reinforcement learning in POIBMDPs. In each subplot, each single line is colored by the value of $\zeta$ in the corresponding POIBMDP in which learning occurs. Each single learning curve represent the sub-optimality gap averaged over 100 seeds.

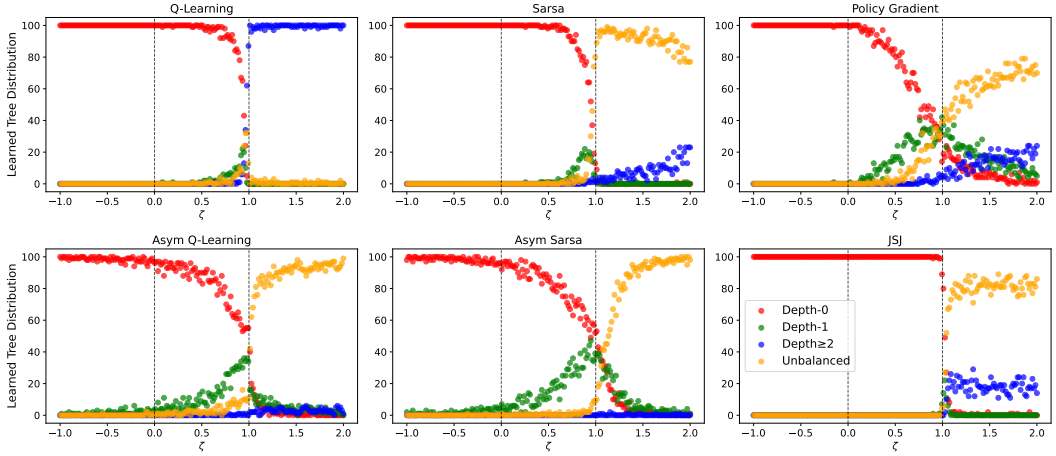
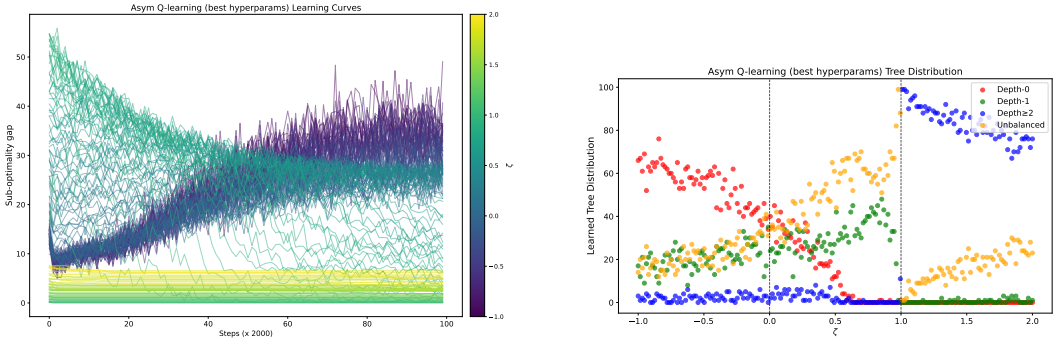



Figure 12: Distributions of final tree policies learned across the 100 seeds. For each $\zeta$ value, there are four colored points. Each point represent the share of depth-0 trees (red), depth-1 trees (green), unbalanced depth-2 trees (orange) and depth-2 trees (blue).




(a) Learning curves for asymmetric Q-learning with good hyperparameters.

(b) Trees distributions for asymmetric Q-learning with good hyperparameters

Figure 13: Analysis of the top-performing asymmetric Q-learning instantiation. (left) Learning curves, and (right) tree distributions across different POIBMDP configurations.

**Algorithm 5:** Asymmetric Sarsa

**Data:** POMDP $\mathcal{M}_{po} = \langle X, O, A, R, T, T_0, \Omega \rangle$, learning rates $\alpha_u, \quad \alpha_q$, exploration rate $\epsilon$
**Result:** $\pi : O \to A$
Initialize $U(x, a) = 0$ for all $x \in X, a \in A$
Initialize $Q(o, a) = 0$ for all $o \in O, a \in A$
**for** *each episode* **do**
  Initialize state $x_0 \sim T_0$
  Initialize observation $\boldsymbol{o}_0 \sim \Omega(x_0)$
  Choose action $a_0$ using $\epsilon$-greedy: $a_0 = \text{argmax}_a \, Q(\boldsymbol{o}_0, a)$ with prob. $1 - \epsilon$
  **for** *each step* $t$ **do**
    Take action $a_t$, observe $r_t = R(x_t, a_t)$, $x_{t+1} \sim T(x_t, a_t)$, and $\boldsymbol{o}_{t+1} \sim \Omega(x_{t+1})$
    Choose action $a_{t+1}$ using $\epsilon$-greedy: $a_{t+1} = \text{argmax}_a \, Q(\boldsymbol{o}_{t+1}, a)$ with prob. $1 - \epsilon$
    $y \leftarrow r + \gamma U(x_{t+1}, a_{t+1})$ `// TD target using actual next action`
    $U(x_t, a_t) \leftarrow (1 - \alpha_u) U(x_t, a_t) + \alpha_u y$
    $Q(\boldsymbol{o}_t, a_t) \leftarrow (1 - \alpha_q) Q(\boldsymbol{o}_t, a_t) + \alpha_q y$
    $x_t \leftarrow x_{t+1}$
    $\boldsymbol{o}_t \leftarrow \boldsymbol{o}_{t+1}$
    $a_t \leftarrow a_{t+1}$
  **end**
**end**
$\pi(\boldsymbol{o}) = \text{argmax}_a \, Q(\boldsymbol{o}, a)$ `// Extract greedy policy`

**Algorithm 6:** Asymmetric policy gradient algorithm. Uses Monte Carlo estimates of the average reward value functions to perform policy improvements.

**Data:** POMDP $\mathcal{M}_{po} = \langle X, O, A, R, T, T_0, \Omega \rangle$, learning rate $\alpha$, policy parameters $\theta$, number of trajectories $N$
**Result:** Stochastic partially observable policy $\pi_\theta : O \to \Delta(A)$
Initialize policy parameters $\theta$
Initialize $Q(\boldsymbol{o}, a) = 0$ for all observations $o$ and actions $a$
**for** *each episode* **do**
  **for** $i = 1$ *to* $N$ **do**
    Generate trajectory $\tau_i = (\boldsymbol{s}_0, a_0, r_0, \boldsymbol{s}_1, a_1, r_1, \ldots, \boldsymbol{s}_T)$ following $\pi_\theta$
    **for** *each timestep* $t$ *in trajectory* $\tau_i$ **do**
      $G_t \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r_k$ `// Compute return`
      Store $(\boldsymbol{o}_t, a_t, G_t)$ for later averaging
    **end**
  **end**
  **for** *each unique observation-action pair* $(o, a)$ **do**
    $Q(o, a) \leftarrow \frac{1}{|\{(\boldsymbol{o}, a)\}|} \sum_{(\boldsymbol{o}, a, G)} G$ `// Monte Carlo estimate`
  **end**
  **for** *each observation* $o$ **do**
    **for** *each action* $a$ **do**
      $\pi_1(a|\boldsymbol{o}) \leftarrow 1.0$ if $a = \text{argmax}_{a'} \, Q(\boldsymbol{o}, a')$, 0.0 otherwise `// Deterministic policy from Q-values`
      $\pi(a|\boldsymbol{o}) \leftarrow (1 - \alpha)\pi(a|\boldsymbol{o}) + \alpha \pi_1(a|\boldsymbol{o}o)$ `// Policy improvement step`
    **end**
  **end**
  Reset $Q(\boldsymbol{o}, a) = 0$ for all observations $\boldsymbol{o}$ and actions $a$ `// Reset for next episode`
**end**

Table 6: Summary of RL baselines Hyperparameters

| algorithm | Problem | Hyperparameters comb. |
|---|---|---|
| Policy Gradient | PO/IB/MDP | 420 |
| JSJ | POIBMDP | 15 |
| Q-learning | PO/IB/MDP | 192 |
| Asym Q-learning | POIBMDP | 768 |
| Sarsa | PO/IB/MDP | 192 |
| Asym Sarsa | POIBMDP | 768 |

Table 7: PG Hyperparameter Space (140 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Learning Rate (lr) | 0.001, 0.005, 0.01, 0.05, 0.1 | Policy gradient step size |
| Entropy Regularization (tau) | -1.0, -0.1, -0.01, 0.0, 0.01, 0.1, 1.0 | Entropy regularization coefficient |
| Temperature (eps) | 0.01, 0.1, 1.0, 10 | Softmax temperature |
| Episodes per Update (n_steps) | 20, 200, 2000 | Number of episodes per policy update |

Table 8: PG-IBMDP Hyperparameter Space (140 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Learning Rate (lr) | 0.001, 0.005, 0.01, 0.05, 0.1 | Policy gradient step size |
| Entropy Regularization (tau) | -1.0, -0.1, -0.01, 0.0, 0.01, 0.1, 1.0 | Entropy regularization coefficient |
| Temperature (eps) | 0.01, 0.1, 1.0, 10 | Softmax temperature |
| Episodes per Update (n_steps) | 10, 100, 1000 | Number of episodes per policy update |

Table 9: QL Hyperparameter Space (192 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Epsilon Schedules | (0.3, 1), (0.3, 0.99), (1, 1) | Initial exploration and decrease rate |
| Epsilon Schedules | (0.1, 1), (0.1, 0.99), (0.3, 0.99) | Initial exploration and decrease rate |
| Lambda | 0.0, 0.3, 0.6, 0.9 | Eligibility trace decay |
| Learning Rate (lr_o) | 0.001, 0.005, 0.01, 0.1 | Observation Q-learning rate |
| Optimistic | True, False | Optimistic initialization |

Table 10: QL-Asym Hyperparameter Space (768 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Epsilon Schedules | (0.3, 1), (0.3, 0.99), (1, 1) | Initial exploration and decrease rate |
| Epsilon Schedules | (0.1, 1), (0.1, 0.99), (0.3, 0.99) | Initial exploration and decrease rate |
| Lambda | 0.0, 0.3, 0.6, 0.9 | Eligibility trace decay |
| Learning Rate (lr_o) | 0.001, 0.005, 0.01, 0.1 | Observation Q-learning rate |
| Learning Rate (lr_v) | 0.001, 0.005, 0.01, 0.1 | State-action Q-learning rate |
| Optimistic | True, False | Optimistic initialization |

Table 11: QL-IBMDP Hyperparameter Space (192 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Epsilon Schedules | (0.3, 1), (0.3, 0.99), (1, 1) | Initial exploration and decrease rate |
| Epsilon Schedules | (0.1, 1), (0.1, 0.99), (0.3, 0.99) | Initial exploration and decrease rate |
| Lambda | 0.0, 0.3, 0.6, 0.9 | Eligibility trace decay |
| Learning Rate (lr_v) | 0.001, 0.005, 0.01, 00.1 | State-action Q-learning rate |
| Optimistic | True, False | Optimistic initialization |

Table 12: SARSA Hyperparameter Space (192 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Epsilon Schedules | (0.3, 1), (0.3, 0.99), (1, 1) | Initial exploration and decrease rate |
| Epsilon Schedules | (0.1, 1), (0.1, 0.99), (0.3, 0.99) | Initial exploration and decrease rate |
| Lambda | 0.0, 0.3, 0.6, 0.9 | Eligibility trace decay |
| Learning Rate (lr_o) | 0.001, 0.005, 0.01, 0.1 | Observation SARSA learning rate |
| Optimistic | True, False | Optimistic initialization |

Table 13: SARSA-Asym Hyperparameter Space (768 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Epsilon Schedules | (0.3, 1), (0.3, 0.99), (1, 1) | Initial exploration and decrease rate |
| Epsilon Schedules | (0.1, 1), (0.1, 0.99), (0.3, 0.99) | Initial exploration and decrease rate |
| Lambda | 0.0, 0.3, 0.6, 0.9 | Eligibility trace decay |
| Learning Rate (lr_o) | 0.001, 0.005, 0.01, 0.1 | Observation SARSA learning rate |
| Learning Rate (lr_v) | 0.001, 0.005, 0.01, 0.1 | State-action SARSA learning rate |
| Optimistic | True, False | Optimistic initialization |

Table 14: SARSA-IBMDP Hyperparameter Space (192 combinations)

| Hyperparameter | Values | Description |
|---|---|---|
| Epsilon Schedules | (0.3, 1), (0.3, 0.99), (1, 1) | Initial exploration and decrease rate |
| Epsilon Schedules | (0.1, 1), (0.1, 0.99), (0.3, 0.99) | Initial exploration and decrease rate |
| Lambda | 0.0, 0.3, 0.6, 0.9 | Eligibility trace decay |
| Learning Rate (lr_v) | 0.001, 0.005, 0.01, 0.1 | State-action SARSA learning rate |
| Optimistic | True, False | Optimistic initialization |

Table 15: Asymmetric sarsa hyperparameters (768 combinations each run 10 times)

| Hyperparameter | Values | Description |
|---|---|---|
| Epsilon Schedules | (0.3, 1), (0.3, 0.99), (1, 1) | Initial exploration and decrease rate |
| Epsilon Schedules | (0.1, 1), (0.1, 0.99), (0.3, 0.99) | Initial exploration and decrease rate |
| Lambda | 0.0, 0.3, 0.6, 0.9 | Eligibility trace decay |
| Learning Rate $U$ | 0.001, 0.005, 0.01, 0.1 | learning rate for the Q-function |
| Learning Rate $Q$ | 0.001, 0.005, 0.01, 0.1 | learning rate for the partial observation dependent Q-function |
| Optimistic | True, False | Optimistic initialization |

| Hyperparameter | Asym Q-learning (10/10) | Asym Sarsa (10/10) | PG (4/10) |
|---|---|---|---|
| epsilon_start | 1.0 | 1.0 | - |
| epsilon_decay | 0.99 | 0.99 | - |
| batch_size | 1 | 1 | - |
| lambda_ | 0.0 | 0.0 | - |
| lr_o | 0.01 | 0.1 | - |
| lr_v | 0.1 | 0.005 | - |
| optimistic | False | False | - |
| lr | - | - | 0.05 |
| tau | - | - | 0.1 |
| eps | - | - | 0.1 |
| n_steps | - | - | 2000 |

Table 16: Best hyperparameters for each algorithm on the POIBMDP problem

## 12 POIBMDPs for classification tasks

Let us show that, POIBMDPs associated with MDPs encoding supervised learning tasks, are in fact MDPs themselves. Let us define such supervised learning MDPs in the context of a classification task (this definition extends trivially to regression tasks).

**Definition 9** (Classification Markov decision process)**.** Given a set of $N$ examples denoted $\mathcal{E} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where each datum $\boldsymbol{x}_i \in \mathcal{X}$ is described by a set of $p$ features $x_{ij}$ with $1 \leq j \leq p$, and $y_i \in \mathbb{Z}^m$ is the label associated with $\boldsymbol{x}_i$, a classification Markov decision Process is an MDP $\langle S, A, R, T, T_0 \rangle$ (definition 1). The state space is $S = \{\boldsymbol{x}_i\}_{i=1}^N$, the set of training data features. The action space is $A = \mathbb{Z}^m$, the set of unique labels. The reward function is $R : S \times A \to \{0, 1\}$ with $R(\boldsymbol{s} = \boldsymbol{x}_i, a) = 1_{\{a=y_i\}}$. The transition function is $T : S \times A \to \Delta(S)$ with $T(\boldsymbol{s}, a, \boldsymbol{s}') = \frac{1}{N} \quad \forall \boldsymbol{s}, a, \boldsymbol{s}'$. The initial distribution is $T_0(\boldsymbol{s}_0 = \boldsymbol{s}) = \frac{1}{N}$.

One can be convinced that policies that maximize the RL objective (definition 2) in classification MDPs are classifiers that maximize the prediction accuracy because $\sum_{i=1}^N 1_{\pi(\boldsymbol{x}_i)=y_i} = \sum_{i=1}^N R(\boldsymbol{x}_i, \pi(\boldsymbol{x}_i))$. We defer the formal proof in the next part of the manuscript in which we extensively study supervised learning problems.

Now let us show that associated POIBMDPs are in fact MDPs. We show this by construction.

**Definition 10** (Classification POIBMDP)**.** Given a classification MDP $\langle \{\boldsymbol{x}_i\}_{i=1}^N, \mathbb{Z}^m, R, T, T_0 \rangle$ (definition 9), and an associated POIBMDP $\langle S, O, A, A_{info}, R, \zeta, T_{info}, T, T_0 \rangle$ (definition 7), a classification POIBMDP is an MDP (definition 1):

$$\langle \overbrace{O}^{\text{State space}}, \overbrace{\mathbb{Z}^m, A_{info}}^{\text{Action space}}, \overbrace{R, \zeta}^{\text{Reward function}}, \overbrace{\mathcal{P}, \mathcal{P}_0}^{\text{Transition functions}} \rangle$$

$O$ is the set of possible observations in $[L_1, U_1] \times \cdots \times [L_p, U_p] \times [L_1, U_1] \times \cdots \times [L_p, U_p]$ where $L_j$ is the minimum value of feature $j$ over all data $\boldsymbol{x}_i$ and $U_j$ the maximum. $\mathbb{Z}^m \cup A_{info}$ is action space: actions can be label assignments in $\mathbb{Z}^m$ or bounds refinements in $A_{info}$. The reward for assigning label $a \in \mathbb{Z}^m$ when observing some observation $\boldsymbol{o} = (L'_1, U'_1, \ldots, L'_p, U'_p)$ is the proportion of training data satisfying the bounds and having label $a$: $R(o, a) = \frac{|\{\boldsymbol{x}_i : L'_j \leq x_{ij} \leq U'_j \forall i,j\} \cap \{\boldsymbol{x}_i : y_i = a \forall i\}|}{|\{\boldsymbol{x}_i : L'_j \leq x_{ij} \leq U'_j \forall i,j\}|}$. The reward for taking an information gathering action that refines bounds is $\zeta$. The transition function is $\mathcal{P} : O \times (\mathbb{Z}^m \cup A_{info}) \to \Delta(O)$. When $a \in \mathbb{Z}^m$, $\mathcal{P}(\boldsymbol{o}, a, (L_1, U_1, \ldots, L_p, U_p)) = 1$ (reset to full bounds). When $a = (k, v) \in A_{info}$, from $\boldsymbol{o} = (L'_1, U'_1, \ldots, L'_p, U'_p)$, the MDP will transit to $\boldsymbol{o}_{left} = (L'_1, U'_1, \ldots, L_k, v, \ldots, L'_p, U'_p)$ (resp. $\boldsymbol{o}_{right} = (L'_1, U'_1, \ldots, U'_k, v, \ldots, L'_p, U'_p)$) with probability $\frac{|\{\boldsymbol{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j \wedge x_{ik} \leq v\}|}{|\{\boldsymbol{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j\}|}$ (resp. $\frac{|\{\boldsymbol{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j \wedge x_{ik} > v\}|}{|\{\boldsymbol{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j\}|}$).

Those classification POIBMDPs are essentially MDPs with stochastic transitions. It means that deterministic memoryless policies $O :\to A \cup A_{info}$ are in fact Markovian policy for those classification POIBMDPs. More importantly, it means that, for a given $\gamma$ and $\zeta$, if we were to know the whole POIBMDP model, we could use planning, to compute *optimal* decision tree policies.

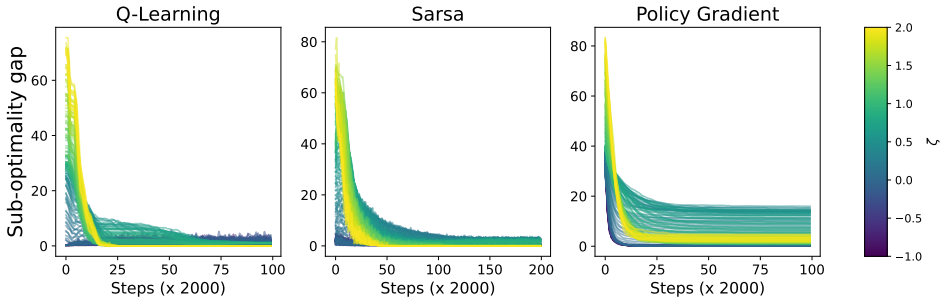



Figure 14: We reproduce the same plot as in figure 11 for classification POIBMDPs. Each individual curve is the sub-optimality gap of the learned policy during training averaged over 100 runs for a single $\zeta$ value.