Limits of reinforcement learning for decision trees in Markov decision processes

Anonymous Authors[1]

Abstract

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

1. Introduction

Interpretability in machine learning is commonly 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 tools such as 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). Although widely used, 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 learned model is transparent by construction. Decision trees (Breiman et al., 1984) are a canonical example, as their predictions can be inspected, reasoned about, and formally verified. This makes them particularly attractive for safety-critical applications and has motivated extensive research in supervised learning (Murthy & Salzberg, 1995; Verwer & Zhang, 2019; Demirovic et al., 2022; Demirović et al., 2023; 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 learned using reinforcement learning, 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 and enable formal verification (Wu et al., 2024), but they optimize a surrogate objective—policy imitation—rather than the original reinforcement learning objective. As a result, the best decision tree policy for the task may differ substantially from the tree that best approximates a neural expert.

This limitation motivates the study of *direct* approaches that learn interpretable policies by optimizing the reinforcement learning objective itself. While direct decision tree learning is well understood in supervised settings, it is far less developed for sequential decision making. Understanding why direct optimization is difficult—and when it can succeed—is the central focus of this work.

We show that direct reinforcement learning of decision tree policies for MDPs, i.e. learning a decision tree that optimizes the cumulative reward of the process without relying on a black-box expert, is often very difficult. In particular, we provide some insights as to why it is so difficult and show that imitating a neural network expert policy with a decision tree, despite not somving the downstream task, often yields very good tree policies in practice.

In section **??**, we describe Nicholay Topin and colleagues' framework for direct reinforcement learning of decision tree policies (Topin et al., 2021) and reproduce their key experiment. In section **??**, we show that this direct approach is

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

equivalent to learning a deterministic memoryless policy for partially observable MDP (POMDP)(Sondik, 1978)–which is a hard problem (Littman, 1994)–and show that this might be the main reason for failures. In section **??**, we further support this claim by constructing special instances of such POMDPs where the observations contain all the information about hidden states, and show that in those cases, direct reinforcement learning of decision trees works well.

2. Related work

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

Parametric decision trees fix the tree structure *a priori*—including depth, node arrangement, and selected state features—and only learn the decision thresholds. This formulation enables differentiability and allows direct optimization of the RL objective using policy gradient methods (Sutton et al., 1999). Several works (Silva et al., 2020; Vos & Verwer, 2024; Marton et al., 2025) employ PPO to train such differentiable trees. 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 learning in order to outperform indirect imitation methods such as VIPER.

Non-parametric decision trees, by contrast, are the standard model in supervised learning, where greedy algorithms (Breiman et al., 1984; Quinlan, 1986; 1993) efficiently construct trees that balance predictive performance and interpretability. However, their extension to reinforcement learning remains largely unexplored. To the best of our knowledge, the only work that studies learning non-parametric decision trees that optimize a trade-off between interpretability and cumulative rewards is that of Topin et al. (Topin et al., 2021). Topin et al. introduce *iterative bounding MDPs* (IBMDPs), which augment a base MDP with additional state features, actions, rewards, and transitions. They show that certain policies in the IBMDP correspond to decision tree policies for the base MDP. Hence, standard RL algorithms can be used to learn such policies in IBMDPs.

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 dynamics and rewards are known, (Vos & Verwer, 2023) use planning to compute shallow parametric decision tree policies (up to depth 3). Next, we recall useful technical material.

3. Technical preliminaries

3.1. Markov decision processes

Markov decision processes (MDPs) were first introduced in the 1950s by Richard Bellman (Bellman, 1957). Informally, an MDP models how an agent acts over time to achieve a goal. At every time step, the agent observes its current state (e.g., patient weight and tumor size) and takes an action (e.g., administers a certain amount of chemotherapy). The agent receives a reward that helps evaluate the quality of the action with respect to the goal (e.g., tumor size decreases when the objective is to cure cancer). Finally, the agent transitions to a new state (e.g., the updated patient state) and repeats this process over time. Following Martin L. Puterman's book on MDPs (Puterman, 1994), we formally define:

Definition 3.1 (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 3.2 (Reinforcement learning objective)**.** Given an MDP $\mathcal{M} \equiv \langle S, A, R, T, T_0 \rangle$, the goal of reinforcement learning for sequential decision making is to find a model, also known as 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 controls the trade-off between immediate and future rewards.

Algorithms presented in this manuscript aim to find an optimal policy $\pi^\star \in \underset{\pi}{\operatorname{argmax}}\, J(\pi)$ that maximizes the above reinforcement learning (RL) objective. In particular, RL algorithms (Sutton & Barto, 1998; **?**; Watkins & Dayan, 1992; Mnih et al., 2015; Schulman et al., 2017) learn 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 3.3 (Value of a state)**.** In an MDP $\mathcal{M}$ (cf. definition 3.1), the value of a state $s \in S$ under policy π is the expected discounted sum of rewards starting from state s and following policy π:

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

Applying the Markov property gives a recursive definition of the value of s under policy π: $V^\pi(s) = R(s, \pi(s)) + \gamma \mathbb{E}\left[V^\pi(s')|s' \sim T(s, \pi(s))\right]$. 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 (cf. definition 3.2)): $V^\star(s) = V^{\pi^\star}(s)$. 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: $Q^\star(s, a) = R(s, a) + \gamma \mathbb{E}\left[V^\star(s')|s' \sim T(s, a)\right]$.

Next, we present the class of MDPs introduced in (Topin et al., 2021) useful for our goal of direct reinforcement learning of decision tree policies.

3.2. Iterative bounding Markov decision processes

The key thing to know about IBMDPs is that they are, as their name suggests, MDPs. Hence, IBMDPs admit an optimal deterministic Markovian policy that maximizes the RL objective. In this part we will assume that all the MDPs we consider are MDPs with continuous state spaces (cf. section **??**) with a finite set of actions, so we use bold fonts for states and observations as they are vector-valued. However all our results generalize to discrete states (in $\mathbb{Z}^m$) MDPs that we can factor using one-hot encodings. Given an MDP for which we want to learn a decision tree policy–the base MDP–IBMDP states are concatenations of the base MDP state features and some observations. Those observations are information about the base state features that are refined–"iteratively bounded"– at each step. Those observations essentially represent some knowledge about where some base state features lie in the state space. Actions available in an IBMDP are: 1) the actions of the base MDP, that change base state features, and 2) *information gathering* actions that change the aforementioned observations. Now, base actions in an IBMDP are rewarded like in the base MDP, this ensures that the RL objective w.r.t. the base MDP is encoded in the IBMDP reward. When taking an information gathering action, the reward is an arbitrary value such that optimizing the RL objective in the IBMDP is equivalent to optimizing some trade-off between interpretability and the RL objective in the base MDP.

Before showing how to get decision tree policies from IBMDP policies, we give a formal definition of IBMDPs following Topin et. al. (Topin et al., 2021).

Definition 3.4 (Iterative bounding Markov decision process)**.** Given an MDP $\mathcal{M} \equiv \langle S, A, R, T, T_0 \rangle$ (cf. definition 3.1), 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 base MDP state features. Base state features $\boldsymbol{s} = (s_1, \dots, s_p) \in S$ 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 base 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 base state features and observations. Given some base state features $\boldsymbol{s} = (s_1, \dots, s_p) \in S$ and some observation $\boldsymbol{o} = (L_1, U_1, \dots, L_p, U_p)$, an IBMDP state is $\boldsymbol{s}_{IB} = (\overbrace{s_1, \dots, s_p}^{\text{base state features}}, \overbrace{L_1, U_1, \dots, L_p, U_p}^{\text{observation}})$. A are the base MDP actions. A_{info} are *information gathering* actions (IGAs) 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 base MDP actions and information gathering actions $A \cup A_{info}$. $R : S \times A \to \mathbb{R}$ is the base MDP reward function. ζ is a reward signal for taking an information gathering action. So the IBMDP reward function is to get a reward from the base MDP if the action is a base MDP action or to get ζ 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 $\boldsymbol{o}_t = (L'_1, U'_1, \dots, L'_p, U'_p) \in O$ and base state features $\boldsymbol{s}_t = (s'_1, s'_2, \dots, s'_p)$ if an IGA $\langle j, v \rangle$ is taken, the new observation is:

$$\boldsymbol{o}_{t+1} = \begin{cases} (L'_1, U'_1, \dots, L'_j, \min\{v, U'_j\}, \dots, L'_p, U'_p) \text{ if } s_j \leq v \\ (L'_1, U'_1, \dots, \max\{v, L'_j\}, U'_j, \dots, L'_p, U'_p) \text{ if } s_j > v \end{cases}$$

If a base action is taken, the observation is reset to the default base state feature bounds $(L_1, U_1, \dots, L_p, U_p)$ and the base state features change according to the base MDP transition function: $\boldsymbol{s}_{t+1} \sim T(\boldsymbol{s}_t, a_t)$. At initialization, the base state features are drawn from the base MDP initial distribution T_0 and the observation is always set to the default base state features bounds $\boldsymbol{o}_0 = (L_1, U_1, \dots, L_p, U_p)$.

We present an IBMDP for a simple grid-world MDP in appendix **??**. Now remains to extract a decision tree policy for MDP $\mathcal{M}$ from a policy for an associated IBMDP $\mathcal{M}_{IB}$.

3.3. From policies to trees

One can notice that information gathering actions (cf. definition 3.4) resemble the Boolean functions $1_{\{x_{\cdot,j} \leq v\}}$ that make up internal decision tree nodes (cf. figure 8). Indeed,

Algorithm 1 Extract a Decision Tree Policy (algorithm 1 from (Topin et al., 2021))

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

a policy taking actions in an IBMDP essentially builds a tree by taking sequences of IGAs (internal nodes) and then a base action (leaf node) and repeats this process over time. In particular, the IGA rewards ζ can be seen as a regularization or a penalty for interpretability: if ζ is very small compared to base rewards, a policy will try to take base actions as often as possible, i.e. build shallow trees with short paths between root and leaves.

Authors from (Topin et al., 2021) show that not all IBMDP policies are decision tree policies for the base MDP. In particular, their algorithm that converts IBMDP policies into decision trees (cf. algorithm 1) takes as input deterministic policies depending solely on the observations of the IBMDP.

While the connections between partially observable MDPs (POMDPs (Sondik, 1978; Sigaud & Buffet, 2013)) and extracting decision tree policies from IBMDPs is obvious, they are absent from the original IBMDP paper (Topin et al., 2021). In the next section we bridge this gap.

3.4. Bridging the gap with the partially observable MDPs literature

A POMDP is an MDP where the current state is hidden; only some information about the current state is observable.

Definition 3.5 (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.

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

Definition 3.6 (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 (cf. definition 3.4). The transition function Ω 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 base MDP state features as the *hidden variables* and feature bounds as *visible* variables. POIBMDPs can also be seen as non-stationary MDPs (N-MDPs) (Singh et al., 1994) in which there is one different transition function per base MDP state: these are called hidden-mode MDPs (Choi et al., 2001). Following (Singh et al., 1994) we can write the value of a deterministic partially observable policy $\pi : O \to A \cup A_{info}$ in observation $\boldsymbol{o}$.

Definition 3.7 (Partially observable value function)**.** In a POIBMDP (cf. definition 3.6), the expected cumulative discounted reward of a deterministic partially observable 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 (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 (cf. definition 3.3). We abuse notation and denote both values of observations and values of states by V since the function input is not ambiguous.

The asymptotic occupancy distribution is the probability of a policy π to arrive in $(s, \boldsymbol{o}')$ while observing $\boldsymbol{o}$ in some trajectory. Now that we presented all the necessary technical backgrounds we move on to our contributions.

4. Methodology

The goal of this section is to check if the direct approach described above can consistently retrieve optimal decision tree policies for a simple 2×2 grid world MDP. In particular, we use reinforcement learning to train decision tree policies for MDPs by seeking deterministic partially observable policies that optimize the RL objective in POIBMDPs (cf. example **??**).

4.1. Computing some decision tree policies

To assess the performance of reinforcement learning, we identify decision tree policies that maximize the RL objective in POIBMDPs with different trade-off rewards ζ for different discount factors γ. Each of those policies can be one of the trees illustrated in figure 1: (i) a depth-0 tree equivalent to always taking the same base action ($\pi_{\mathcal{T}_0}$), (ii) a depth-1 tree equivalent alternating between an IGA and a base action ($\pi_{\mathcal{T}_1}$), (iii) an unbalanced depth-2 tree that sometimes takes two IGAs then a base action and sometimes a an IGA then a base action ($\pi_{\mathcal{T}_u}$), (iv) a depth-2 tree that alternates between taking two IGAs and a base action ($\pi_{\mathcal{T}_2}$), or (v) an infinite "tree" that only takes IGAs. Furthermore, because from (Singh et al., 1994) we know that for POMDPs, stochastic partially observable policies can sometimes get better expected discounted rewards than deterministic partially observable policies, we also compute the value of the stochastic policy that randomly alternates between two base actions: $\rightarrow$ and $\downarrow$. Taking those two base actions always lead to the goal state in expectation (cf. figure **??**). Because we know all the base states, all the observations, all the actions, all the rewards and all the transitions of our POIBMDP (cf. example 11), using definiton 3.7, we can compute exactly the values of those different deterministic partially observable policies given ζ the reward for IGAs and γ the discount factor.

We plot, in figure 2, the RL objective values of the decision tree policies as functions of ζ when we fix $\gamma = 0.99$ (standard choice of discount in practice **??**). When $\gamma = 0.99$, 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 the optimal one, w.r.t. the RL objective, deterministic partially observable POIBMDP policy for $0 < \zeta < 1$.

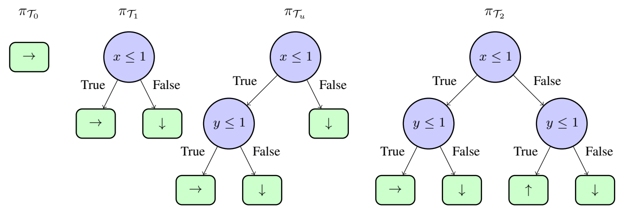


Figure 1. For each decision tree structure, e.g., depth-1 or unbalanced depth-2, we illustrate a decision tree which maximizes the RL objective (cf. definition 3.2) in the grid world MDP.

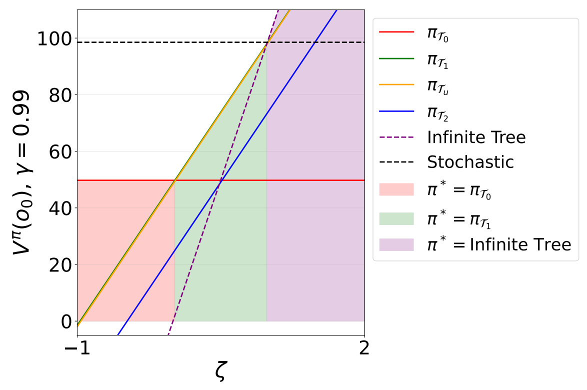


Figure 2. RL objective values (cf. definition **??**) of different partially observable policies as functions of ζ. Shaded areas show the optimal *deterministic* partially observable policies in different ranges of ζ values.

4.2. Reinforcement learning algorithms

In general, the policy that maximizes the RL objective (cf. definition 3.2) in a POMDP (cf. definition 3.5) maps "belief states" or observation histories (Sigaud & Buffet, 2013) to actions. Hence, those policies do not correspond to decicion trees since we require that policies depend only on the current observation. If we did not have this constraint, we could apply any standard RL algorithm to solve POIBMDPs by seeking such policies because both histories and belief states are sufficient statistics for POMDP hidden states (Sigaud & Buffet, 2013; Lambrechts et al., 2025a).

In particular, the problem of finding the optimal deterministic partially observable policies for POMDPs is NP-HARD, even with full knowledge of transitions and rewards (**?**)section 3.2]littman1. 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 partially observable policies. Hence it is interesting to study reinforcement learning for finding the best deterministic partially observable policy since it would not search the whole solution space. However applying RL to our interpretable RL objective (cf. definition **??**) is non-trivial.

In (Singh et al., 1994), the authors show that the optimal partially observable policy can be stochastic. Hence, policy gradient algorithms (Sutton et al., 1999)–that return stochastic policies–are to avoid since we seek the best *deterministic* policy. Furthermore, the optimal deterministic partially observable policy might not maximize all the values of all observations simultaneously (Singh et al., 1994) which makes it difficult to use TD-learning (cf. algorithms 3 and **??**). Indeed, doing a TD-learning update of one partially observable value (cf. definition 3.7) with, e.g. Q-learning, can change the value of *all* other observations in an uncontrollable manner because of the dependence in $P^\pi((s, \boldsymbol{o}')|\boldsymbol{o})$ (cf. definition 3.7). Interestingly, those two challenges of learning in POMDPs described in (Singh et al., 1994) are visible in figure 2. First, there is a whole range of ζ values for which the optimal partially observable policy is stochastic. Second, for e.g. $\zeta = 0.5$, while a depth-1 tree is the optimal deterministic partially observable policy, the value of state $(\boldsymbol{s}_2, \boldsymbol{o}_0) = (1.5, 1.5, 0, 2, 0, 2)$ is not maximized by this partially observable policy but by the sub-optimal policy that always goes down.

Despite those hardness results, empirical results of applying RL to POMDPs by naively replacing $\boldsymbol{x}$ by $\boldsymbol{o}$ in Q-learning or Sarsa, has already demonstrated successful in practice (Loch & Singh, 1998). More recently, the framework of Baisero et. al. called asymmetric RL (Baisero et al., 2022; Baisero & Amato, 2022) has also shown promising results to learn POMDP solutions. 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. 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 algorithm 2 we present asymmetric Q-learning. It is a variant of Q-learning (cf. algorithm 3) that returns a deterministic partially observable policy like modified DQN 4. 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 temporal difference learning update. We also consider an asymmetric version of Sarsa that applies similar modifications to the standard Sarsa (cf. algorithm **??**). We present asymmetric Sarsa in the appendix (cf. algorithm **??**). In (Jaakkola et al., 1994), the authors introduce a policy search algorithm A.2 that learns a (stochastic) policy $\pi : O \to \Delta(A)$ and a critic $V : X \to \mathbb{R}$ using Monte Carlo estimates to guide policy improvement. We also consider this algorithm in our experiments that we call JSJ (for the authors names Jaakkola, Singh, Jordan). We present the JSJ algorithm in the appendix (cf. algorithm **??**). JSJ is equivalent to a tabular asymmetric policy gradient algorithm (cf. algorithm **??**).

Until recently, the benefits of asymmetric RL over standard RL was only shown empirically and only for history-dependent models. The work of Gaspard Lambrechts (Lambrechts et al., 2025b) proves that some asymmetric RL algorithms learn better history-dependent **or** partially observable policies for solving POMDPs. This is exactly what we wish for. However, those algorithms are not practical because they require estimations of the asymptotic occupancy distribution $P^\pi((s, \boldsymbol{o}')|\boldsymbol{o})$ (cf. definition 3.7) for candidate policies which in turn would require to gather a lot of on-policy samples. 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.

In the original work of Topin et. al. (Topin et al., 2021), they use RL algorithms corresponding to asymmetric DQN or asymmetric PPO from (Baisero et al., 2022; Baisero & Amato, 2022) before those were formally published.

Algorithm 2 Asymmetric Q-Learning. We highlight in green the differences with the standard Q-learning **??**

```
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: $a_t = \text{argmax}_a Q(\boldsymbol{o}_t, a)$ 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 $\boldsymbol{o}_{t+1} \sim \Omega(\boldsymbol{x}_{t+1})$
        $y \leftarrow r + \gamma U(\boldsymbol{x}_{t+1}, \text{argmax}_{a'} Q(\boldsymbol{o}_{t+1}, a'))$
        $U(\boldsymbol{x}_t, a_t) \leftarrow (1 - \alpha_u) U(\boldsymbol{x}_t, a_t) + \alpha_u y$
        $Q(\boldsymbol{o}_t, a_t) \leftarrow (1 - \alpha_q) Q(\boldsymbol{o}_t, a_t) + \alpha_q y$
        $x_t \leftarrow \boldsymbol{x}_{t+1}$
        $\boldsymbol{o}_t \leftarrow \boldsymbol{o}_{t+1}$
    end
end
$\pi(o) = \text{argmax}_a Q(\boldsymbol{o}, a)$
```

Next, we apply asymmetric and standard RL algorithms to the problem of learning the optimal depth-1 tree for the grid world MDP (cf. section 4) by optimizing the interpretable RL objective in POIBMDPs.

4.3. Results

The results presented in the section show that (asymmetric) reinforcement learning fails for the aforementioned problem. Let us understand why.

4.3.1. Experimental setup

Baselines: we consider two groups of RL algorithms. The first group is standard tabular RL algorithms to optimize the interpretable RL objective in POIBMDPs; Q-learning, Sarsa, and Policy Gradient with a softmax policy (cf. section A.2, algorithms 3, **??**, and **??**). In theory the Policy Gradient algorithm should not be a good candidate for our problem since it searches for stochastic policies that we showed can be better than our sought depth-1 decision tree policy (cf. figure 2).

In addition to the traditional tabular RL algorithms above, we also apply asymmetric Q-learning, asymmetric Sarsa, and JSJ (algorithms 2, **??** and **??**). We use at least 200 000 POIBMDP time steps per experiment. Each experiment, i.e an RL algorithm learning in a POIBMDP, is repeated 100 times.

Hyperparameters: For all baselines we use, when applicable, exploration rates $\epsilon = 0.3$ and learning rates $\alpha = 0.1$.

Metrics: We will consider two metrics. First, the sub-optimality gap during training, w.r.t. the interpretable RL objective, between the learned partially observable policy and the optimal deterministic partially observable 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 ζ the interpretable RL objective value of the optimal deterministic partially observable policy (cf. figure 2), we can report the *exact* sub-optimality gaps.

Second, we consider the distribution of the learned trees over the 100 training seeds. Indeed, since for every POIBMDP we run each algorithm 100 times, at the end of training we get 100 deterministic partially observable policies (we compute the greedy policy for stochastic policies returned by JSJ and Policy Gradient), from which we can extract the equivalent 100 decision tree policies using algorithm 1 and we can count which one are of e.g. depth 1. This helps understand which trees RL algorithms tend to learn.

4.4. Can (asymmetric) RL learn optimal deterministic partially observable POIBMDP policies?

In figure 15, 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 ζ values uniformly in $[-1, 2]$. In figure 15, a different color represents a different POIBMDP.

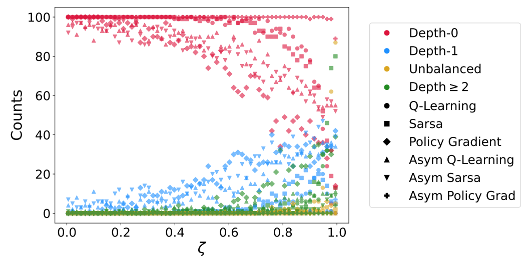


Figure 3. Distributions of final tree policies learned across the 100 seeds. For each ζ value, there are four colored points. Each point represent the share of depth-0 trees (red), depth-1 trees (green), unbalanced depth-2 trees (orange) and depth-2 trees (blue).

Recall from figure 2 that for: (i) $\zeta \in [-1, 0]$, the optimal deterministic partially observable policy is a depth-0 tree, (ii) $\zeta \in]0, 1[$, the optimal deterministic partially observable policy is a depth-1 tree, and (iii) $\zeta \in [1, 2]$, the optimal deterministic partially observable policy is a "infinite" tree that contains infinite number of internal nodes. We observe that, despite all sub-optimality gaps converging independently of the ζ 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.

In figure 3, we plot the distributions of the final learned trees over the 100 random seeds in function of ζ from the above runs. For example, in figure 3, in the top left plot, when learning 100 times in a POIBMDP with $\zeta = 0.5$, Q-learning returned almost 100 times a depth-0 tree. Again, on none of those subplots do we see a high rate of learned depth-1 trees for $\zeta \in]0, 1[$. It is alerting that the most frequent learned trees are the depth-0 trees for $\zeta \in]0, 1[$ because such trees are way more sub-optimal w.r.t. the interpretable RL objective (cf. definition **??**) than e.g. the depth-2 unbalanced trees (cf. figure 2). One interpretation of this phenomenon is that the learning in POIBMDPs is very difficult and so agents tend to converge to trivial policies, e.g., repeating the same base action.

However, on the positive side, we observe that asymmetric versions of Q-learning and Sarsa have found the optimal deterministic partially observable 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 difficult it is to do RL to learn partially observable policies in POIBMDPs.

4.5. How difficult is it to learn in POIBMDPs?

In this section we run the same (asymmetric) reinforcement learning algorithms to optimize either the RL objective (cf. definition 3.2) in MDPs (cf. definition 3.1) or IBMDPs (cf. definition 3.4), or the interpretable RL objective in POIBMDPs(cf. definition **??**). This essentially results in three distinct problems:

1. Learning an optimal standard Markovian policy in an MDP, i.e. optimizing the RL objective in an MDP.
2. Learning an optimal standard Markovian policy in an IBMDP, i.e. optimizing the RL objective in an IBMDP.
3. Learning an optimal deterministic partially observable policy in a POIBMDP.

In order to see how difficult each of these three problems is, we can run a *great* number of experiments for each problem and compare solving rates. To make solving rates comparable we consider a unique instance for each of those problems. Problem 1 is learning one of the optimal standard Markovian deterministic policy from figure **??** for the grid world from example **??** with $\gamma = 0.99$. Problem 2 is learning one of the optimal standard Markovian deterministic for the IBMDP from figure 11 with $\gamma = 0.99$ and $\zeta = 0.5$. This is similar to the previous chapter experiments where we applied DQN or PPO to an IBMDP for CartPole without constraining the search to partially observable policies (see e.g. figure12b). Problem 3 is what has been done in the previous section to learn deterministic partially observable policies where in addition of fixing $\gamma = 0.99$ we also fix $\zeta = 0.5$.

We use the six (asymmetric) RL algorithms from the previous section and try a wide set of hyperparameters and additional learning tricks (optimistic Q-function, eligibility traces, entropy regularization and ϵ-decay, all are described in (Sutton & Barto, 1998)). We only provide the detailed hyperparameters for asymmetric Sarsa and an overall summary for all the algorithms in tables 2 and 1. The complete detailed lists of hyperparameters are given in the appendix **??**. Furthermore, the careful reader might notice that there is no point running asymmetric RL on MDPs or IBMDPs when the problem does not require partial observability. Hence, we only run asymmetric RL for POIBMDPs and otherwise run all other RL algorithms and all problems.

Each unique hyperparameter combination for a given algorithm on a given problem is run 10 times on 1 million learning steps. For example, for asymmetric Sarsa, we run a total of $10 \times 768 = 7680$ experiments for learning deterministic partially observable policies for a POIBMDP (cf. table 2). To get a success rate, we can simply divide the number of learned depth 1 tree by 7680 (recall that for $\gamma = 0.99$ and $\zeta = 0.5$, the optimal policy is a depth-1 tree (e.g. figure 1) as per figure 2).

Table 1. Summary of RL baselines Hyperparameters

algorithm	**Problem**	**Total Hyperparameter Combination**
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

The key observations from figure 4 is that reinforcement learning a deterministic partially observable policy in a POIBMDP, is way harder than learning a standard Markovian policy. For example, Q-learning only finds the optimal solution (cf. definition **??**) in only 3% of the experiments while the same algorithms to optimize the standard RL objective (cf. definition 3.2) in an MDP or IBMDP found the optimal solutions 50% of the time. Even though asymmetry seems to increase performances; learning 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 difficult and costly as successes might require a million steps for such a seemingly simple problem. An other difficulty in practice that we did not cover here, is the choice of information gathering actions. For the grid world MDP, choosing good IGAs ($x \leq 1$ and $y \leq 1$) is simple but what about more complicated MDPs: how to instantiate the (PO)IBMDP action space such that internal nodes in resulting trees are useful for predictions?

To go even further, on figure 16 we re-run experiments from figure 15 and figure 3 using the top performing hyperparameters for asymmetric Q-learning (given in appendix **??**). While those hyperparameters resulted in asymmetric Q-learning returning 10 of out 10 times an optimal depth 1 tree, the performances didn't transfer. On figure 16 despite higher success rates in the region $\zeta \in]0, 1[$ compared to figure 3.

5. Conclusion

In this chapter, we have shown that direct learning of decision tree policies for MDPs can be reduced to learning deterministic partially observable policies in POMDPs that we called POIBMDPs. By crafting a POIBMDP for which we know exactly the optimal deterministic partially observable policy w.r.t. the interpretable RL objective (cf. definition **??**), we were able to benchmark the sub-optimality of solutions learned with (asymmetric) reinforcement learning.

Across our experiments, we found that no algorithm could

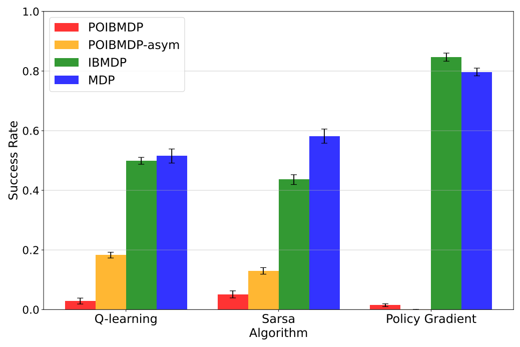


Figure 4. Success rates of different (asymmetric) RL algorithms over thousands of runs when applied to learning deterministic partially observable policies in a POIBMDP or learning deterministic policies in associated MDP and IBMDP.

consistently learn a depth-1 decision tree policy for a grid world MDP despite it being optimal both w.r.t. the interpretable RL objective and standard RL objective (cf. definition 3.2). When compared to the results of VIPER from figure 10, direct RL is worse at retrieving decision tree policies with good interpretability-performance trade-offs. This echoes the results from the previous chapter in which we saw that direct deep RL performed worse than imitation learning to find decision tree policies for CartPole.

In the next chapter, we find that RL can find optimal deterministic partially observable policies for a special class of POIBMDPs that we believe makes for a convincing argument as to why direct learning of decision tree policies that optimize the RL objective (cf. definition 3.2) is so difficult.

6. Classification tasks

In this section, we show that for a special class of POIBMDPs (cf. definition 3.6), reinforcement learning (cf. section A.2) can learn optimal deterministic partially observable policies w.r.t to the interpretable RL objective (cf. definition **??**), i.e. we can do direct decision tree policy learning for MDPs. This class of POIBMDPs are those for which base MDPs have uniform transitions, i.e. $T(\boldsymbol{s}, a, \boldsymbol{s}') = \frac{1}{|S|}$ (cf. definitions 3.1 and 3.4). The supervised learning objective (cf. definition **??**) can be re-formulated in terms of the RL objective (cf. definition 3.2) and MDPs with such uniform transitions. Indeed a supervised learning task can be formulated as maximizing the RL objective in an MDP where, actions are class (or target) labels, states are training data, the reward at every step is 1 if the correct label was predicted and 0 otherwise, and the transitions are uniform: the next state is given by uniformly sampling a new training datum. This implies that learning deterministic partially observable policies in POIBMDPs where the base MDP encodes a supervised learning task is equivalent to doing decision tree induction to optimize the supervised learning objective. If RL does work for such fully observable POIBMDPs, this would mean that: 1) the difficulty of direct learning of decision tree policies for *any* MDP using POIBMDPs, exhibited in the previous chapters, is most likely due to the partial observability, and 2), we can design new decision tree induction algorithms for the supervised learning objective by solving MDPs. 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 6.1 (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 $\langle S, A, R, T, T_0 \rangle$ (cf. definition 3.1) where:

- 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 (cf. definition 3.2) 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.

In figure 5 we give an example of such classification MDP with 4 data in the training set and 2 classes:

$$\mathcal{X} = \{(0.5, 0.5), (0.5, 1.5), (1.5, 1.5), (1.5, 0.5)\}$$
$$y = \{0, 0, 1, 1\}$$

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

Definition 6.2 (Classification POIBMDP)**.** Given a classification MDP $\langle \{\boldsymbol{x}_i\}_{i=1}^N, \mathbb{Z}^m, R, T, T_0 \rangle$ (cf. definition 6.1), and an associated POIBMDP $\langle S, O, A, A_{info}, R, \zeta, T_{info}, T, T_0 \rangle$ (cf. definition 3.6), a classification POIBMDP is an MDP (cf. definition 3.1):

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

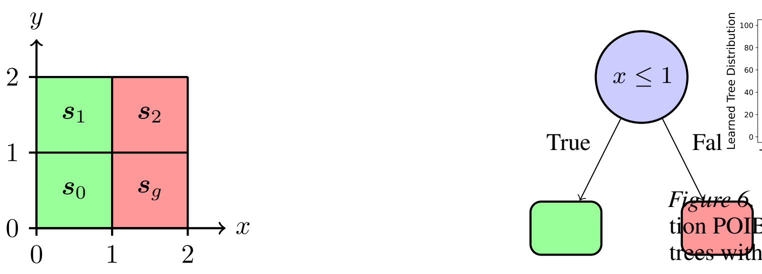


Figure 5. Classification MDP optimal actions. In this classification MDP, there are four data to which to assign either a green or red label. On the right, there is the unique optimal depth-1 tree for this particular classification MDP. This depth-1 tree also maximizes the accuracy on the corresponding classification task.

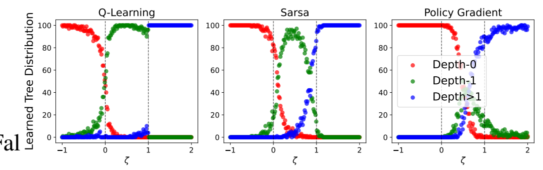


Figure 6. We reproduce the same plot as in figure 3 for classification POIBMDPs. Each colored dot is the number of final learned trees with a specific structure for a given ζ.

- 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 ζ

- The transition function is $\mathcal{P} : O \times (\mathbb{Z}^m \cup A_{info}) \rightarrow \Delta(O)$ where:
  - For $a \in \mathbb{Z}^m$: $\mathcal{P}(\boldsymbol{o}, a, (L_1, U_1, \ldots, L_p, U_p)) = 1$ (reset to full bounds)
  - For $a = (k, v) \in A_{info}$: 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 partially observable policies (cf. definition **??**) $O :\rightarrow A \cup A_{info}$ are in fact Markovian policy for those classification POIBMDPs. More importantly, it means that, for a given γ and ζ, if we were to know the whole POIBMDP model, we could use planning, e.g. value iteration (cf. definition **??**), to compute *optimal* decision tree policies. Similarly, standard RL algorithms like Q-learning (cf. definition A.2) should work as well as for any MDP to learn optimal decision tree policies.

This is exactly what we check next. We use the same direct approach to learn decision tree policies as in previous chapters, except that now the base MDP is a classification task and not a sequential decision making task.

7. How well do RL agents learn in classification POIBMDPs?

Similarly to the previous chapter, we are interested in a very simple classification POIBMDP. We study classification POIBMDPs associated with the example classification MDP from figure 5.

We construct classification POIBMDPs with $\gamma = 0.99$, 200 values of $\zeta \in [0, 1]$ and IGAs $x \leq 1$ and $y \leq 1$. Since classification POIBMDPs are MDPs, we do not need to analyze asymmetric RL and JSJ baselines like in the previous chapter (cf. algorithms 2, **??**, and **??**).

Fortunately this time, compared to general POIBMDPs, RL can be used to learn optimal deterministic partially observable policies $O :\rightarrow A \cup A_{info}$ w.r.t. the interpretable RL objective (cf. definition **??**) in classification POIBMDPs. Such policies are equivalent to decision tree classifiers. We observe on figure 6 that both Q-learning and Sarsa consistently minimize the sub-optimality gap independently of the interpretability-performance trade-off ζ. Hence they are able to learn the optimal depth-1 decision tree classifier (cf. figure 5) most of the time in the optimality range $\zeta \in]0, 1[$ (cf. figure 7).

8. Conclusion

In this part of the manuscript we were interested in algorithms that can learn decision tree policies that optimize some trade-off of interpretability and performance w.r.t. the RL objective (cf. definition 3.2) in MDPs. In particular, using the framework of Topin et. al. (Topin et al., 2021), we were able to explicitly write an interpretable RL objective function (cf. definition **??**).

In chapter **??**, we compared the algorithms proposed in (Topin et al., 2021) that directly optimize this objective, to imitation learning algorithms that only solve a proxy problem. While those direct RL algorithms are able to *learn*, i.e. find better and better solutions with time (cf. figures 12a and 12b), the decision tree policies returned perform worse in average than imitated decision trees w.r.t. the RL objective of interest (cf. figure 13) for similar number of nodes and depth.

We further analyzed the failure mode of direct learning of decision tree policies by making connections with POMDPs (Sondik, 1978; Sigaud & Buffet, 2013). In chapter **??**, we showed that learning decision tree policies for MDPs could be explicitly formulated as learning a deterministic partially observable (also known as memoryless or reactive) policy in a specific POMDP that we called POIBMDP (cf. definition 3.6). We showed that both RL and asymmetric RL, a class of algorithms specifically designed for POMDPs (Baisero & Amato, 2022; Baisero et al., 2022), were unable to consistently learn an optimal depth-1 decision tree policies for a very small grid world MDP when using the POIBMDP framework. In particular, we compared, in a very controlled experiment, the success rates of the same learning algorithms when seeking standard Markovian policies versus partially observable policies in decision processes that shared the same transitions and rewards (cf. section 4.5). We demonstrated on figure 4 that introducing partial observability greatly reduced the success rates (we also observed this implicitly on figures 12a and 12b).

Finally, in this chapter we showed that using RL to optimize the interpretable RL objective in fully observable POIBMDPs, i.e. POIBMDPs that are just MDPs, could learn optimal decision tree policies (cf. figures 6 and 7) adding new evidence that direct interpretable RL is difficult because it involves POMDPs.

This class of fully observable POIBMDPs (cf. definition 6.2) contains the decision tree induction problem for supervised learning tasks (cf. definition **??**). This sparks the question: what kind of decision tree induction algorithm can we get using the MDP formalism? This is exactly what we study in the next part of this manuscript.

Those few chapters raise other interesting questions. We focused on non-parametric tree learning because RL algorithms can learn decision tree policies with potentially optimal interpretability-performance trade-offs through the reward of information gathering actions in (PO)IBMDPs (cf. definitions 3.4 and 3.6). However this comes at a cost of partial observability which makes learning difficult. Parametric tree policies on the other hand, can be computed with reinforcement learning directly in the base MDP. However existing RL algorithms for parametric decision tree policies (Silva et al., 2020; Vos & Verwer, 2024; Marton et al., 2025) require to re-train a policy entirely for each desired level of interpretability, i.e. each unique tree structure, future research in this direction should focus on algorithms for parametric tree policies that can re-use samples from one tree learning to train a different tree structure more efficiently. This would reduce the required quantity of a priori knowledge on the decision tree policy structure mentioned in section 2.

Attempting to overcome the partial observability challenges highlighted so far seems like a bad research direction. Indeed, while algorithms tailored specifically for the problem of learning deterministic partially observable policies for POIBMDPs might exist, we clearly saw that imitation learning was in practice a good alternative to direct interpretable reinforcement learning. Some limitations that we did not cover still exist such as how to choose good candidates information gathering actions or simply how to choose ζ for a target interpretability-performance trade-off.

References

Araya-López, M., Thomas, V., Buffet, O., and Charpillet, F. 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.

Atrey, A., Clary, K., and Jensen, D. 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.

Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *International conference on machine learning*, pp. 263–272. PMLR, 2017.

Baisero, A. and Amato, C. 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.

Baisero, A., Daley, B., and Amato, C. Asymmetric DQN for partially observable reinforcement learning. In Cussens, J. and Zhang, K. (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.

Barto, A. G., Sutton, R. S., and Anderson, C. W. 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.

Bastani, O., Pu, Y., and Solar-Lezama, A. Verifiable reinforcement learning via policy extraction. 2018.

Bellman, R. *Dynamic Programming*. 1957.

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

Breiman, L., Friedman, J., Olshen, R., and Stone, C. *Classification and Regression Trees*. Wadsworth, 1984.

Choi, S. P.-M., Zhang, N. L., and Yeung, D.-Y. Solving hidden-mode markov decision problems. In Richardson, T. S. and Jaakkola, T. S. (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.

Delfosse, Q., Shindo, H., Dhami, D. S., and Kersting, K. Interpretable and explainable logical policies via neurally guided symbolic abstraction. *Advances in Neural Information Processing (NeurIPS)*, 2023.

Delfosse, Q., Blüml, J., Gregori, B., Sztwiertnia, S., and Kersting, K. OCAtari: Object-centric Atari 2600 reinforcement learning environments. *Reinforcement Learning Journal*, 1:400–449, 2024a.

Delfosse, Q., Sztwiertnia, S., Rothermel, M., Stammer, W., and Kersting, K. Interpretable concept bottlenecks to align reinforcement learning agents. 2024b. URL https://openreview.net/forum?id=ZC0PSk6Mc6.

Demirovic, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., and Stuckey, P. J. 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.

Demirović, E., Hebrard, E., and Jean, L. Blossom: an anytime algorithm for computing optimal decision trees. *Proceedings of the 40th International Conference on Machine Learning*, 202:7533–7562, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/demirovic23a.html.

Glanois, C., Weng, P., Zimmer, M., Li, D., Yang, T., Hao, J., and Liu, W. A survey on interpretable reinforcement learning. *Machine Learning*, pp. 1–44, 2024.

Jaakkola, T., Singh, S. P., and Jordan, M. I. 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.

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

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

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

Littman, M. L. 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.

Loch, J. and Singh, S. P. 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.

Lundberg, S. M. and Lee, S.-I. 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.

Mania, H., Guy, A., and Recht, B. Simple random search of static linear policies is competitive for reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 1805–1814, Red Hook, NY, USA, 2018. Curran Associates Inc.

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

Marton, S., Grams, T., Vogt, F., Lüdtke, S., Bartelt, C., and Stuckenschmidt, H. Mitigating information loss in tree-based reinforcement learning via direct optimization.

2025. URL https://openreview.net/forum?id=qpXctF2aLZ.

Milani, S., Topin, N., Veloso, M., and Fang, F. 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.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Murthy, S. and Salzberg, S. 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.

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

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. Asymmetric actor critic for image-based robot learning, 2017. URL https://arxiv.org/abs/1710.06542.

Puri, N., Verma, S., Gupta, P., Kayastha, D., Deshmukh, S., Krishnamurthy, B., and Singh, S. 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.

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

Quinlan, J. R. Induction of decision trees. *Mach. Learn.*, 1 (1):81–106, 1986.

Quinlan, J. R. C4. 5: Programs for machine learning. *Morgan Kaufmann google schola*, 2:203–228, 1993.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Ribeiro, M. T., Singh, S., and Guestrin, C. "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.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shi, W., Huang, G., Song, S., Wang, Z., Lin, T., and Wu, C. 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.

Shindo, H., Delfosse, Q., Dhami, D. S., and Kersting, K. Blendrl: A framework for merging symbolic and neural policy learning. *arXiv*, 2025.

Sigaud, O. and Buffet, O. *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.

Silva, A., Gombolay, M., Killian, T., Jimenez, I., and Son, S.-H. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In Chiappa, S. and Calandra, R. (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.

Singh, S. P., Jaakkola, T. S., and Jordan, M. I. 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.

Sondik, E. J. 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.

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

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Solla, S., Leen, T., and Müller, K. (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.

Tesauro, G. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995. ISSN 0001-0782. doi: 10.1145/203330.203343. URL https://doi.org/10.1145/203330.203343.

Topin, N., Milani, S., Fang, F., and Veloso, M. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:9923–9931, 2021.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

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

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. Programmatically interpretable reinforcement learning. pp. 5045–5054, 2018.

Verwer, S. and Zhang, Y. Learning optimal classification trees using a binary linear program formulation. *Proceedings of the AAAI conference on artificial intelligence*, 33: 1625–1632, 2019.

Vos, D. and Verwer, S. 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.

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

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Wu, H., Isac, O., Zeljić, A., Tagomori, T., Daggitt, M., Kokke, W., Refaeli, I., Amir, G., Julian, K., Bassan, S., Huang, P., Lahav, O., Wu, M., Zhang, M., Komendantskaya, E., Katz, G., and Barrett, C. Marabou 2.0: A versatile formal analyzer of neural networks, 2024. URL https://arxiv.org/abs/2401.14461.

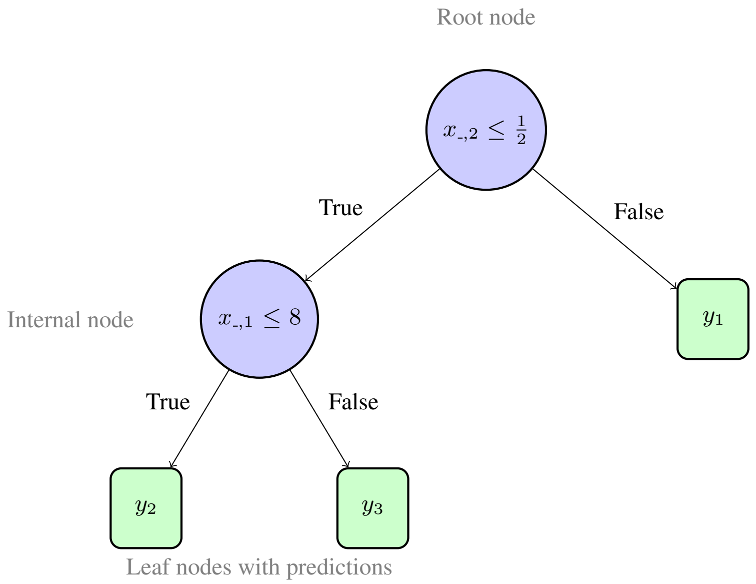


Figure 7. A generic depth 2 decision tree with 2 nodes and 3 leaves. The root node applies the test $1_{\{x_{\cdot,1} \leq \frac{1}{2}\}}$ to check if the first features of data is below $\frac{1}{2}$. Edges represent the outcomes of the tests in each internal nodes(True/False), and leaf nodes contain predictions $y_l \in \mathcal{Y}$. For any input $\boldsymbol{x}_i$, the tree defines a unique path from root to leaf.

A. Appendix for technical perliminaries

A.1. What are decision trees?

As the reader might have already guessed, we will put great emphasis on decision tree models as a means to study interpretability. While other interpretable models might have other properties than the ones we will highlight through this thesis, one conjecture from (Glanois et al., 2024) is that interpretable models are all hard to optimize or learn because they are non-differentiable in nature. This is something that will be key in our study of decision tree models that we introduce next and that we illustrate in figure 8.

Definition A.1 (Decision tree)**.** A decision tree is a rooted tree $T = (\mathcal{N}, E)$. Each internal node $\nu \in \mathcal{N}$ is associated with a test that maps input features $x_{ij} \in \mathcal{X}$ to a Boolean. 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 a prediction $y_l \in \mathcal{Y}$, where $\mathcal{Y}$ is the output space. For any input $x \in \mathcal{X}$, the tree defines a unique path from root to leaf, determining the prediction $T(x) = y_l$ where l is the reached leaf. The depth of a tree is the maximum path length from root to any leaf.

A.2. Reinforcement learning of approximate solutions to MDPs

When the MDP transition function and reward function are unknown, one can use reinforcement learning algorithms–also known as agents–to learn values or policies maximizing the RL objective. Reinforcement learning algorithms popularized by Richard Sutton (Sutton & Barto, 1998) don't **compute** an optimal policy but rather **learn** an approximate one based on sequences of transitions $(s_t, a_t, r_t, s_{t+1})_t$. RL algorithms usually fall into two categories: value-based (Sutton & Barto, 1998) and policy search (Sutton et al., 1999). Examples of these approaches are shown in algorithms 3, **??** and **??**. Q-learning and Sarsa compute an approximation of $Q^\star$ (cf. definition **??**) using temporal difference learning (Sutton & Barto, 1998). Q-learning is *off-policy*: it collects new transitions with a random policy, e.g. epsilon-greedy. Sarsa is *on-policy*: it collects new transitions greedily w.r.t. the current Q-values estimates. Policy gradient algorithms (Sutton et al., 1999) leverage the policy gradient theorem to approximate $\pi^\star$.

Q-learning, Sarsa, and policy gradients algorithms are known to converge to the optimal value or (locally) optimal policy under some conditions. There are many other ways to learn policies such as simple random search (Mania et al., 2018) or model-based reinforcement learning that estimates MDP transitions and rewards before applying e.g. value iteration (Azar et al., 2017). Those RL algorithms–also known as tabular RL because they represent policies as tables with $|S| \times |A|$ entries–are limited to small state spaces. To scale to large state spaces, it is common to use a neural network to represent policies or values (Tesauro, 1995). In the next section, we present deep reinforcement learning algorithms designed specifically for neural networks.

Algorithm 3 Q-Learning (Watkins & Dayan, 1992)

```
Data: MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, learning rate $\alpha$, exploration rate $\epsilon$
Result: Policy $\pi$
Initialize $Q(s, a) = 0$ for all $s \in S, a \in A$
 Initialize state $s_0 \sim T_0$
 for each step $t$ do
    Choose action $a_t$ using e.g. $\epsilon$-greedy policy: $a_t = \text{argmax}_a Q(s_t, a)$ with prob. $1 - \epsilon$
     Take action $a_t$, observe $r_t = R(s_t, a_t)$ and $s_{t+1} \sim T(s_t, a_t)$
     $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$
     $s_t \leftarrow s_{t+1}$
end
$\pi(s) = \text{argmax}_a Q(s, a)$ // Extract greedy policy
```

A.3. Deep reinforcement learning

Reinforcement learning has also been successfully combined with function approximations to solve MDPs with large discrete state spaces or continuous state spaces ($S \subset \mathbb{R}^p$ in definition 3.1). In the rest of this manuscript, unless stated otherwise, we write $\boldsymbol{s}$ a state vector in a continuous state space[1].

Deep Q-Networks (DQN) (Mnih et al., 2015), described in algorithm **??** achieved super-human performance on a set of Atari games. Authors successfully extended the Q-learning (cf. algorithm 3) to the function approximation setting by introducing target networks to mitigate distributional shift in the temporal difference error and replay buffer to increase sample efficiency.

Proximal Policy Optimization (PPO) (Schulman et al., 2017), described in algorithm **??**, is an actor-critic algorithm (Sutton & Barto, 1998) optimizing a neural network policy. In actor-critic algorithms, cumulative discounted rewards starting from a particular state, also known as *the returns*, are also estimated with a neural network. PPO is known to work well in a variety of domains including robot control in simulation among others.

A.4. Imitation learning: a baseline (indirect) interpretable reinforcement learning method

Unlike PPO or DQN for neural networks, there exists no algorithm that trains decision tree policies to optimize the RL objective (cf. definition 3.2). In fact, we will show in the first part of the manuscript that training decision trees that optimize the RL objective is very difficult.

Hence, many interpretable reinforcement learning approaches first train a neural network policy–also called an expert policy–to optimize the RL objective (cf. definition 3.2) using e.g. PPO, and then fit a student policy such as a decision tree using CART (cf. algorithm **??**) to optimize the supervised learning objective (cf. definition **??**) with the neural policy actions as targets. This approach is known as imitation learning and is essentially training a student policy to optimize the objective:

Definition A.2 (Imitation learning objective)**.** Given an MDP $\mathcal{M}$ (cf. definition 3.1), an expert policy π^* and a policy class Π, e.g. decision trees of depth at most 3, the imitation learning objective is to find a student policy $\hat{\pi} \in \Pi$ that minimizes the expected action disagreement with the expert:

$$IL(\pi) = \mathbb{E}_{\boldsymbol{s} \sim \rho(\boldsymbol{s})} \left[\mathcal{L}(\pi(\boldsymbol{s}), \pi^*(\boldsymbol{s})) \right] \tag{1}$$

where $\rho(\boldsymbol{s})$ is the state distribution in $\mathcal{M}$ induced by the student policy π and $\mathcal{L}$ is a loss function measuring the disagreement between the student policy's action $\pi(s)$ and the expert's action $\pi^*(s)$.

There are two main imitation learning methods used for interpretable reinforcement learning. Dagger (cf. algorithm **??**) is a straightforward way to fit a decision tree policy to optimize the imitation learning objective (cf. definition A.2). VIPER

[1]Note that discrete states can be one-hot encoded as state vectors in $\{0, 1\}^{|S|}$.

(cf. algorithm **??**) was designed specifically for interpretable reinforcement learning. VIPER re-weights the transitions collected by the neural network expert by a function of the state-action value (cf. definition **??**). The authors of VIPER showed that decision tree policies fitted with VIPER tend to have the same RL objective value as Dagger trees while being more interpretable (shallower or with fewer nodes) and sometimes outperform Dagger trees. Dagger and VIPER are two strong baselines for decision tree learning in MDPs, but they optimize a surrogate objective only, even though in practice the resulting decision tree policies often achieve high RL objective value. We use these two algorithms extensively throughout the manuscript. Next we show how to learn a decision tree policy for the example MDP (cf. figure **??**).

A.5. Your first decision tree policy

Now the reader should know how to train decision tree classifiers or regressors for supervised learning using CART (cf. section **??**). The reader should also know what an MDP is and how to compute or learn policies that optimize the RL objective (cf. definition 3.2) with (deep) reinforcement learning (cf. section **??**). Finally, the reader should now know how to obtain a decision tree policy for an MDP through imitation learning (cf. definition A.2) by first using RL to get an expert policy and then fitting a decision tree to optimize the supervised learning objective, using the expert actions as labels.

In this section we present the first decision tree policies of this manuscript obtained using Dagger or VIPER after learning an expert Q-function for the grid world MDP from figure **??** using Q-learning (cf. algorithm 3). Recall the optimal policies for the grid world, taking the green actions in each state in figure **??**. 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 9, 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. Indeed, figure **??** shows that the optimal depth-1 tree achieves exactly the same RL objective value as the optimal policies from figure **??**, independently of the discount factor γ.

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 **??**?

We start by running the standard Q-learning algorithm as presented in algorithm 3 with $\epsilon = 0.3$, $\alpha = 0.1$ over 10,000 time steps. The careful reader might wonder how ties are broken in the argmax operation from algorithm 3. While Sutton and Barto break ties by index value in their book (Sutton & Barto, 1998) (the greedy action is the 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 10, 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 10, 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 (cf. figure **??**) to directly search for policies with good interpretability-performance trade-offs with respect to the original RL objective.

A.6. Example: an IBMDP for a grid world

We re-formulate the example MDP (example **??**) as an MDP with a finite number of vector valued states (x, y-coordinates). The states are $S = \{(0.5, 0.5), (0.5, 1.5), (1.5, 1.5), (1.5, 0.5)\} \subsetneq [0, 2] \times [0, 2]$. The actions are the cardinal directions

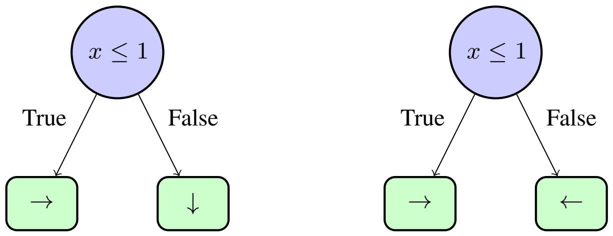


Figure 8. Left, an optimal depth-1 decision tree policy. On the right, a sub-optimal depth-1 decision tree policy.

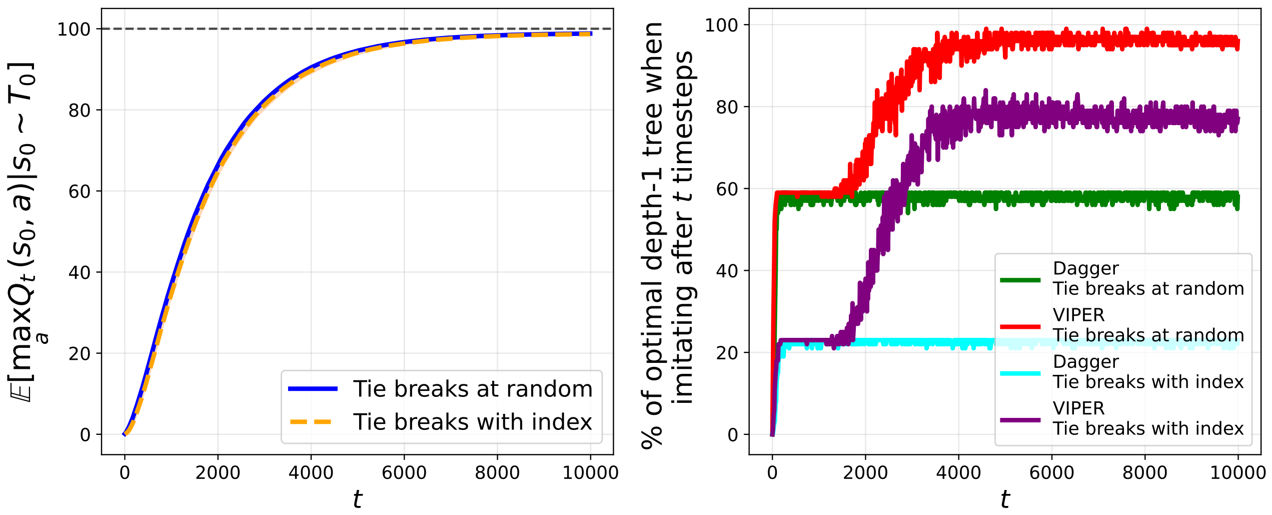


Figure 9. Left, sample complexity curve of Q-learning with default hyperparameters on the 2×2 grid world MDP over 100 random seeds. Right, performance of indirect interpretable methods when imitating the greedy policy with a tree at different Q-learning stages.

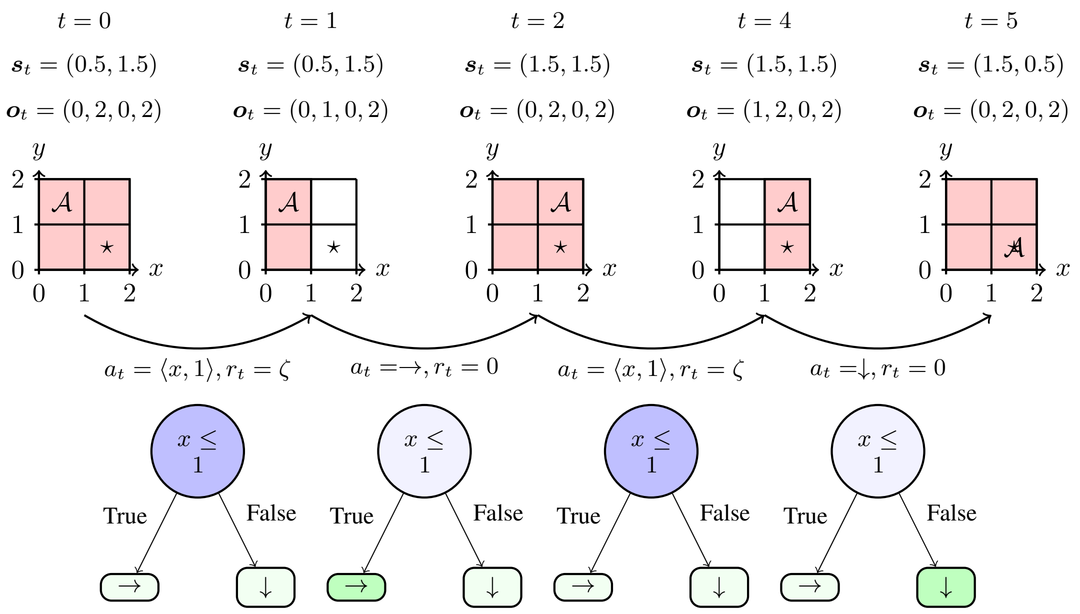

$t = 0$
$\boldsymbol{s}_t = (0.5, 1.5)$
$\boldsymbol{o}_t = (0, 2, 0, 2)$
$t = 1$
$\boldsymbol{s}_t = (0.5, 1.5)$
$\boldsymbol{o}_t = (0, 1, 0, 2)$
$t = 2$
$\boldsymbol{s}_t = (1.5, 1.5)$
$\boldsymbol{o}_t = (0, 2, 0, 2)$
$t = 4$
$\boldsymbol{s}_t = (1.5, 1.5)$
$\boldsymbol{o}_t = (1, 2, 0, 2)$
$t = 5$
$\boldsymbol{s}_t = (1.5, 0.5)$
$\boldsymbol{o}_t = (0, 2, 0, 2)$
$\mathcal{A}$
$\star$
$a_t = \langle x, 1\rangle, r_t = \zeta$
$a_t = \rightarrow, r_t = 0$
$a_t = \langle x, 1\rangle, r_t = \zeta$
$a_t = \downarrow, r_t = 0$
$x \leq 1$
True
False
$\rightarrow$
$\downarrow$

Figure 10. An IBMDP trajectory when the base MDP is 2×2 grid world. In the top row, we write the visited base state features and observations, in the middle row, we graphically represent those, and in the bottom row, we present the corresponding decision tree policy traversal. $\mathcal{A}$ tracks the current state features $\boldsymbol{s}_t$ in the grid. The pink obstructions of the grid represent the current observations $\boldsymbol{o}_t$ of the base state features. When the pink covers the whole grid, the information contained in the observation 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 base state features get. At $t = 0$, the base state features are $\boldsymbol{s}_0 = (0.5, 1.5)$. The initial observation is always the base MDP default state feature bounds, here $\boldsymbol{o}_0 = (0, 2, 0, 2)$ because the base state features are in $[0, 2] \times [0, 2]$. This means that the 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 base state against the value 1 and the reward ζ. 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 base state is below 1, the observation is updated with the refined bounds $\boldsymbol{o}_1 = (0, 1, 0, 2)$, i.e. the pink area shrinks, and the base state features remain unchanged. The agent then takes a base action that is to move right. This gives a reward 0, resets the observation to the original base state feature bounds, and changes the features to $\boldsymbol{s}_2 = (1.5, 1.5)$. And the trajectory continues like this until the absorbing base state $\boldsymbol{s}_5 = (1.5, 0.5)$ is reached.

$A = \{\rightarrow, \leftarrow, \downarrow, \uparrow\}$ that shift the states by one as long as the coordinates remain in the grid. The reward for taking any action is 0 except when in the bottom right state $(1.5, 0.5)$ which is an absorbing state: once in this state, you stay there forever. Standard optimal deterministic Markovian policies were presented for this MDP in example **??**.

Suppose an associated IBMDP (definition 3.4) with two IGAs:

- $\langle x, 1\rangle$ that tests if $x \leq 1$

- $\langle y, 1\rangle$ that tests if $y \leq 1$

The initial observation is always the grid bounds $\boldsymbol{o}_0 = (0, 2, 0, 2)$ because the base state features in the grid world are always in $[0, 2] \times [0, 2]$. There are only finitely many observations since with those two IGAs there are only nine possible observations that can be attained from $\boldsymbol{o}_0$ following the IBMDP transitions (cf. definition 3.4). For example when the IBMDP initial base state features are $\boldsymbol{s}_0 = (0.5, 1.5)$, and taking first $\langle x, 1\rangle$ then $\langle y, 1\rangle$ the corresponding observations are first $\boldsymbol{o}_{t+1} = (0, 1, 0, 2)$ and then $\boldsymbol{o}_{t+2} = (0, 1, 1, 2)$. The full observation set is $O = \{(0, 2, 0, 2), (0, 1, 0, 2), (0, 2, 0, 1), (0, 1, 0, 1), (1, 2, 0, 2), (1, 2, 0, 1), (1, 2, 1, 2), (0, 1, 1, 2), (0, 2, 1, 2)\}$. The transitions and rewards are given in definition (cf. definition 3.4).

In figure 11 we illustrate a trajectory in this IBMDP.

B. 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 policies with similar RL objective values (cf. definition 3.2) for the CartPole. On the other hand, we find that, imitation learning, despite not directly optimizing the RL objective for CartPole, outperforms deep RL that optimizes the interpretable RL objective (cf. definition **??** in which the objective trades off the standard RL objective and interpretability).

Authors of (Topin et al., 2021) use two deep reinforcement learning baselines (cf. section **??**) to which they apply some modifications in order to learn partially observable policies as required by proposition **??** and by the interpretable RL objective (cf. definition **??**). Authors modify the standard DQN (cf. algorithm **??**) to return a partially observable 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 4. Similar modifications are applied to the standard PPO (cf. algorithm **??**) that we present in the appendix (cf. algorithm **??**). 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.

B.1. Experimental setup

B.1.1. (IB)MDP

We use the exact same base MDP and associated IBMDPs for our experiments as (Topin et al., 2021) except when mentioned otherwise.

Base MDP The task at hand is to optimize the RL objective (cf. definition 3.2) 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 (cf. definition 3.4) with $\zeta = -0.01$ and 4 information gathering actions. In appendix **??**, we give more details about how the authors of the original IBMDP paper chose the 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.

Algorithm 4 Modified Deep Q-Network. We highlight in green the changes to the standard DQN (cf. algorithm **??**).

```
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: Partially observable deterministic 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 base state features $\boldsymbol{s}_0 \sim T_0$
    Initialize observation $\boldsymbol{o}_0 = (L_1, U_1, \dots, L_p, U_p)$

    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
```

B.1.2. BASELINES

Modified DQN and Modified PPO as mentioned above, the authors use the modified version of DQN from algorithm 4. 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 (ϵ-greedy with linearly decreasing value ϵ 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 ϵ value to 0.9). We use the same optimizer (RMSprop with hyperparameter 0.95 and learning rate 2.5×10^{-4}) to update the Q-networks. Authors did not share which DQN implementation they used so we use the stable-baselines3 one (Raffin et al., 2021). Authors did not share which activation functions they used so we try both $\tanh$ and relu. For the modified PPO algorithm (cf. algorithm **??**), 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 **??**, **??** and **??**.

Indirect methods We also compare modified RL algorithm to imitation learning (cf. section A.4). To do so, we use VIPER or Dagger (cf. algorithms **??** and **??**) 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 (cf. algorithm **??**) implementation from scikit-learn (Pedregosa et al., 2011) with default hyperparameters and maximum depth of 2 like in (Topin et al., 2021).

B.1.3. METRICS

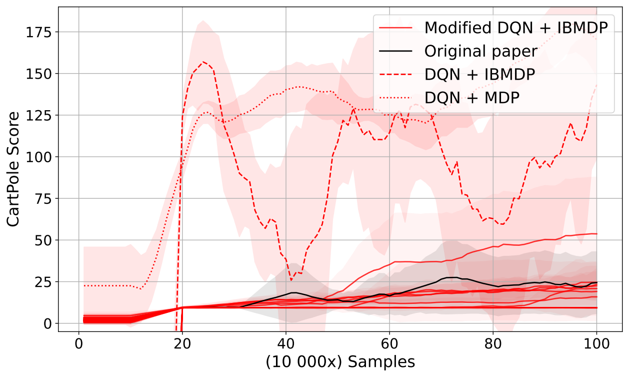
The key metric of this section is performance when controlling the CartPole, i.e, the average *undiscounted* cumulative reward of a policy on 100 trajectories (RL objective with $\gamma = 1$). For modified RL algorithms that learn a partially observable policy (or Q-function) in an IBMDP, we periodically extract the policy (or Q-function) and use algorithm 1 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 base 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 base MDP is given, i.e. when $a_t \in A$ in the IBMDP (cf. definition 3.4)). 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 base 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).

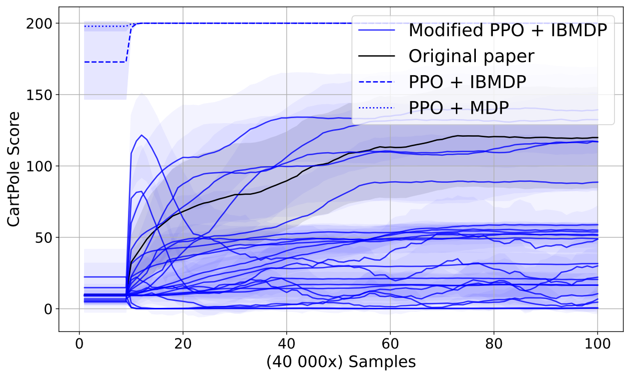
B.2. Results

B.2.1. HOW WELL DO MODIFIED DEEP RL BASELINES LEARN IN IBMDPS?

On figure 12a, 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 in average–the curves flatten at the end of the



(a) DQN variants



(b) PPO variants

Figure 11. Comparison of modified reinforcement learning algorithms on different CartPole IBMDPs. (a) Shows variations of modified DQN and DQN (cf. table **??**), while (b) shows variations of modified PPO and PPO (cf. table **??**). 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.

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 12b, 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 12a and 12b, we observe that standard RL baselines (RL + IBMDP and RL + MDP), learn better CartPole policies in average than their modified counterparts that learn partially observable policies (cf. proposition **??**). On figure 12b, 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.

B.2.2. Which decision tree policies does direct reinforcement learning return for the CartPole MDP?

On figure 13, 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 (cf. definition A.2) 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 13 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 14, we present the best decision tree policies for CartPole returned by modified DQN and modified PPO. We used algorithm 1 to extract 20 trees from the 20 partially observable 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 base action forever or repeat the same IGA (cf. definition 3.4) forever.

We have shown that compared to learning non-interpretable but standard Markovian neural network policies for the base MDP or some associated IBMDP, reinforcement learning of partially observable policies in IBMDP is less efficient (cf.

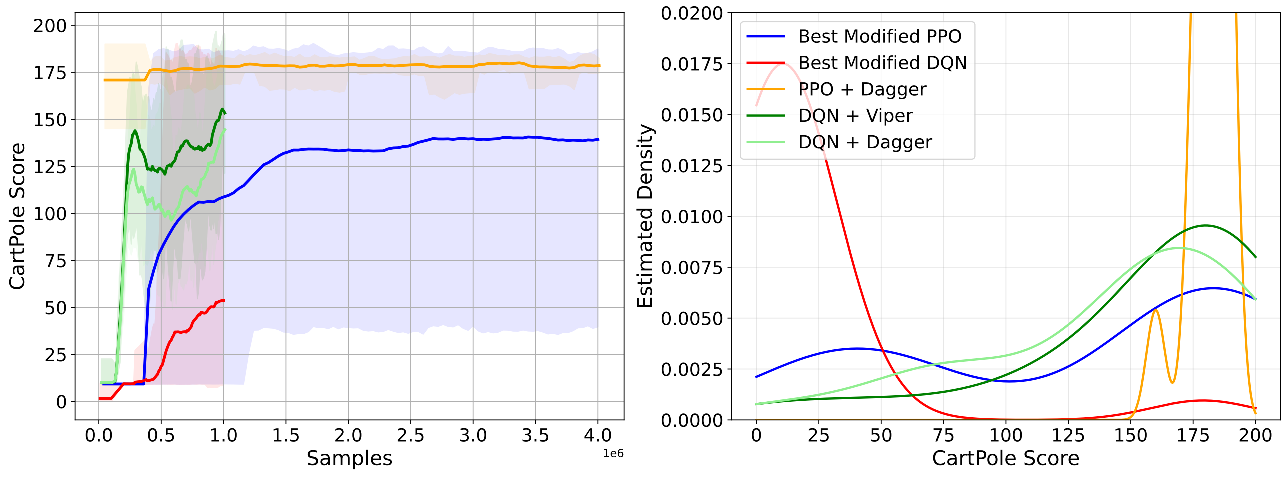


Figure 12. (left) Mean performance of the best–w.r.t. the RL objective for CartPole–modified RL + IBMDP combination. Shaded areas represent the min and max performance over the 20 seeds during training. (right) Corresponding score distribution of the final decision tree policies w.r.t. the RL objective for CartPole.

figures 12a and 12b). As a consequence, only a handful of modified RL runs are able to learn decision tree policies that are on par with imitated trees (cf. figure 13). In the next chapter, we highlight the connections between direct interpretable RL (cf. definition **??**) and POMDPs to get insights on the hardness of direct reinforcement learning of decision trees.

From the previous chapter, we know that to directly learn decision tree policies that optimize the RL objective (cf. definition 3.2) for an MDP (cf. definition 3.1), one can learn a deterministic partially observable policy that optimizes the interpretable RL objective (cf. definition **??**) in an IBMDP (cf. definition 3.4 and proposition **??**). Such problems are classical instances of partially observable Markov decision processes (POMDPs) (Sondik, 1978; Sigaud & Buffet, 2013). This connection with POMDPs brings novel insights to direct reinforcement learning of decision tree policies. In this chapter, all the decision processes have a finite number of vector-valued states and observations. Hence we will use bold fonts for states and observations but we can still use summations rather than integrals when required.

C. RL objective values

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 base 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 base 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 $\boldsymbol{s}_g$:

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

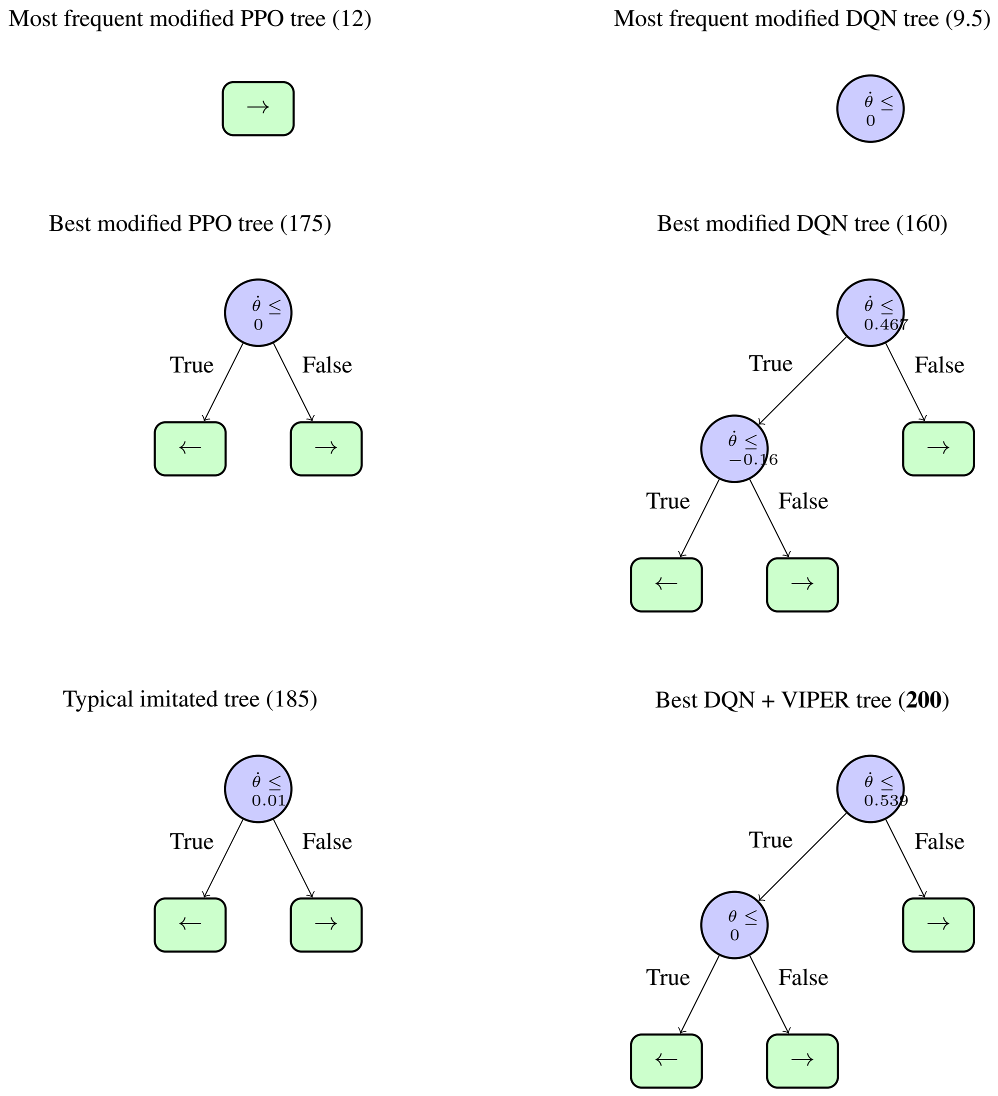

Most frequent modified PPO tree (12)
→
Most frequent modified DQN tree (9.5)
$\dot{\theta} \leq 0$
Best modified PPO tree (175)
$\dot{\theta} \leq 0$
True
False
←
→
Best modified DQN tree (160)
$\dot{\theta} \leq 0.467$
True
False
$\dot{\theta} \leq -0.16$
→
True
False
←
→
Typical imitated tree (185)
$\dot{\theta} \leq 0.01$
True
False
←
→
Best DQN + VIPER tree (**200**)
$\dot{\theta} \leq 0.539$
True
False
$\theta \leq 0$
→
True
False
←
→

Figure 13. Trees obtained by modified deep RL in IBMDPs against trees obtained with imitation (RL objective value). θ and $\dot{\theta}$ are respectively the angle and the angular velocity of the pole

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 base states at initialization given that the observation is $\boldsymbol{o}_0$ is simply the probability of being in any base 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 base 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 base 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 base 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:

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

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

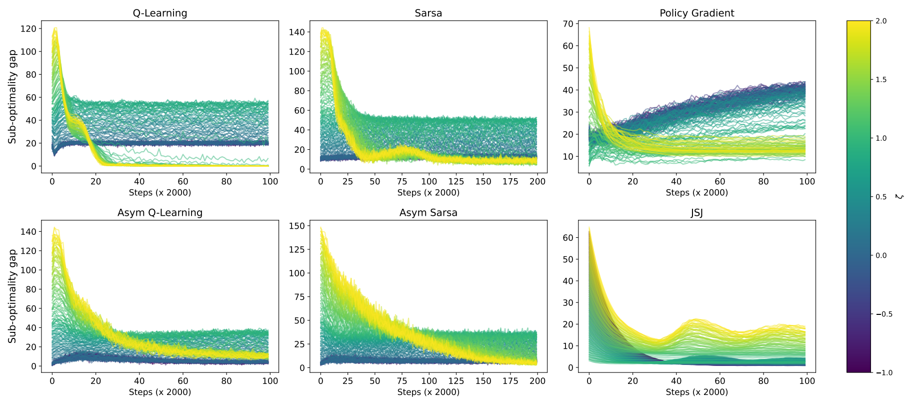


Figure 14. (Asymmetric) reinforcement learning in POIBMDPs. In each subplot, each single line is colored by the value of ζ in the corresponding POIBMDP in which learning occurs. Each single learning curve represent the sub-optimality gap averaged over 100 seeds.

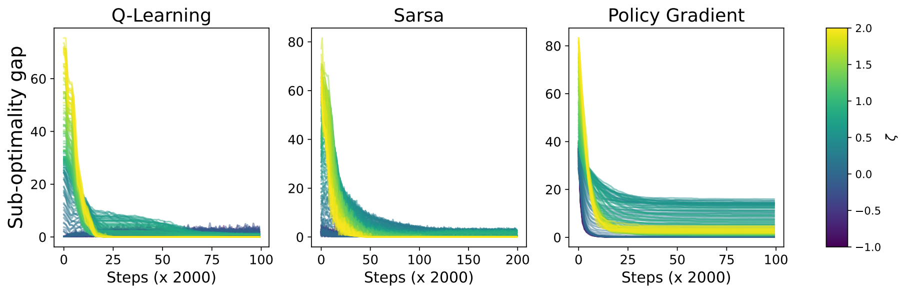


Figure 15. We reproduce the same plot as in figure 15 for classification POIBMDPs. Each individual curve is the sub-optimality gap of the learned policy during training averaged over 100 runs for a single ζ value.

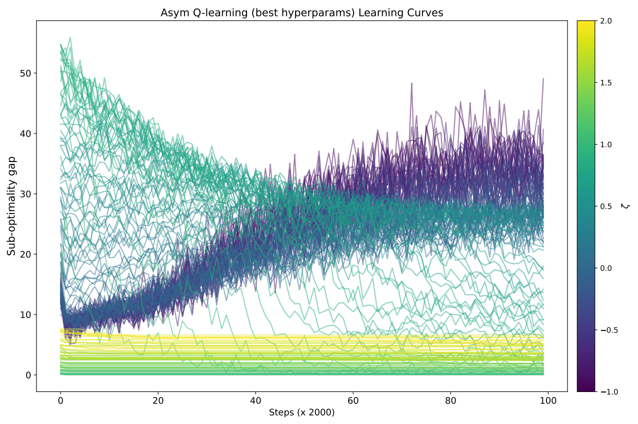
$$\begin{aligned}
J(\mathcal{T}_{\text{stoch}}) &= \frac{1}{4}(V_G^{\text{stoch}} + V_{S_0}^{\text{stoch}} + V_{S_1}^{\text{stoch}} + V_{S_2}^{\text{stoch}}) \\
&= \frac{1}{4}\left(\frac{1}{1-\gamma} + 2\cdot\frac{\gamma(\frac{1}{2}+\frac{1}{4}\gamma)}{(1-\frac{1}{4}\gamma^2)(1-\gamma)} + \frac{\gamma(\frac{1}{4}\gamma+\frac{1}{2})}{(1-\frac{1}{4}\gamma^2)(1-\ga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

D. Training curves

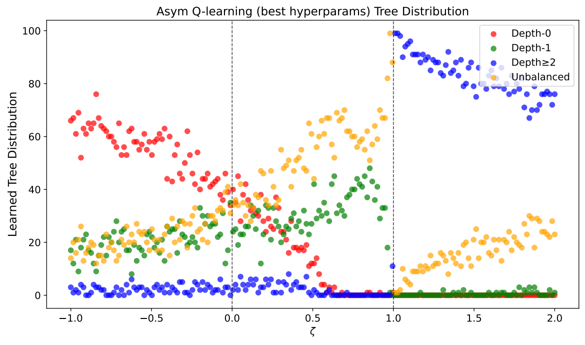
E. Hyperparameters

Table 2. 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



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



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

Figure 16. Analysis of the top-performing asymmetric Q-learning instantiation. (left) Learning curves, and (right) tree distributions across different POIBMDP configurations.