
Limits of reinforcement learning for decision trees in Markov decision processes

Anonymous Authors¹

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 (Ghois 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 construc-

tion. 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 solving 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

¹Anonymous 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.

055
 056 equivalent to learning a deterministic memoryless policy for
 057 partially observable MDP (POMDP)([Sondik, 1978](#))—which
 058 is a hard problem ([Littman, 1994](#))—and show that this might
 059 be the main reason for failures. In section ??, we further
 060 support this claim by constructing special instances of such
 061 POMDPs where the observations contain all the information
 062 about hidden states, and show that in those cases, direct
 063 reinforcement learning of decision trees works well.
 064

2. Learning decision tree policies for MDPs

066 There exist reinforcement learning algorithms that directly
 067 learn decision tree policies optimizing the cumulative re-
 068wards in a given MDP. These approaches can be broadly di-
 069 vided into methods based on *parametric* and *non-parametric*
 070 trees.
 071

072 Parametric decision trees fix the tree structure *a priori*—including depth, node arrangement, and selected state
 073 features—and only learn the decision thresholds. This for-
 074 mulation enables differentiability and allows direct opti-
 075 mization of the RL objective using policy gradient meth-
 076 ods ([Sutton et al., 1999](#)). Several works ([Silva et al., 2020](#);
 077 [Vos & Verwer, 2024](#); [Marton et al., 2025](#)) employ PPO to
 078 train such differentiable trees. While these methods can
 079 achieve strong performance, they require the tree structure
 080 to be specified in advance, making it difficult to adaptively
 081 trade off interpretability and performance. An overly com-
 082 plex structure may require post-hoc pruning, whereas an
 083 insufficiently expressive structure may fail to represent good
 084 policies. Moreover, ([Marton et al., 2025](#)) reports that addi-
 085 tional stabilization techniques, such as adaptive batch sizes,
 086 are often necessary for direct learning in order to outperform
 087 indirect imitation methods such as VIPER.
 088

089 Non-parametric decision trees, by contrast, are the stan-
 090 dard model in supervised learning, where greedy algo-
 091 rithms ([Breiman et al., 1984](#); [Quinlan, 1986](#); [1993](#)) effi-
 092 ciently construct trees that balance predictive performance
 093 and interpretability. However, their extension to reinfor-
 094 cements learning remains largely unexplored. To the best of
 095 our knowledge, the only work that studies learning non-
 096 parametric decision trees that optimize a trade-off between
 097 interpretability and cumulative rewards is that of Topin et
 098 al. ([Topin et al., 2021](#)). Topin et al. introduce *iterative*
 099 *bounding MDPs* (IBMDPs), which augment a base MDP
 100 with additional state features, actions, rewards, and transi-
 101 tions. They show that certain policies in the IBMDP cor-
 102 respond to decision tree policies for the base MDP. Hence,
 103 standard RL algorithms can be used to learn such policies
 104 in IBMDPs.
 105

106 Finally, a few specialized methods exist for restricted prob-
 107 lem classes. For maze-like MDPs, ([Mansour et al., 2022](#))
 108 proves the existence of optimal decision tree policies and
 109

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 and problems

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 $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$ where:

- 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.
- $R : S \times A \rightarrow \mathbb{R}$ 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.
- $T : S \times A \rightarrow \Delta(S)$ 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. For example, in cancer treatment, the best outcome is to eliminate the patient’s tumor as quickly as possible. We can formally define this objective, that we call the reinforcement learning objective, as follows:

Definition 3.2 (Reinforcement learning objective). Given an MDP $\mathcal{M} \equiv \langle S, A, R, T, T_0 \rangle$ (cf. definition 3.2), the goal of reinforcement learning for sequential decision making is to find a model, also known as a policy, $\pi : S \rightarrow A$ that maximizes the expected discounted sum of rewards:

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

110 where $0 < \gamma \leq 1$ is the discount factor that controls the
 111 trade-off between immediate and future rewards.

112 Algorithms presented in this manuscript aim to find an optimal policy $\pi^* \in \arg\max_{\pi} J(\pi)$ that maximizes the above
 113 reinforcement learning (RL) objective.
 114

117 3.2. Exact solutions for Markov decision problems

118 We begin with the planning setting in which the MDP transitions and rewards (cf. definition 3.2) are known. Leveraging the Markov property, one can use dynamic programming (Bellman, 1957) to compute optimal policies with respect to the RL objective (cf. definition 3.3). Algorithms based on dynamic programming use the notion of *value* of states and actions:

126 **Definition 3.3** (Value of a state). In an MDP \mathcal{M} (cf. definition 3.2), 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 :

$$131 V^\pi(s) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_t = \pi(s_t), s_{t+1} \sim T(s_t, a_t) \right]$$

134 Applying the Markov property gives a recursive definition
 135 of the value of s under policy π :

$$137 V^\pi(s) = R(s, \pi(s)) + \gamma \mathbb{E}[V^\pi(s') \mid s' \sim T(s, \pi(s))]$$

138 **Definition 3.4** (Optimal value of a state). The optimal value
 139 of a state $s \in S$, $V^*(s)$, is the value of state s when following the optimal policy π^* (the policy that maximizes the RL
 140 objective (cf. definition 3.3)).

$$143 V^*(s) = V^{\pi^*}(s)$$

144 **Definition 3.5** (Optimal value of a state–action pair). The
 145 optimal value of a state–action pair $(s, a) \in S \times A$, $Q^*(s, a)$,
 146 is the value when taking action a in state s and then following
 147 the optimal policy.

$$149 Q^*(s, a) = R(s, a) + \gamma \mathbb{E}[V^*(s') \mid s' \sim T(s, a)]$$

150 More realistically, neither the transition function T nor the
 151 reward function R of the MDP are known, e.g. the doctor
 152 cannot **know** how the tumor and the patient’s health will
 153 change after a dose of chemotherapy, but can only **observe**
 154 the change. This distinction in available information parallels
 155 the distinction between dynamic programming and
 156 reinforcement learning, described next.

159 3.3. Iterative bounding Markov decision processes

160 The key thing to know about IBMDPs is that they are, as
 161 their name suggests, MDPs (cf definition 3.2). Hence, IB-
 162 MDPs admit an optimal deterministic Markovian policy that
 163

164 maximizes the RL objective. In this part we will assume that
 165 all the MDPs we consider are MDPs with continuous state
 166 spaces (cf. section ??) with a finite set of actions, so we
 167 use bold fonts for states and observations as they are vector-
 168 valued. However all our results generalize to discrete states
 169 (in \mathbb{Z}^m) MDPs that we can factor using one-hot encodings.
 170 Given an MDP for which we want to learn a decision tree
 171 policy—the base MDP–IBMDP states are concatenations of
 172 the base MDP state features and some observations. Those
 173 observations are information about the base state features
 174 that are refined—“iteratively bounded”—at each step. Those
 175 observations essentially represent some knowledge about
 176 where some base state features lie in the state space. Actions
 177 available in an IBMDP are: 1) the actions of the base MDP,
 178 that change base state features, and 2) *information gathering*
 179 actions that change the aforementioned observations. Now,
 180 base actions in an IBMDP are rewarded like in the base
 181 MDP, this ensures that the RL objective w.r.t. the base MDP
 182 is encoded in the IBMDP reward. When taking an information
 183 gathering action, the reward is an arbitrary value such
 184 that optimizing the RL objective in the IBMDP is equivalent
 185 to optimizing some trade-off between interpretability and
 186 the RL objective in the base MDP.

187 Before showing how to get decision tree policies from IB-
 188 MDP policies, we give a formal definition of IBMDPs fol-
 189 lowing Topin et. al. (Topin et al., 2021).

190 **Definition 3.6** (Iterative bounding Markov decision process). Given an MDP $\mathcal{M} \equiv \langle S, A, R, T, T_0 \rangle$ (cf. definition 3.2), an associated iterative bounding Markov decision process \mathcal{M}_{IB} is a tuple:

$$192 \begin{array}{ccc} \text{State space} & & \text{Reward function} \\ \overbrace{S \times O}^{\text{Action space}}, A \cup A_{info}, & \overbrace{(R, \zeta)}^{\text{Transitions}} & , \underbrace{(T_{info}, T, T_0)}_{\text{Transitions}} \end{array}$$

193 , where:

- S are the base MDP state features. Base state features $s = (s_1, \dots, s_p) \in S$ 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: $O \subsetneq S^2 = [L_1, U_1] \times \dots \times [L_p, U_p] \times [L_1, U_1] \times \dots \times [L_p, U_p]$. So the complete IBMDP state space is $S \times O$, the concatenations of base state features $s = (s_1, \dots, s_p) \in S$ and some observation $o = (L_1, U_1, \dots, L_p, U_p)$, an IBMDP state is

$$194 s_{IB} = (\underbrace{s_1, \dots, s_p}_{\text{base state features}}, \underbrace{L_1, U_1, \dots, L_p, U_p}_{\text{observation}}).$$

- A are the base MDP actions.

- 165 • A_{info} are *information gathering* actions (IGAs) of
 166 the form $\langle j, v \rangle$ where j is a state feature index $1 \leq$
 167 $j \leq p$ and v is a real number between L_j and U_j . So
 168 the complete action space of an IBMDP is the set of
 169 base MDP actions and information gathering actions
 170 $A \cup A_{info}$.
- 171 • $R : S \times A \rightarrow \mathbb{R}$ is the base MDP reward function.
- 172 • ζ is a reward signal for taking an information gathering
 173 action. So the IBMDP reward function is to get a
 174 reward from the base MDP if the action is a base MDP
 175 action or to get ζ if the action is an IGA action.
- 176 • $T_{info} : S \times O \times (A_{info} \cup A) \rightarrow \Delta(S \times O)$ is the
 177 transition function of IBMDPs: Given some observa-
 178 tion $\mathbf{o}_t = (L'_1, U'_1, \dots, L'_p, U'_p) \in O$ and base state
 179 features $\mathbf{s}_t = (s'_1, s'_2, \dots, s'_p)$ if an IGA $\langle j, v \rangle$ is taken,
 180 the new observation is:

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

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: $\mathbf{s}_{t+1} \sim T(\mathbf{s}_t, a_t)$. 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 $\mathbf{o}_0 = (L_1, U_1, \dots, L_p, U_p)$.

Now remains to extract a decision tree policy for MDP \mathcal{M} from a policy for an associated IBMDP \mathcal{M}_{IB} .

3.4. From policies to trees

One can notice that information gathering actions (cf. definition 5.1) resemble the Boolean functions $1_{\{x_{-j} \leq v\}}$ that make up internal decision tree nodes (cf. figure 1). Indeed, 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, they show that only deterministic policies depending solely on the observations of the IBMDP are guaranteed to correspond to decision tree policies for the base MDP.

Proposition 3.7 (Deterministic partially observable IBMDP policies are decision tree policies). *Given a base MDP*

Algorithm 1 Extract a Decision Tree Policy

Data: Deterministic partially observable policy π_{po} for IBMDP $\langle S \times O, A \cup A_{info}, (R, \zeta), (T_{info}, T, T_0) \rangle$ and IBMDP observation $\mathbf{o} = (L'_1, U'_1, \dots, L'_p, U'_p)$

Result: Decision tree policy π_T for MDP $\langle S, A, R, T, T_0 \rangle$

Function Subtree_From_Policy(\mathbf{o}, π_{po}):

```

 $a \leftarrow \pi_{po}(\mathbf{o})$ 
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
|  $\mathbf{o}_L \leftarrow \mathbf{o}; \mathbf{o}_R \leftarrow \mathbf{o}$ 
|  $\mathbf{o}_L \leftarrow (L'_1, U'_1, \dots, L'_j, v, \dots, L'_p, U'_p); \mathbf{o}_R \leftarrow (L'_1, U'_1, \dots, v, U'_j, \dots, L'_p, U'_p)$ 
| childL  $\leftarrow$  Subtree_From_Policy( $\mathbf{o}_L, \pi_{po}$ )
| childR  $\leftarrow$  Subtree_From_Policy( $\mathbf{o}_R, \pi_{po}$ )
| return Internal_Node(feature:  $i$ , value:  $v$ , children: (childL, childR))

```

$\mathcal{M} \langle S, A, R, T, T_0 \rangle$ and an associated IBMDP $\mathcal{M}_{IB} \langle S \times O, A \cup A_{info}, (R, \zeta), (T_{info}, T, T_0) \rangle$ (cf. definition 5.1), a deterministic partially observable policy $\pi_{po} : O \rightarrow A \cup A_{info}$ is a decision tree policy $\pi_T : S \rightarrow A$ for the base MDP \mathcal{M} .

Proof. (Sketch) algorithm 2 that takes as input a deterministic partially observable policy (cf. definition 5.2) for an IBMDP $\mathcal{M}_{IB} \langle S \times O, A \cup A_{info}, (R, \zeta), (T_{info}, T, T_0) \rangle$ (cf. definition 5.1), returns a decision tree policy π_T (cf. definition ??) for the base MDP $\mathcal{M} \langle S, A, R, T, T_0 \rangle$ and always terminates unless the deterministic partially observable policy takes IGAs in every state. \square

While the connections with partially observable MDPs (Sondik, 1978; Sigaud & Buffet, 2013) is obvious, we defer the implications to chapter ?? as this connection was absent from the original IBMDP paper (Topin et al., 2021). Next we present an IBMDP for the MDP of example 2.

4. Reproducing “Iterative Bounding MDPs: Learning Interpretable Policies via Non-Interpretable Methods”

We attempt to reproduce the results from (?)table 1]topin2021iterative 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 3.3) for the CartPole. On the other hand, we

220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274

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 5.2 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 $O \rightarrow \mathbb{R}^{|A \cup A_{info}|}$ rather than $S \times O \rightarrow \mathbb{R}^{|A \cup A_{info}|}$. In this modified DQN, the temporal difference error target for the Q -function $O \rightarrow A \cup A_{info}$ is approximated by a neural network $S \times O \rightarrow A \cup A_{info}$ that is in turn trained by bootstrapping the temporal difference error with itself. We present the modifications in algorithm 3. Similar modifications are applied to the standard PPO (cf. algorithm ??) that we present in the appendix (cf. algorithm ??). In the modified PPO, neural network policy $O \rightarrow A \cup A_{info}$ is trained using a neural network value function $S \times O \rightarrow A \cup A_{info}$ 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 (?)table 1]topin2021iterative in order to study direct deep reinforcement learning of decision tree policies for the CartPole MDP.

5. Experimental setup

5.1. (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 3.3) 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

Algorithm 2 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 α , exploration rate ϵ , partially observable Q-network parameters θ , Q-network parameters ϕ , replay buffer \mathcal{B} , update frequency C

Result: Partially observable deterministic policy π_{po}

Initialize partially observable Q-network parameters θ

Initialize Q-network parameters ϕ and target network parameters $\phi^- = \phi$

Initialize replay buffer $\mathcal{B} = \emptyset$

for each episode **do**

- Initialize base state features $s_0 \sim T_0$
- Initialize observation $o_0 = (L_1, U_1, \dots, L_p, U_p)$
- for** each step t **do**
- Choose action a_t using ϵ -greedy: $a_t = \text{argmax}_a Q_\theta(o_t, a)$ with prob. $1 - \epsilon$
- Take action a_t , observe r_t
- Store transition $(s_t, o_t, a_t, r_t, s_{t+1})$ in \mathcal{B}
- Sample random batch $(s_i, o_i, a_i, r_i, s_{i+1}) \sim \mathcal{B}$
- $a' = \text{argmax}_a Q_\theta(o_i, a)$
- $y_i = r_i + \gamma Q_\phi(s_{i+1}, a')$ // Compute target
- $\phi \leftarrow \phi - \alpha \nabla_\phi (Q_\phi(s_i, a_i) - y_i)^2$ // Update Q-network
- $\theta \leftarrow \theta - \alpha \nabla_\theta (Q_\theta(o_i, a_i) - y_i)^2$ // Update partially observable Q-network
- if** $t \bmod C = 0$ **then**
- $\theta^- \leftarrow \theta$ // Update target network
- end**
- $s_t \leftarrow s_{t+1}$
- $o_t \leftarrow o_{t+1}$

end

end

$\pi_{po}(o) = \text{argmax}_a Q_\theta(o, a)$ // Extract greedy policy

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 5.1) 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.

5.2. Baselines

Modified DQN and Modified PPO as mentioned above, the authors use the modified version of DQN from algorithm 3. We use the exact same hyperparameters for modified DQN as the authors when possible. We use the same layers width (128) and number of hidden layers (2), the same exploration strategy (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 $\pi : S \times O \rightarrow A \cup A_{info}$ and when learning standard $\pi : S \rightarrow A$ 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 4.3). 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).

5.3. 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 2 to extract a decision tree for the CartPole MDP. We then evaluate the tree on 100 independent trajectories in the MDP and report the mean undiscounted cumulative reward. For RL applied to IBMDPs, since we can’t deploy learned policies directly to the base MDP as the state dimensions mismatch—such policies are $S \times O \rightarrow A \cup A_{info}$ 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 5.1)). 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 fitted 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 (?)table 1]topin2021iterative.

6. Results

6.1. How well do modified deep RL baselines learn in IBMDPs?

On figure 6a, 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 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 6b, 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 6a and 6b, 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 5.2). On figure 6b, 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.

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

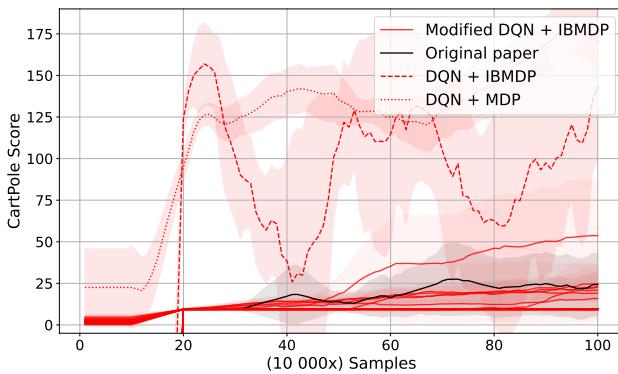
On figure 7, we isolate the best performing algorithms instantiations that learn decision tree policies for the CartPole MDP. We compare the best modified DQN and modified PPO to imitation learning baselines that use the surrogate imitation objective (cf. definition 4.4) to find CartPole decision tree 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 7 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.

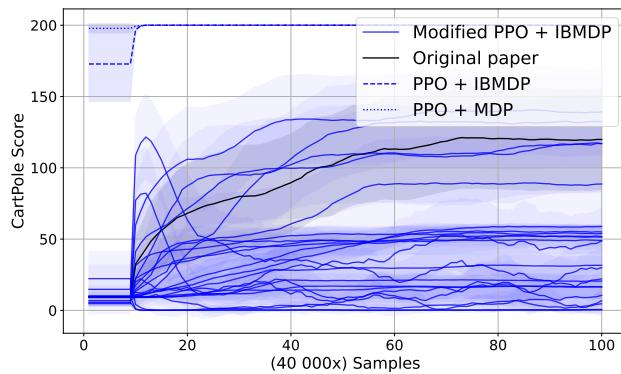
On figure 8, we present the best decision tree policies for CartPole returned by modified DQN and modified PPO. We used algorithm 2 to extract 20 trees from the 20 partially observable policies returned by the modified deep reinforcement learning algorithms over the 20 training seeds. We then plot the 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 5.1) 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. figures 6a and 6b). 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 7). 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 3.3) for an MDP (cf. definition 3.2), one can learn a deterministic partially observable policy that optimizes the interpretable RL objective (cf. definition ??) in an IBMDP (cf. definition 5.1 and proposition 5.2). 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.



(a) DQN variants



(b) PPO variants

Figure 1. 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.

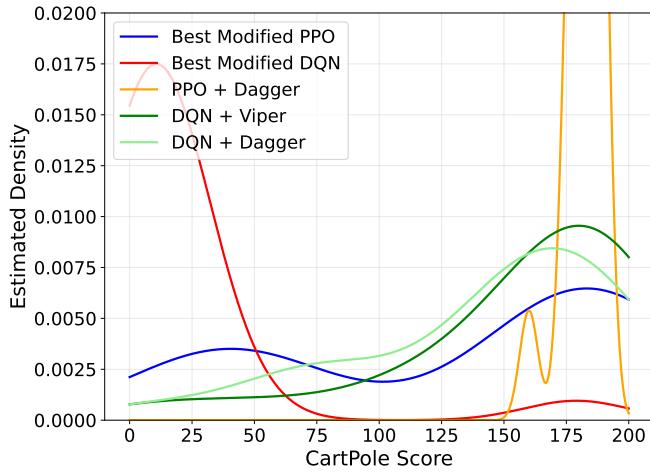
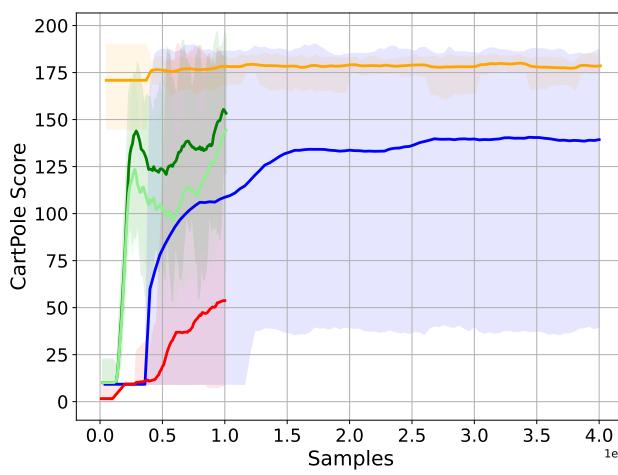


Figure 2. (left) Mean performance of the best-w.r.t. the RL objective for CartPole-modified RL + IBMDP combination. Shaded areas represent the min and max performance over the 20 seeds during training. (right) Corresponding score distribution of the final decision tree policies w.r.t. the RL objective for CartPole.



Figure 3. 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

7. Partially observable iterative Markov decision processes

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

Definition 7.1 (Partially observable Markov decision process). A partially observable Markov decision process is a tuple $\langle X, A, O, R, T, T_0, \Omega \rangle$ where:

- X is the hidden state space.
 - A is a finite set of actions.
 - O is a set of observations.
 - $T : X \times A \rightarrow \Delta(X)$ is the transition function, where $T(\mathbf{x}_t, a, \mathbf{x}_{t+1}) = P(\mathbf{x}_{t+1} | \mathbf{x}_t, a)$ is the probability of transitioning to state \mathbf{x}_t when taking action a in state \mathbf{x}
 - T_0 : is the initial distribution over states.
 - $\Omega : X \rightarrow \Delta(O)$ is the observation function, where $\Omega(o, a, x) = P(o | x, a)$ is the probability of observing o in state x
 - $R : X \times A \rightarrow \mathbb{R}$ is the reward function, where $R(x, a)$ is the immediate reward for taking action a in state x

Note that $\langle X, A, R, T, T_0 \rangle$ defines an MDP.

Let us define explicitly a partially observable iterative bounding Markov decision process (POIBMDP). It is essentially an IBMMDP for which we explicitly define an observation space and an observation function:

Definition 7.2 (Partially observable iterative bounding Markov decision process). a partially observable iterative bounding Markov decision process \mathcal{M}_{POIB} is a tuple:

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

, where $\langle S \times O, A \cup A_{info}, (R, \zeta), (T, T_0, T_{info}) \rangle$ is an IBMDP (cf. definition 5.1). The transition function Ω maps concatenation of state features and observations–IBMDP states–to observations, $\Omega : S \times O \rightarrow O$, with $P(o|(s, o)) = 1$

One can see POIBMDPs as 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 $\pi : O \rightarrow A \cup A_{info}$ in observation o .

Definition 7.3 (Partially observable value function). In a POIBMDP (cf. definition 9.2), the expected cumulative discounted reward of a deterministic partially observable policy $\pi : O \rightarrow A \cup A_{info}$ starting from observation o is $V^\pi(o)$:

$$V^\pi(o) = \sum_{(s,o') \in S \times O} P^\pi((s,o')|o) V^\pi((s,o'))$$

with $P^\pi((s, o')|o)$ the asymptotic occupancy distribution (see (?)section 4)learning-pomdp for the full definition) of the hidden POIBMDP state (s, o') given the partial observation o and $V^\pi((s, o'))$ the classical state-value function (cf. definition 4.1). 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, o') while observing o in some trajectory. In this chapter, we use reinforcement learning to train decision tree policies for MDPs by seeking deterministic partially observable policies that optimize the interpretable RL objective (cf. definition ??) in POIBMDPs (cf. definition 9.2).

The goal of the following sections is to see if, unlike indirect approaches tested in section 4.4 (cf. figures ?? and 4), direct approaches can consistently learn the depth-1 decision tree policies that have good interpretability-performance trade-offs for the 2×2 grid world from example 2 (cf. figures 3 and 9). The direct approach optimizes the interpretable RL objective ?? in POIBMDPs (cf. definition 9.2). We will use reinforcement learning to learn deterministic partially observable policies for the IBMMDP from example 5 re-written as a POIBMDP. Next we show how to choose γ and ζ in the POIBMDP such that the optimal deterministic partially observable policies w.r.t. the interpretable RL objective correspond exactly to the depth-1 decision tree policies with good interpretability-performance trade-offs. Hence, if we find that RL can consistently find optimal policies w.r.t. interpretable RL objective, it means that this direct approach can consistently find the depth-1 decision tree policies with good interpretability-performance trade-off that indirect approaches could not consistently find.

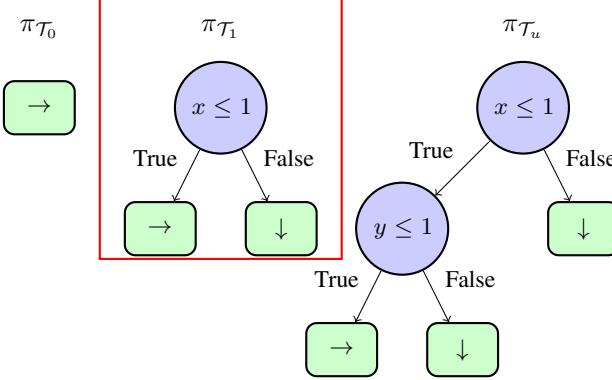


Figure 4. For each decision tree structure, e.g., depth-1 or unbalanced depth-2, we illustrate a decision tree which maximizes the RL objective (cf. definition 3.3) in the grid world MDP.

8. Constructing POIBMDPs whose optimal solutions are depth-1 decision tree policies

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 5), we can compute exactly the values of different deterministic partially observable policies given, ζ the reward for IGAs, and γ the discount factor. Each of those policies can be one of the trees illustrated in figure 9:

- π_{T_0} : a depth-0 tree equivalent to always taking the same base action
- π_{T_1} : a depth-1 tree equivalent alternating between an IGA and a base action
- π_{T_u} : an unbalanced depth-2 tree that sometimes takes two IGAs then a base action and sometimes a base action then an IGA
- π_{T_2} : a depth-2 tree that alternates between taking two IGAs and a base action
- 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 . Those two base actions always lead to the goal state (cf. figure 2).

Proposition 8.1 (Depth-1 decision tree objective value). *The interpretable RL objective value (cf. definition ??) of depth-1 decision tree policies with good interpretability–performance trade-offs in the grid world MDP (cf. figures 3 and 9) is $\frac{4\zeta + \gamma + 2\gamma^3 + \gamma^5}{4(1 - \gamma^2)}$.*

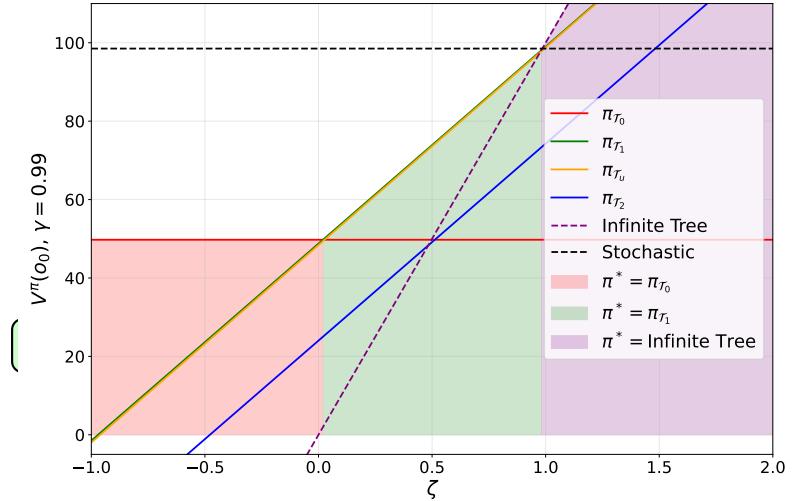


Figure 5. Interpretable RL objective values (cf. definition ??) of different partially observable policies as functions of ζ . Shaded areas show the optimal *deterministic* partially observable policies in different ranges of ζ values.

We defer the lengthy proofs of this proposition, as well as the interpretable RL objective values of other decision tree policies, to the appendix ??.

We can now plot, in figure 10, the interpretable RL objective values of the different partially observable policies as functions of ζ when we fix $\gamma = 0.99$. 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 interpretable RL objective, deterministic partially observable POIBMDP policy for $0 < \zeta < 1$.

Let us now define a POIBMDP with the grid world (cf. figure 2) as the base MDP, with IGAs as in the IBMDP from example 5, with $\gamma = 0.99$ and $0 < \zeta < 1$ and verify if RL can learn the optimal deterministic partially observable policies w.r.t. the interpretable RL objective, which are equivalent to depth-1 decision tree policies, in this very controlled experiment.

9. Reinforcement learning in PO(IB)MDPs

In general, the policy that maximizes the RL objective (cf. definition 3.3) in a POMDP (cf. definition 9.1) maps “belief states” or observation histories (Sigaud & Buffet, 2013) to actions. Hence, those policies are not solutions to our problem 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).

605
 606
 607
 608
 609
 610
 611
 612
 613
 614
 615
 616
 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.
 617
 618
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 In (?)Fact 2]learning-pomdp, 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 (?)Fact 5]learning-pomdp which makes it difficult to use TD-learning (cf. algorithms 1 and ??). Indeed, doing a TD-learning update of one partially observable value (cf. definition 9.3) with, e.g. Q-learning, can change the value of *all* other observations in an uncontrollable manner because of the dependence in $P^\pi((s, o')|o)$ (cf. definition 9.3). Interestingly, those two challenges of learning in POMDPs described in (Singh et al., 1994) are visible in figure 10. 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 $(s_2, o_0) = (1.5, 1.5, 0, 2, 0, 2)$ is not maximized by this partially observable policy but by the sub-optimal policy that always goes down.
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 Despite those hardness results, empirical results of applying RL to POMDPs by naively replacing x by 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 4 we present asymmetric Q-learning. It is a variant of Q-learning (cf. algorithm 1) that returns a deterministic partially observable policy like modified DQN 3. Given a POMDP, asymmetric Q-learning trains a partially observable Q-function $Q : O \times A \rightarrow \mathbb{R}$ and a Q-function $U : X \times A \rightarrow \mathbb{R}$.

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 4.1 that learns a (stochastic) policy $\pi : O \rightarrow \Delta(A)$ and a critic $V : X \rightarrow \mathbb{R}$ 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 $P^\pi((s, o')|o)$ (cf. definition 9.3) 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.

Note that, in the previous chapter and in the original work of Topin et. al. (Topin et al., 2021), modified DQN (cf. algorithm 3) and modified PPO (cf. algorithm ??) are respectively asymmetric DQN and asymmetric PPO from (Baisero et al., 2022; Baisero & Amato, 2022). In the next section, we use (asymmetric) RL to optimize the interpretable RL objective in POIBMDPs.

In the next section, we apply asymmetric and standard RL algorithms to the problem of learning the optimal depth-1 tree for the grid world MDP (cf. section 10) by optimizing the interpretable RL objective in POIBMDPs.

10. Results

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

10.1. 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 4.1, algorithms 1, ??, and ??). In theory the Policy Gradient

660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 710
 711
 712
 713
 714

Algorithm 3 Asymmetric Q-Learning. We highlight in green the differences with the standard Q-learning (cf. algorithm 1)

Data: POMDP $\mathcal{M}_{po} = \langle X, O, A, R, T, T_0, \Omega \rangle$, learning rates α_u , α_q , exploration rate ϵ

Result: $\pi : O \rightarrow A$

Initialize $U(\mathbf{x}, a) = 0$ for all $\mathbf{x} \in X, a \in A$
 Initialize $Q(\mathbf{o}, a) = 0$ for all $\mathbf{o} \in O, a \in A$

for each episode do
 Initialize state $x_0 \sim T_0$
 Initialize observation $\mathbf{o}_0 \sim \Omega(x_0)$

for each step t do
 Choose action a_t using ϵ -greedy: $a_t = \text{argmax}_a Q(\mathbf{o}_t, a)$ with prob. $1 - \epsilon$
 Take action a_t , observe $r_t = R(\mathbf{x}_t, a_t)$, $x_{t+1} \sim T(x_t, a_t)$, and $\mathbf{o}_{t+1} \sim \Omega(x_{t+1})$
 $y \leftarrow r + \gamma U(\mathbf{x}_{t+1}, \text{argmax}_{a'} Q(\mathbf{o}_{t+1}, a'))$ // TD target
 $U(\mathbf{x}_t, a_t) \leftarrow (1 - \alpha_u)U(\mathbf{x}_t, a_t) + \alpha_u y$
 $Q(\mathbf{o}_t, a_t) \leftarrow (1 - \alpha_q)Q(\mathbf{o}_t, a_t) + \alpha_q y$
 $x_t \leftarrow x_{t+1}$
 $\mathbf{o}_t \leftarrow \mathbf{o}_{t+1}$

end
end
 $\pi(o) = \text{argmax}_a Q(\mathbf{o}, a)$ // Extract greedy policy

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 10).

In addition to the traditional tabular RL algorithms above, we also apply asymmetric Q-learning, asymmetric Sarsa, and JSJ (algorithms 4, ?? 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: $|\mathbb{E}[V^{\pi^*}(s_0, o_0)|s_0 \sim T_0] - \mathbb{E}[V^\pi(s_0, o_0)|s_0 \sim T_0]|$. 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 10), 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 2 and we can count which one are of e.g. depth 1. This helps understand which trees RL algorithms tend to learn.

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

In figure 11, we plot the sub-optimality gaps—averaged over 100 seeds—of learned policies during training. We do so for 200 different POIBMDPs where we change the reward for information gathering actions: we sample 200 ζ values uniformly in $[-1, 2]$. In figure 11, a different color represents a different POIBMDP.

Recall from figure 10 that for:

- $\zeta \in [-1, 0]$, the optimal deterministic partially observable policy is a depth-0 tree
- $\zeta \in]0, 1[$, the optimal deterministic partially observable policy is a depth-1 tree
- $\zeta \in [1, 2]$, the optimal deterministic partially observable policy is a “infinite” tree that contains infinite number of internal nodes.

715
 716 We observe that, despite all sub-optimality gaps converging
 717 independently of the ζ values, not all algorithms in all POIB-
 718 MDPs fully minimize the sub-optimality gap. In particular,
 719 all algorithms seem to consistently minimize the gap, i.e.
 720 learn the optimal policy or Q-function, only for $\zeta \in [1, 2]$
 721 (all the yellow lines go to 0). However, we are interested in
 722 the range $\zeta \in]0, 1[$ where the optimal decision tree policy is
 723 non-trivial, i.e. not taking the same action forever. In that
 724 range, no baseline consistently minimizes the sub-optimality
 725 gap.

726 In figure 12, we plot the distributions of the final learned
 727 trees over the 100 random seeds in function of ζ from the
 728 above runs. For example, in figure 12, in the top left plot,
 729 when learning 100 times in a POIBMDP with $\zeta = 0.5$, Q-
 730 learning returned almost 100 times a depth-0 tree. Again, on
 731 none of those subplots do we see a high rate of learned depth-
 732 1 trees for $\zeta \in]0, 1[$. It is alerting that the most frequent
 733 learned trees are the depth-0 trees for $\zeta \in]0, 1[$ because such
 734 trees are way more sub-optimal w.r.t. the interpretable RL
 735 objective (cf. definition ??) than e.g. the depth-2 unbalanced
 736 trees (cf. figure 10). One interpretation of this phenomenon
 737 is that the learning in POIBMDPs is very difficult and so
 738 agents tend to converge to trivial policies, e.g., repeating the
 739 same base action.

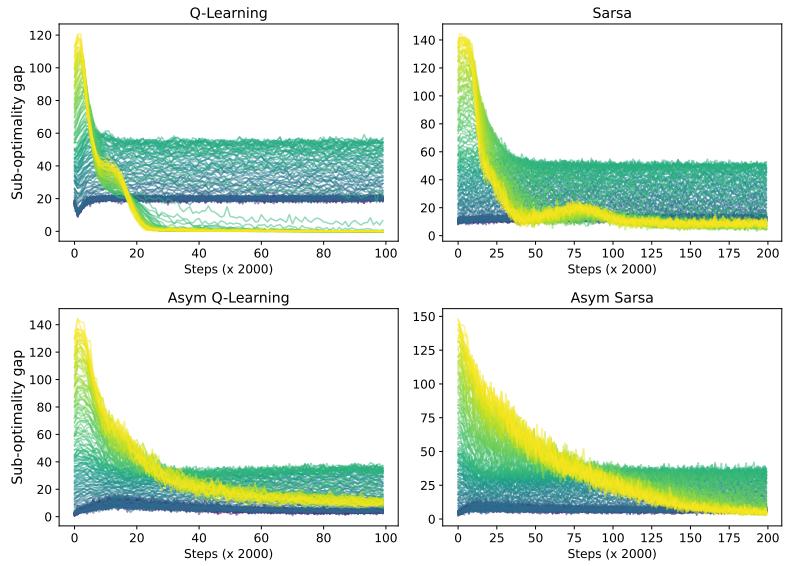
740 However, on the positive side, we observe that asymmetric
 741 versions of Q-learning and Sarsa have found the optimal
 742 deterministic partially observable policy—the depth-1 deci-
 743 sion tree—more frequently throughout the optimality range
 744 $]0, 1[$, than their symmetric counter-parts for $\zeta \in]0, 1[$. Next,
 745 we quantify how difficult it is to do RL to learn partially
 746 observable policies in POIBMDPs.

747 10.3. How difficult is it to learn in POIBMDPs?

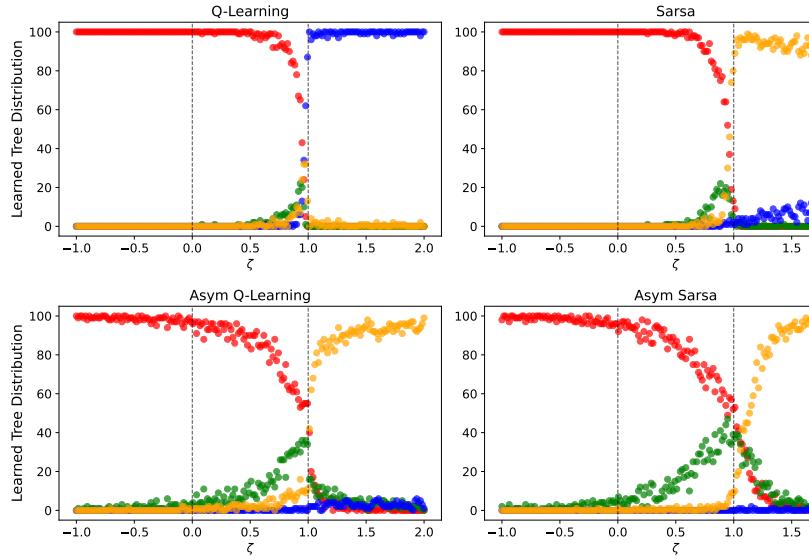
748 In this section we run the same (asymmetric) reinforcement
 749 learning algorithms to optimize either the RL objective (cf.
 750 definition 3.3) in MDPs (cf. definition 3.2) or IBMDPs
 751 (cf. definition 5.1), or the interpretable RL objective in
 752 POIBMDPs (cf. definition ??). This essentially results in
 753 three distinct problems:

- 754 1. Learning an optimal standard Markovian policy in an
 755 MDP, i.e. optimizing the RL objective in an MDP.
- 756 2. Learning an optimal standard Markovian policy in an
 757 IBMDP, i.e. optimizing the RL objective in an IBMDP.
- 758 3. Learning an optimal deterministic partially observable
 759 policy in a POIBMDP.

760 In order to see how difficult each of these three problems
 761 is, we can run a *great* number of experiments for each
 762 problem and compare solving rates. To make solving rates
 763 comparable we consider a unique instance for each of those



764 *Figure 6.* (Asymmetric) reinforcement learning in POIBMDPs. In
 765 each subplot, each single line is colored by the value of ζ in the
 766 corresponding POIBMDP in which learning occurs. Each single
 767 learning curve represent the sub-optimality gap averaged over 100
 768 seeds.



769 *Figure 7.* Distributions of final tree policies learned across the 100
 770 seeds. For each ζ value, there are four colored points. Each point
 771 represent the share of depth-0 trees (red), depth-1 trees (green),
 772 unbalanced depth-2 trees (orange) and depth-2 trees (blue).

Limits of RL for decision tree policies

770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
Table 1. 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 Asym Q-learning
Learning Rate Q	0.001, 0.005, 0.01, 0.1	the Q-function learning rate for the Asym Sarsa
Optimistic	True, False	partial observation dependent Q-function Optimistic initialization

problems. Problem 1 is learning one of the optimal standard Markovian deterministic policy from figure 2 for the grid world from example 2 with $\gamma = 0.99$. Problem 2 is learning one of the optimal standard Markovian deterministic for the IBMDP from figure 5 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. figure 6b). 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 1 and 2. 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 1). 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 9) as per figure 10).

The key observations from figure 13 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 ob-

Table 2. Summary of RL baselines Hyperparameters

algorithm	Problem	Total Hyperparameter Combinations
Policy Gradient	PO/IB/MDP	420
SJ	POIBMDP	15
Qlearning	PO/IB/MDP	192
Asym Q-learning	POIBMDP	768
Sarsa	PO/IB/MDP	192
Asym Sarsa	POIBMDP	768
partial observation dependent Q-function		
Optimistic initialization		

jective (cf. definition 3.3) 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 14 we re-run experiments from figure 11 and figure 12 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 14 despite higher success rates in the region $\zeta \in]0, 1[$ compared to figure 12.

11. 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 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 3.3). When compared to the results of VIPER from figure 4, 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.

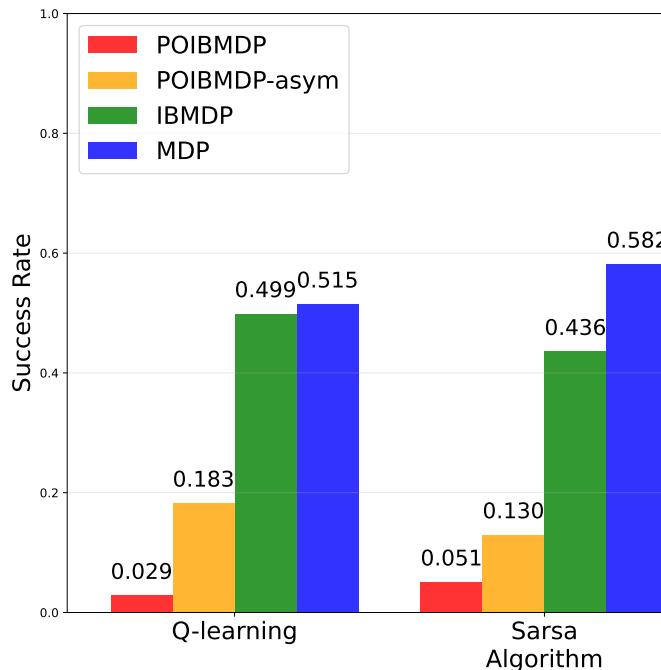
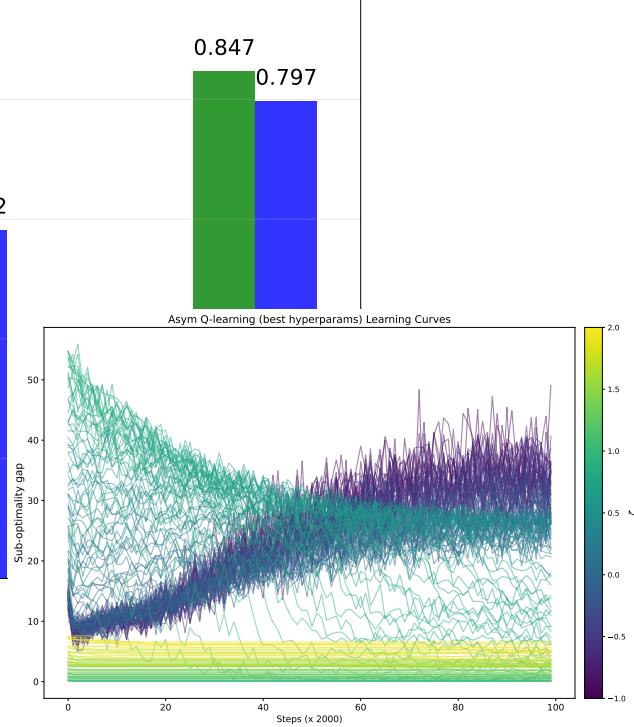
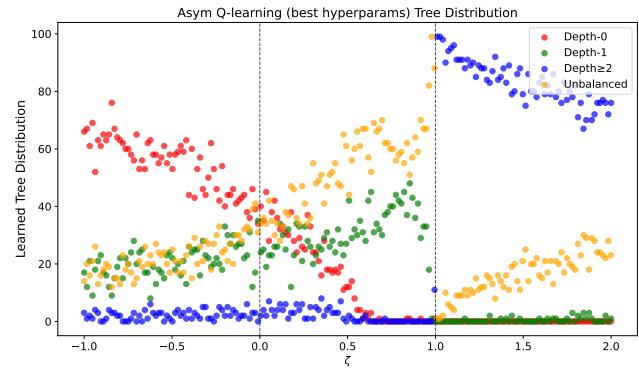


Figure 8. Success rates of different (asymmetric) RL algorithms over thousands of runs when applied to learning deterministic partially observable policies in a POIBMDP or learning deterministic policies in associated MDP and IBMDP.



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



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

Figure 9. Analysis of the top-performing asymmetric Q-learning instantiation. (left) Learning curves, and (right) tree distributions across different POIBMDP configurations.

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 3.3) is so difficult.

12. Classification tasks

In this section, we show that for a special class of POIBMDPs (cf. definition 9.2), reinforcement learning (cf. section 4.1) 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. $T(s, a, s') = \frac{1}{|S|}$ (cf. definitions 3.2 and 5.1). The supervised learning objective (cf. definition ??) can be re-formulated in terms of the RL objective (cf. definition 3.3) 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 12.1 (Classification Markov decision process). Given a set of N examples denoted $\mathcal{E} = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^N$ where each datum $\mathbf{x}_i \in \mathcal{X}$ 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 \mathbf{x}_i , a classification Markov decision Process is an MDP $\langle S, A, R, T, T_0 \rangle$ (cf. definition 3.2) where:

- the state space is $S = \{\mathbf{x}_i\}_{i=1}^N$, the set of training data features
- the action space is $A = \mathbb{Z}^m$, the set of unique labels
- the reward function is $R : S \times A \rightarrow \{0, 1\}$ with $R(\mathbf{s} = \mathbf{x}_i, a) = 1_{\{a=y_i\}}$
- the transition function is $T : S \times A \rightarrow \Delta(S)$ with $T(\mathbf{s}, a, \mathbf{s}') = \frac{1}{N} \quad \forall \mathbf{s}, a, \mathbf{s}'$
- the initial distribution is $T_0(\mathbf{s}_0 = \mathbf{s}) = \frac{1}{N}$

One can be convinced that policies that maximize the RL objective (cf. definition 3.3) in classification MDPs are classifiers that maximize the prediction accuracy because $\sum_{i=1}^N 1_{\pi(\mathbf{x}_i)=y_i} = \sum_{i=1}^N R(\mathbf{x}_i, \pi(\mathbf{x}_i))$. We defer the formal proof in the next part of the manuscript in which we extensively study supervised learning problems.

In figure 15 we give an example of such classification MDP with 4 data in the training set and 2 classes:

$$\begin{aligned} \mathcal{X} &= \{(0.5, 0.5), (0.5, 1.5), (1.5, 1.5), (1.5, 0.5)\} \\ y &= \{0, 0, 1, 1\} \end{aligned}$$

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

Definition 12.2 (Classification POIBMDP). Given a classification MDP $\langle \{\mathbf{x}_i\}_{i=1}^N, \mathbb{Z}^m, R, T, T_0 \rangle$ (cf. definition 14.1), and an associated POIBMDP $\langle S, O, A, A_{info}, R, \zeta, T_{info}, T, T_0 \rangle$ (cf. definition 9.2), a classification POIBMDP is an MDP (cf. definition 3.2):

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

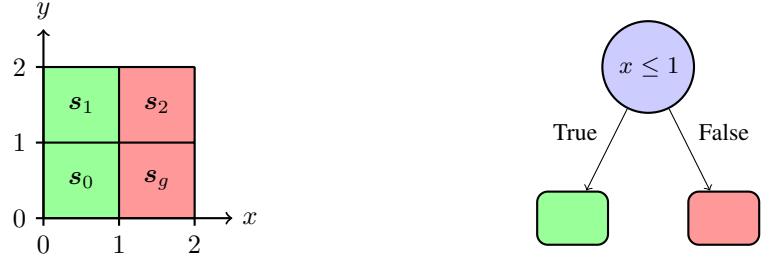


Figure 10. Classification MDP optimal actions. In this classification MDP, there are four data to which to assign either a green or red label. On the right, there is the unique optimal depth-1 tree for this particular classification MDP. This depth-1 tree also maximizes the accuracy on the corresponding classification task.

- 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 \mathbf{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 $\mathbf{o} = (L'_1, U'_1, \dots, L'_p, U'_p)$ is the proportion of training data satisfying the bounds and having label a : $R(o, a) = \frac{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall i, j\} \cap \{\mathbf{x}_i : y_i = a \forall i\}|}{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall i, j\}|}$. 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 $a \in \mathbb{Z}^m$: $\mathcal{P}(\mathbf{o}, a, (L_1, U_1, \dots, L_p, U_p)) = 1$ (reset to full bounds)
 - For $a = (k, v) \in A_{info}$: from $\mathbf{o} = (L'_1, U'_1, \dots, L'_p, U'_p)$, the MDP will transit to $\mathbf{o}_{left} = (L'_1, U'_1, \dots, L_k, v, \dots, L'_p, U'_p)$ (resp. $\mathbf{o}_{right} = (L'_1, U'_1, \dots, U'_k, v, \dots, L'_p, U'_p)$) with probability $\frac{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j \wedge x_{ik} \leq v\}|}{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j\}|}$ (resp. $\frac{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j \wedge x_{ik} > v\}|}{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall j\}|}$)

Those classification POIBMDPs are essentially MDPs with stochastic transitions. It means that deterministic partially observable policies (cf. definition 5.2) $O : \rightarrow A \cup A_{info}$ 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 4.1) should work as well as for any MDP to learn optimal decision tree policies.

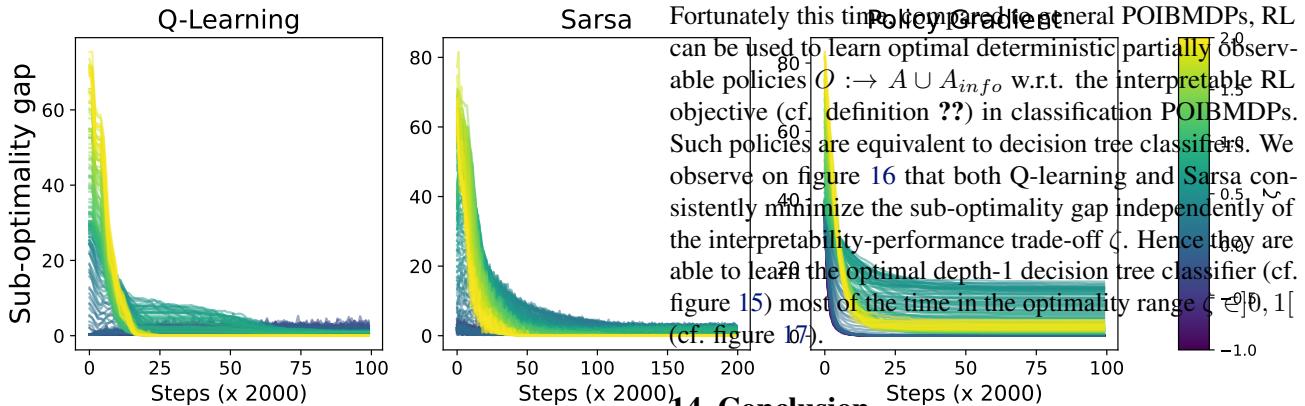


Figure 11. We reproduce the same plot as in figure 11 for classification POIBMDPs. Each individual curve is the sub-optimality gap of the learned policy during training averaged over 100 runs for a single ζ value.

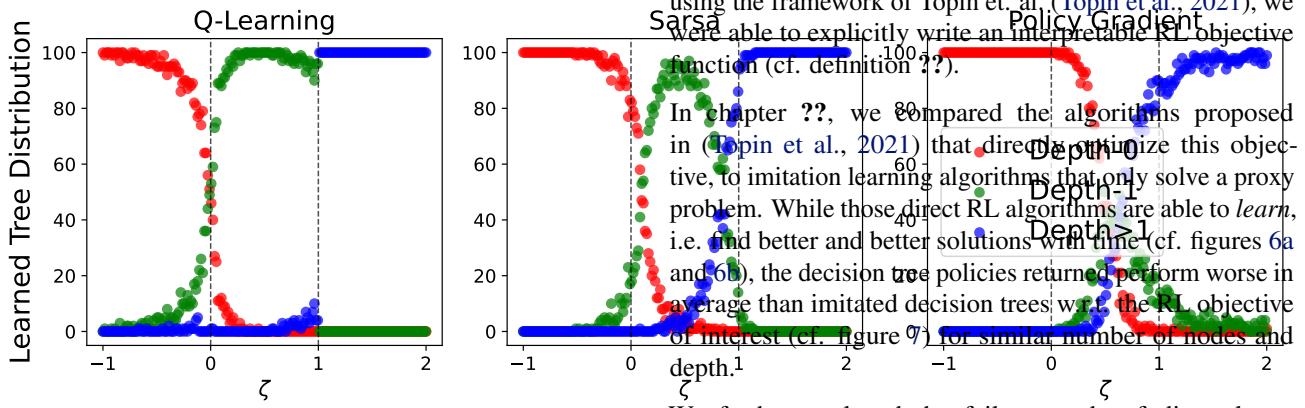


Figure 12. We reproduce the same plot as in figure 12 for classification POIBMDPs. Each colored dot is the number of final learned trees with a specific structure for a given ζ .

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.

13. 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 15.

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 4, ??, and ??).

Fortunately this time Policy Gradient 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 16 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 15) most of the time in the optimality range $\zeta \in [0, 1[$ (cf. figure 10).

14. 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 3.3) 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 6a and 6b), the decision tree policies returned perform worse in average than imitated decision trees w.r.t. the RL objective of interest (cf. figure 7) for similar number of nodes and depth.¹

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 9.2). 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 12.3). We demonstrated on figure 13 that introducing partial observability greatly reduced the success rates (we also observed this implicitly on figures 6a and 6b).

Finally, in this chapter we showed that using RL to optimize the interpretable RL objective in fully observable POIB-

990 MDPs, i.e. POIBMDPs that are just MDPs, could learn
 991 optimal decision tree policies (cf. figures 16 and 17) adding
 992 new evidence that direct interpretable RL is difficult because
 993 it involves POMDPs.

994 This class of fully observable POIBMDPs (cf. definition 14.2) contains the decision tree induction problem for
 995 supervised learning tasks (cf. definition ??). This sparks
 996 the question: what kind of decision tree induction algorithm
 997 can we get using the MDP formalism? This is exactly what
 998 we study in the next part of this manuscript.
 999

1000 Those few chapters raise other interesting questions. We
 1001 focused on non-parametric tree learning because RL al-
 1002 gorithms can learn decision tree policies with potentially
 1003 optimal interpretability-performance trade-offs through the
 1004 reward of information gathering actions in (PO)IBMDPs
 1005 (cf. definitions 5.1 and 9.2). However this comes at a cost
 1006 of partial observability which makes learning difficult. Para-
 1007 metric tree policies on the other hand, can be computed with
 1008 reinforcement learning directly in the base MDP. However
 1009 existing RL algorithms for parametric decision tree poli-
 1010 cies (Silva et al., 2020; Vos & Verwer, 2024; Marton et al.,
 1011 2025) require to re-train a policy entirely for each desired
 1012 level of interpretability, i.e. each unique tree structure, fu-
 1013 ture research in this direction should focus on algorithms
 1014 for parametric tree policies that can re-use samples from
 1015 one tree learning to train a different tree structure more effi-
 1016 ciently. This would reduce the required quantity of a priori
 1017 knowledge on the decision tree policy structure mentioned
 1018 in section 2.
 1019

1020 Attempting to overcome the partial observability challenges
 1021 highlighted so far seems like a bad research direction. In-
 1022 deed, while algorithms tailored specifically for the problem
 1023 of learning deterministic partially observable policies for
 1024 POIBMDPs might exist, we clearly saw that imitation learn-
 1025 ing was in practice a good alternative to direct interpretable
 1026 reinforcement learning. Some limitations that we did not
 1027 cover still exist such as how to choose good candidates in-
 1028 formation gathering actions or simply how to choose ζ for a
 1029 target interpretability-performance trade-off.
 1030

1031 References

1032 Araya-López, M., Thomas, V., Buffet, O., and Charpillet,
 1033 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, 1034 Arras, France, October 2010. IEEE. URL <https://inria.hal.science/inria-00535559>.

1041 Atrey, A., Clary, K., and Jensen, D. Exploratory not ex-
 1042 planatory: Counterfactual analysis of saliency maps for
 1043 deep reinforcement learning. In *International Confer-
 1044*

ence on Learning Representations

2020. URL <https://openreview.net/forum?id=rkl3mlBFDB>.

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.

- 1045 Delfosse, Q., Blüml, J., Gregori, B., Sztwiertnia, S., and Ker-
 1046 sting, K. OCAtari: Object-centric Atari 2600 reinforce-
 1047 ment learning environments. *Reinforcement Learning*
 1048 *Journal*, 1:400–449, 2024a.
- 1049
- 1050 Delfosse, Q., Sztwiertnia, S., Rothermel, M., Stam-
 1051 mer, W., and Kersting, K. Interpretable concept
 1052 bottlenecks to align reinforcement learning agents.
 1053 2024b. URL <https://openreview.net/forum?id=ZC0PSk6Mc6>.
- 1054
- 1055 Demirovic, E., Lukina, A., Hebrard, E., Chan, J., Bai-
 1056 ley, J., Leckie, C., Ramamohanarao, K., and Stuckey,
 1057 P. J. Murtree: Optimal decision trees via dynamic pro-
 1058 gramming and search. *Journal of Machine Learning Re-*
 1059 *search*, 23(26):1–47, 2022. URL <http://jmlr.org/papers/v23/20-520.html>.
- 1060
- 1061 Demirović, E., Hebrard, E., and Jean, L. Blossom: an
 1062 anytime algorithm for computing optimal decision
 1063 trees. *Proceedings of the 40th International Confer-
 1064 ence on Machine Learning*, 202:7533–7562, 23–29 Jul
 1065 2023. URL <https://proceedings.mlr.press/v202/demirovic23a.html>.
- 1066
- 1067 Glanois, C., Weng, P., Zimmer, M., Li, D., Yang, T., Hao,
 1068 J., and Liu, W. A survey on interpretable reinforcement
 1069 learning. *Machine Learning*, pp. 1–44, 2024.
- 1070
- 1071 Jaakkola, T., Singh, S. P., and Jordan, M. I. Reinforcement
 1072 learning algorithm for partially observable markov deci-
 1073 sion problems. In *Proceedings of the 8th International
 1074 Conference on Neural Information Processing Systems*,
 1075 NIPS’94, pp. 345–352, Cambridge, MA, USA, 1994.
 1076 MIT Press.
- 1077
- 1078 Lambrechts, G., Bolland, A., and Ernst, D. Informed
 1079 POMDP: Leveraging additional information in model-
 1080 based RL. *Reinforcement Learning Journal*, 2:763–784,
 1081 2025a.
- 1082
- 1083 Lambrechts, G., Ernst, D., and Mahajan, A. A the-
 1084 oretical justification for asymmetric actor-critic algo-
 1085 rithms. In *Forty-second International Conference on Ma-*
 1086 *chine Learning*, 2025b. URL <https://openreview.net/forum?id=F1yANMCnAn>.
- 1087
- 1088 Lipton, Z. C. The mythos of model interpretability: In
 1089 machine learning, the concept of interpretability is both
 1090 important and slippery. *Queue*, 16(3):31–57, 2018.
- 1091
- 1092 Littman, M. L. Memoryless policies: theoretical limitations
 1093 and practical results. In *Proceedings of the Third Inter-
 1094 national Conference on Simulation of Adaptive Behavior:
 1095 From Animals to Animats 3: From Animals to Animats 3*,
 1096 SAB94, pp. 238–245, Cambridge, MA, USA, 1994. MIT
 1097 Press. ISBN 0262531224.
- 1098
- 1099 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 Inter-
 national 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 inter-
 preting model predictions. In *Proceedings of the 31st Inter-
 national 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 learn-
 ing 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., Fidje-
 land, 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., Cour-
 napeau, D., Brucher, M., Perrot, M., and Duchesnay, E.
 Scikit-learn: Machine learning in Python. *Journal of
 Machine Learning Research*, 12:2825–2830, 2011.

- 1100 Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W.,
 1101 and Abbeel, P. Asymmetric actor critic for image-based
 1102 robot learning, 2017. URL <https://arxiv.org/abs/1710.06542>.
- 1103
- 1104 Puri, N., Verma, S., Gupta, P., Kayastha, D., Deshmukh,
 1105 S., Krishnamurthy, B., and Singh, S. Explain your
 1106 move: Understanding agent actions using specific and
 1107 relevant feature attribution. In *International Conference
 1108 on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SJgzLkBKPB>.
- 1109
- 1110 Puterman, M. L. *Markov Decision Processes: Discrete
 1111 Stochastic Dynamic Programming*. John Wiley & Sons,
 1112 1994.
- 1113
- 1114 Quinlan, J. R. Induction of decision trees. *Mach. Learn.*, 1
 1115 (1):81–106, 1986.
- 1116
- 1117 Quinlan, J. R. C4. 5: Programs for machine learning. *Mor-
 1118 gan Kaufmann google schola*, 2:203–228, 1993.
- 1119
- 1120 Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus,
 1121 M., and Dormann, N. Stable-baselines3: Reliable rein-
 1122 forcement learning implementations. *Journal of Machine
 1123 Learning Research*, 22(268):1–8, 2021.
- 1124
- 1125 Ribeiro, M. T., Singh, S., and Guestrin, C. “why
 1126 should i trust you?”: Explaining the predictions of
 1127 any classifier. pp. 1135–1144, 2016. doi: 10.1145/
 1128 2939672.2939778. URL <https://doi.org/10.1145/2939672.2939778>.
- 1129
- 1130
- 1131 Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of
 1132 imitation learning and structured prediction to no-regret
 1133 online learning. 2010.
- 1134
- 1135 Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and
 1136 Klimov, O. Proximal policy optimization algorithms.
 1137 *arXiv preprint arXiv:1707.06347*, 2017.
- 1138
- 1139 Shi, W., Huang, G., Song, S., Wang, Z., Lin, T., and Wu, C.
 1140 Self-supervised discovering of interpretable features for
 1141 reinforcement learning. *IEEE Transactions on Pattern
 1142 Analysis and Machine Intelligence*, 44(5):2712–2724,
 1143 2022. doi: 10.1109/TPAMI.2020.3037898.
- 1144
- 1145 Shindo, H., Delfosse, Q., Dhami, D. S., and Kersting, K.
 1146 Blendrl: A framework for merging symbolic and neural
 1147 policy learning. *arXiv*, 2025.
- 1148
- 1149 Sigaud, O. and Buffet, O. *Partially Observable Markov
 1150 Decision Processes*, chapter 7, pp. 185–228. John
 1151 Wiley Sons, Ltd, 2013. ISBN 9781118557426. doi:
 1152 <https://doi.org/10.1002/9781118557426.ch7>. URL
 1153 <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118557426.ch7>.
- 1154
- Silva, A., Gombolay, M., Killian, T., Jimenez, I., and Son,
 1155 S.-H. Optimization methods for interpretable differentiable
 1156 decision trees applied to reinforcement learning.
 1157 In Chiappa, S. and Calandra, R. (eds.), *Proceedings of
 1158 the Twenty Third International Conference on Artificial
 1159 Intelligence and Statistics*, volume 108 of *Proceedings of
 1160 Machine Learning Research*, pp. 1855–1865. PMLR, 26–
 1161 28 Aug 2020. URL <https://proceedings.mlr.press/v108/silva20a.html>.
- Singh, S. P., Jaakkola, T. S., and Jordan, M. I. Learning
 1162 without state-estimation in partially observable marko-
 1163 vian decision processes. In *Proceedings of the Eleventh
 1164 International Conference on International Conference on
 1165 Machine Learning*, ICML’94, pp. 284–292, San Fran-
 1166 cisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
 1167 ISBN 1558603352.
- Sondik, E. J. The optimal control of partially observable
 1168 markov processes over the infinite horizon: Discounted
 1169 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
 1170 Introduction*. The MIT Press, Cambridge, MA, 1998.
- Sutton, R. S., McAllester, D., Singh, S., and Mansour,
 1171 Y. Policy gradient methods for reinforcement learning
 1172 with function approximation. In Solla, S., Leen,
 1173 T., and Müller, K. (eds.), *Advances in Neural Infor-
 1174 mation Processing Systems*, volume 12. MIT Press,
 1175 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-
 1176 0782. doi: 10.1145/203330.203343. URL <https://doi.org/10.1145/203330.203343>.
- Topin, N., Milani, S., Fang, F., and Veloso, M. Iterative
 1177 bounding mdps: Learning interpretable policies via non-
 1178 interpretable methods. *Proceedings of the AAAI Confer-
 1179 ence on Artificial Intelligence*, 35:9923–9931, 2021.
- Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola,
 1180 G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M.,
 1181 KG, A., et al. Gymnasium: A standard interface for
 1182 reinforcement learning environments. *arXiv preprint
 1183 arXiv:2407.17032*, 2024.
- van der Linden, J., de Weerdt, M., and Demirović, E. Nec-
 1184 essary and sufficient conditions for optimal decision trees
 1185 using dynamic programming. *Advances in Neural Infor-
 1186 mation Processing Systems*, 36:9173–9212, 2023.

1155 Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri,
1156 S. Programmatically interpretable reinforcement learning.
1157 pp. 5045–5054, 2018.

1158
1159 Verwer, S. and Zhang, Y. Learning optimal classification
1160 trees using a binary linear program formulation. *Proceed-
1161 ings of the AAAI conference on artificial intelligence*, 33:
1162 1625–1632, 2019.

1163 Vos, D. and Verwer, S. Optimal decision tree policies
1164 for markov decision processes. In *Proceedings of the
1165 Thirty-Second International Joint Conference on Artifi-
1166 cial Intelligence*, IJCAI ’23, 2023. ISBN 978-1-956792-
1167 03-4. doi: 10.24963/ijcai.2023/606. URL <https://doi.org/10.24963/ijcai.2023/606>.

1168
1169
1170 Vos, D. and Verwer, S. Optimizing interpretable decision
1171 tree policies for reinforcement learning. 2024. URL
1172 <https://arxiv.org/abs/2408.11632>.

1173
1174 Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*,
1175 8(3):279–292, 1992.

1176 Wu, H., Isac, O., Zeljić, A., Tagomori, T., Daggitt, M.,
1177 Kokke, W., Refaeli, I., Amir, G., Julian, K., Bassan,
1178 S., Huang, P., Lahav, O., Wu, M., Zhang, M., Komen-
1179 dantskaya, E., Katz, G., and Barrett, C. Marabou 2.0: A
1180 versatile formal analyzer of neural networks, 2024. URL
1181 <https://arxiv.org/abs/2401.14461>.

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

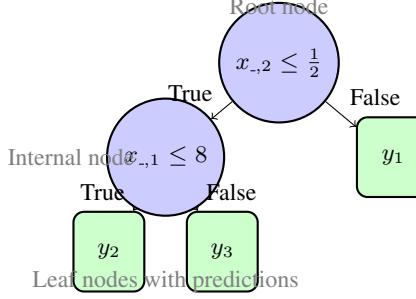


Figure 13. A generic depth 2 decision tree with 2 nodes and 3 leaves. The root node applies the test $1_{\{x_{\cdot,1} \leq \frac{1}{2}\}}$ to check if the first features of data is below $\frac{1}{2}$. Edges represent the outcomes of the tests in each internal nodes(True/False), and leaf nodes contain predictions $y_l \in \mathcal{Y}$. For any input x_i , the tree defines a unique path from root to leaf.

A. Appendix for technical preliminaries

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 1.

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 1, ?? and ???. Q-learning and Sarsa compute an approximation of Q^* (cf. definition 4.3) 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 π^* .

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.

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 3.2). In the rest of this manuscript, unless stated

1265 **Algorithm 4** Q-Learning (Watkins & Dayan, 1992)

1266 **Data:** MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, learning rate α , exploration rate ϵ

1267 **Result:** Policy π

1268 Initialize $Q(s, a) = 0$ for all $s \in S, a \in A$

1269 Initialize state $s_0 \sim T_0$

1270 **for** each step t **do**

1271 Choose action a_t using e.g. ϵ -greedy policy: $a_t = \text{argmax}_a Q(s_t, a)$ with prob. $1 - \epsilon$

1272 Take action a_t , observe $r_t = R(s_t, a_t)$ and $s_{t+1} \sim T(s_t, a_t)$

1273 $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$

1274 $s_t \leftarrow s_{t+1}$

1275 **end**

1276 $\pi(s) = \text{argmax}_a Q(s, a)$ // Extract greedy policy

1279 otherwise, we write s a state vector in a continuous state space¹.

1280 Deep Q-Networks (DQN) (Mnih et al., 2015), described in algorithm ?? achieved super-human performance on a set of
 1281 Atari games. Authors successfully extended the Q-learning (cf. algorithm 1) to the function approximation setting by
 1282 introducing target networks to mitigate distributional shift in the temporal difference error and replay buffer to increase
 1283 sample efficiency.

1284 Proximal Policy Optimization (PPO) (Schulman et al., 2017), described in algorithm ??, is an actor-critic algorithm (Sutton
 1285 & Barto, 1998) optimizing a neural network policy. In actor-critic algorithms, cumulative discounted rewards starting from a
 1286 particular state, also known as *the returns*, are also estimated with a neural network. PPO is known to work well in a variety
 1287 of domains including robot control in simulation among others.

1288 **A.4. Imitation learning: a baseline (indirect) interpretable reinforcement learning method**

1289 Unlike PPO or DQN for neural networks, there exists no algorithm that trains decision tree policies to optimize the RL
 1290 objective (cf. definition 3.3). In fact, we will show in the first part of the manuscript that training decision trees that optimize
 1291 the RL objective is very difficult.

1292 Hence, many interpretable reinforcement learning approaches first train a neural network policy—also called an expert
 1293 policy—to optimize the RL objective (cf. definition 3.3) using e.g. PPO, and then fit a student policy such as a decision tree
 1294 using CART (cf. algorithm ??) to optimize the supervised learning objective (cf. definition ??) with the neural policy actions
 1295 as targets. This approach is known as imitation learning and is essentially training a student policy to optimize the objective:

1296 **Definition A.2** (Imitation learning objective). Given an MDP \mathcal{M} (cf. definition 3.2), an expert policy π^* and a policy class
 1297 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
 1298 expected action disagreement with the expert:

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

1300 where $\rho(s)$ is the state distribution in \mathcal{M} induced by the student policy π and \mathcal{L} is a loss function measuring the disagreement
 1301 between the student policy's action $\pi(s)$ and the expert's action $\pi^*(s)$.

1302 There are two main imitation learning methods used for interpretable reinforcement learning. Dagger (cf. algorithm ??) is a
 1303 straightforward way to fit a decision tree policy to optimize the imitation learning objective (cf. definition 4.4). VIPER
 1304 (cf. algorithm ??) was designed specifically for interpretable reinforcement learning. VIPER re-weights the transitions
 1305 collected by the neural network expert by a function of the state-action value (cf. definition 4). The authors of VIPER
 1306 showed that decision tree policies fitted with VIPER tend to have the same RL objective value as Dagger trees while being
 1307 more interpretable (shallower or with fewer nodes) and sometimes outperform Dagger trees. Dagger and VIPER are two
 1308 strong baselines for decision tree learning in MDPs, but they optimize a surrogate objective only, even though in practice the
 1309 resulting decision tree policies often achieve high RL objective value. We use these two algorithms extensively throughout
 1310 the manuscript. Next we show how to learn a decision tree policy for the example MDP (cf. figure 2).

1311 ¹Note that discrete states can be one-hot encoded as state vectors in $\{0, 1\}^{|S|}$.

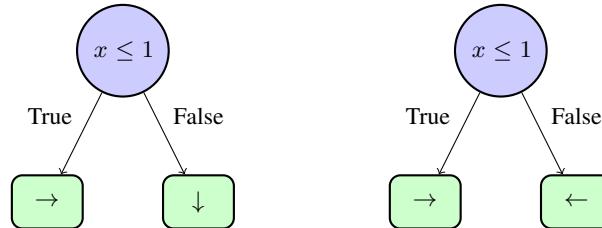


Figure 14. Left, an optimal depth-1 decision tree policy. On the right, a sub-optimal depth-1 decision tree policy.

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 3.3) 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 4.4) 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 2 using Q-learning (cf. algorithm 1). Recall the optimal policies for the grid world, taking the green actions in each state in figure 2. 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 3, 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 2, 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 2?

We start by running the standard Q-learning algorithm as presented in algorithm 1 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 1. 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 4, 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 4, 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.

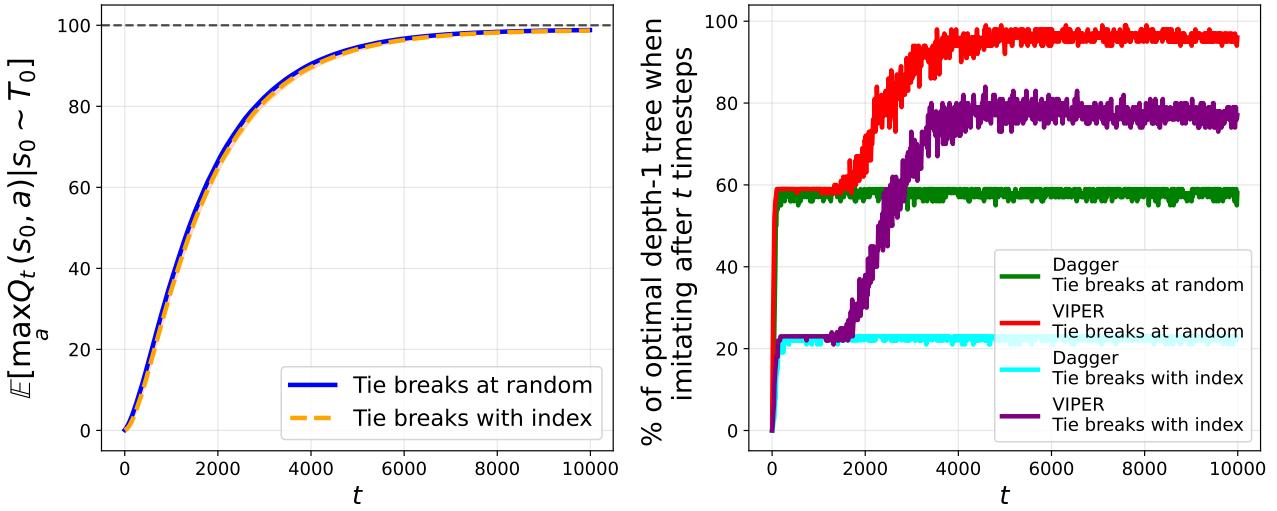


Figure 15. Left, sample complexity curve of Q-learning with default hyperparameters on the 2×2 grid world MDP over 100 random seeds. Right, performance of indirect interpretable methods when imitating the greedy policy with a tree at different Q-learning stages.

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

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

Suppose an associated IBMDP (definition 5.1) 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 $o_0 = (0, 2, 0, 2)$ 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 o_0 following the IBMDP transitions (cf. definition 5.1). For example when the IBMDP initial base state features are $s_0 = (0.5, 1.5)$, and taking first $\langle x, 1 \rangle$ then $\langle y, 1 \rangle$ the corresponding observations are first $o_{t+1} = (0, 1, 0, 2)$ and then $o_{t+2} = (0, 1, 1, 2)$. 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 5.1).

In figure 5 we illustrate a trajectory in this IBMDP.

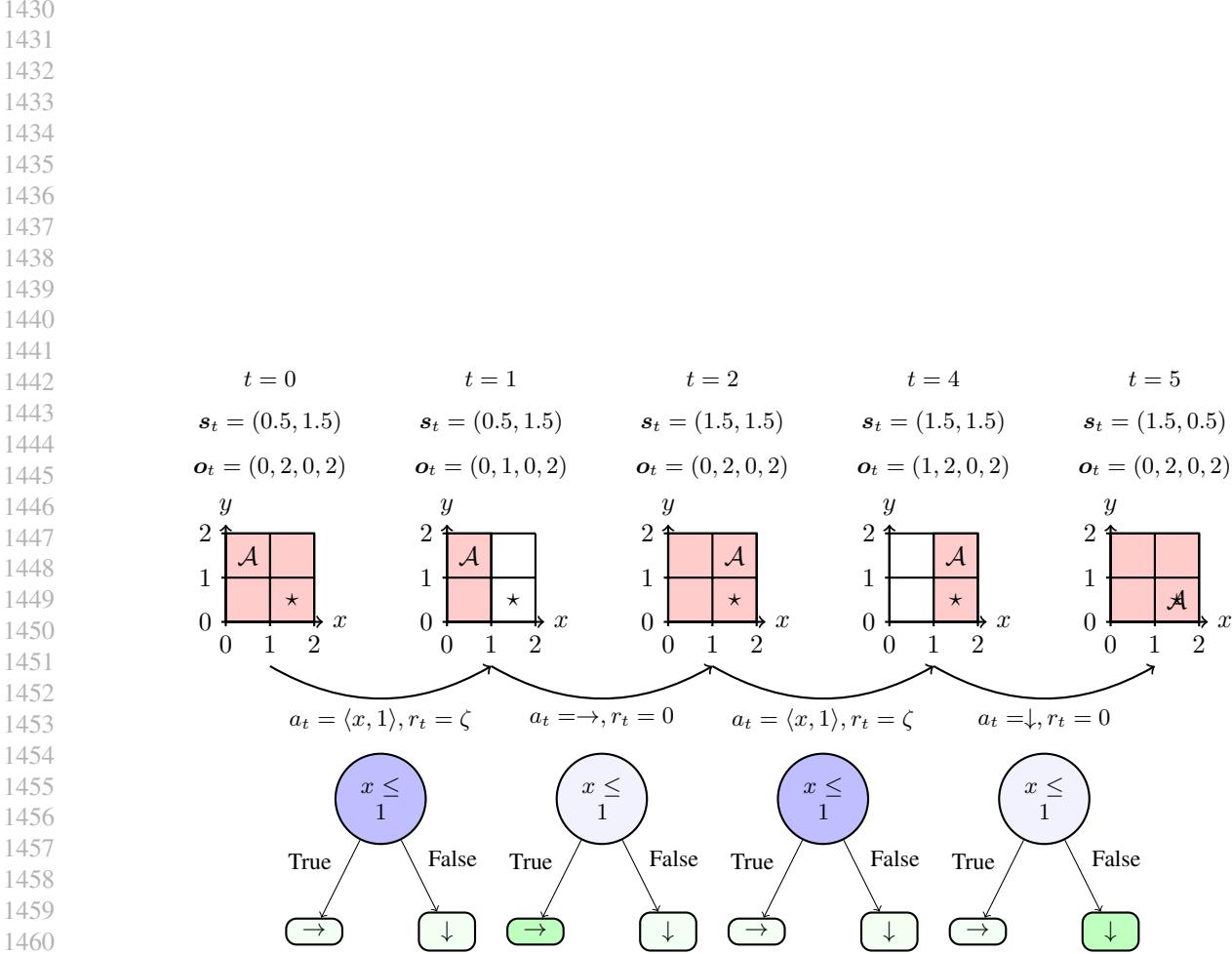


Figure 16. An IBMDP trajectory when the base MDP is 2×2 grid world. In the top row, we write the visited base state features and observations, in the middle row, we graphically represent those, and in the bottom row, we present the corresponding decision tree policy traversal. \mathcal{A} tracks the current state features s_t in the grid. The pink obstructions of the grid represent the current observations 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 $s_0 = (0.5, 1.5)$. The initial observation is always the base MDP default state feature bounds, here $o_0 = (0, 2, 0, 2)$ because the base state features are in $[0, 2] \times [0, 2]$. This means that the IBMDP state is $s_{IB} = (0.5, 1.5, 0, 2, 0, 2)$. 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 $o_1 = (0, 1, 0, 2)$, 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 $s_2 = (1.5, 1.5)$. And the trajectory continues like this until the absorbing base state $s_5 = (1.5, 0.5)$ is reached.