
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.

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

2. Related work

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

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

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

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

provides a constructive algorithm. In settings where the dynamics and rewards are known, ([Vos & Verwer, 2023](#)) use planning to compute shallow parametric decision tree policies (up to depth 3). Next, we recall useful technical material.

3. Technical preliminaries

3.1. Markov decision processes

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

Definition 3.1 (Markov decision process). An MDP is a tuple $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$. S is a finite set of states representing all possible configurations of the environment. A is a finite set of actions available to the agent. $R : S \times A \rightarrow \mathbb{R}$ 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. We can formally define this objective, that we call the reinforcement learning objective, as follows:

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

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

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

Algorithms presented in this manuscript aim to find an optimal policy $\pi^* \in \arg\max_{\pi} J(\pi)$ that maximizes the above reinforcement learning (RL) objective. In particular, RL algorithms ([Sutton & Barto, 1998](#); [?](#); [Watkins & Dayan,](#)

110 1992; Mnih et al., 2015; Schulman et al., 2017) learn such
 111 optimal policies using data of MDP interactions without
 112 prior knowledge of the reward and transition models. Useful
 113 quantities for such algorithms include *value* of states and
 114 actions.

115 **Definition 3.3** (Value of a state). In an MDP \mathcal{M} (cf. definition 3.1), the value of a state $s \in S$ under policy π is the
 116 expected discounted sum of rewards starting from state s
 117 and following policy π :

$$119 V^\pi(s) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s, s_{t+1} \sim T(s_t, \pi(s_t)) \right]$$

122 Applying the Markov property gives a recursive definition
 123 of the value of s under policy π : $V^\pi(s) = R(s, \pi(s)) +$
 124 $\gamma \mathbb{E}[V^\pi(s') \mid s' \sim T(s, \pi(s))]$. The optimal value of a state
 125 $s \in S$, $V^*(s)$, is the value of state s when following the
 126 optimal policy π^* (the policy that maximizes the RL ob-
 127 jective (cf. definition 3.2)): $V^*(s) = V^{\pi^*}(s)$. Similarly,
 128 the optimal value of a state-action pair $(s, a) \in S \times A$,
 129 $Q^*(s, a)$, is the value when taking action a in state s and
 130 then following the optimal policy: $Q^*(s, a) = R(s, a) +$
 131 $\gamma \mathbb{E}[V^*(s') \mid s' \sim T(s, a)]$.

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

136 3.2. Iterative bounding Markov decision processes

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

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

167 **Definition 3.4** (Iterative bounding Markov decision process). Given an MDP $\mathcal{M} \equiv \langle S, A, R, T, T_0 \rangle$ (cf. definition 3.1), an associated iterative bounding Markov decision process \mathcal{M}_{IB} is a tuple:

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

168 S are the base MDP state features. Base state features
 169 $s = (s_1, \dots, s_p) \in S$ are bounded: $s_j \in [L_j, U_j]$ where
 170 $\infty < L_j \leq U_j < \infty \forall 1 \leq j \leq p$. O are observations.
 171 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 and observations. Given some 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

172 $s_{IB} = (\underbrace{s_1, \dots, s_p}_{\text{base state features}}, \underbrace{L_1, U_1, \dots, L_p, U_p}_{\text{observation}})$. A are the
 173 base MDP actions. A_{info} are *information gathering* ac-
 174 tions (IGAs) of the form $\langle j, v \rangle$ where j is a state feature
 175 index $1 \leq j \leq p$ and v is a real number between L_j
 176 and U_j . So the complete action space of an IBMDP is
 177 the set of base MDP actions and information gathering
 178 actions $A \cup A_{info}$. $R : S \times A \rightarrow \mathbb{R}$ is the base MDP
 179 reward function. ζ is a reward signal for taking an infor-
 180 mation gathering action. So the IBMDP reward function
 181 is to get a reward from the base MDP if the action is a
 182 base MDP action or to get ζ if the action is an IGA ac-
 183 tion. $T_{info} : S \times O \times (A_{info} \cup A) \rightarrow \Delta(S \times O)$ is the
 184 transition function of IBMDPs: given some observation
 185 $o_t = (L'_1, U'_1, \dots, L'_p, U'_p) \in O$ and base state features
 186 $s_t = (s'_1, s'_2, \dots, s'_p)$ if an IGA $\langle j, v \rangle$ is taken, the new
 187 observation is:

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

188 If a base action is taken, the observation is reset to the
 189 default base state feature bounds $(L_1, U_1, \dots, L_p, U_p)$ and
 190 the base state features change according to the base MDP
 191 transition function: $s_{t+1} \sim T(s_t, a_t)$. At initialization, the
 192 base state features are drawn from the base MDP initial dis-
 193 tribution T_0 and the observation is always set to the default
 194 base state features bounds $o_0 = (L_1, U_1, \dots, L_p, U_p)$.

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

198 3.3. From policies to trees

199 One can notice that information gathering actions (cf. def-
 200 inition 3.4) resemble the Boolean functions $1_{\{x_{-,j} \leq v\}}$ that
 201 make up internal decision tree nodes (cf. figure 10). Indeed,

165 **Algorithm 1** Extract a Decision Tree Policy (algorithm 1
 166 from (Topin et al., 2021))
 167 **Data:** Deterministic partially observable policy π_{po} for IB-
 168 MDP $\langle S \times O, A \cup A_{info}, (R, \zeta), (T_{info}, T, T_0) \rangle$ and
 169 IBMDP observation $\mathbf{o} = (L'_1, U'_1, \dots, L'_p, U'_p)$
 170 **Result:** Decision tree policy π_T for MDP $\langle S, A, R, T, T_0 \rangle$
 171 **Function** Subtree_From_Policy(\mathbf{o}, π_{po}):
 172 $a \leftarrow \pi_{po}(\mathbf{o})$
 173 **if** a is a base action **then**
 174 **return** Leaf_Node(action: a) // Leaf if base action
 175 **end**
 176 **else**
 177 $\langle i, v \rangle \leftarrow a$ // Splitting action is feature and value
 178 $\mathbf{o}_L \leftarrow \mathbf{o}; \quad \mathbf{o}_R \leftarrow \mathbf{o}$
 179 $\mathbf{o}_L \leftarrow (L'_1, U'_1, \dots, L'_j, v, \dots, L'_p, U'_p); \quad \mathbf{o}_R \leftarrow$
 180 $(L'_1, U'_1, \dots, v, U'_j, \dots, L'_p, U'_p)$
 181 $child_L \leftarrow \text{Subtree_From_Policy}(\mathbf{o}_L, \pi_{po})$
 182 $child_R \leftarrow \text{Subtree_From_Policy}(\mathbf{o}_R, \pi_{po})$
 183 **return** Internal_Node(feature: i , value: v , children:
 184 $(child_L, child_R)$)
 185 **end**

186
 187
 188 a policy taking actions in an IBMDP essentially builds a
 189 tree by taking sequences of IGAs (internal nodes) and then a
 190 base action (leaf node) and repeats this process over time. In
 191 particular, the IGA rewards ζ can be seen as a regularization
 192 or a penalty for interpretability: if ζ is very small compared
 193 to base rewards, a policy will try to take base actions as
 194 often as possible, i.e. build shallow trees with short paths
 195 between root and leaves.

196 Authors from (Topin et al., 2021) show that not all IBMDP
 197 policies are decision tree policies for the base MDP. In
 198 particular, their algorithm that converts IBMDP policies into
 199 decision trees (cf. algorithm 1) takes as input deterministic
 200 policies depending solely on the observations of the IBMDP.
 201

202 While the connections between partially observable MDPs
 203 (POMDPs (Sondik, 1978; Sigaud & Buffet, 2013)) and ex-
 204 tracting decision tree policies from IBMDPs is obvious,
 205 they are absent from the original IBMDP paper (Topin et al.,
 206 2021). In the next section we bridge this gap.

3.4. Bridging the gap with the partially observable MDPs literature

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

209 **Definition 3.5** (Partially observable Markov decision pro-
 210 cess). A partially observable Markov decision process is
 211 a tuple $\langle X, A, O, R, T, T_0, \Omega \rangle$. X is the hidden state space.
 212 A is a finite set of actions. O is a set of observations.
 213 $T : X \times A \rightarrow \Delta(X)$ is the transition function, where
 214 $T(\mathbf{x}_t, a, \mathbf{x}_{t+1}) = P(\mathbf{x}_t | \mathbf{x}_{t+1}, a)$ is the probability of tran-

sitioning 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 $\Omega(\mathbf{o}, a, \mathbf{x}) = P(\mathbf{o} | \mathbf{x}, a)$ is the probability of observing \mathbf{o} in state \mathbf{x} . $R : X \times A \rightarrow \mathbb{R}$ is the reward function, where $R(\mathbf{x}, a)$ is the immediate reward for taking action a in state \mathbf{x} . Note that $\langle X, A, R, T, T_0 \rangle$ defines an MDP.

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

Definition 3.6 (Partially observable iterative bounding Markov decision process). a partially observable iterative bounding Markov decision process \mathcal{M}_{POIB} is a tuple:

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

, where $\langle S \times O, A \cup A_{info}, (R, \zeta), (T, T_0, T_{info}) \rangle$ is an IBMDP (cf. definition 3.4). The transition function Ω maps concatenation of state features and observations–IBMDP states–to observations, $\Omega : S \times O \rightarrow O$, with $P(\mathbf{o} | (s, o)) = 1$

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

Definition 3.7 (Partially observable value function). In a POIBMDP (cf. definition 3.6), the expected cumulative discounted reward of a deterministic partially observable policy $\pi : O \rightarrow A \cup A_{info}$ starting from observation \mathbf{o} is $V^\pi(\mathbf{o})$:

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

with $P^\pi((s, o') | \mathbf{o})$ the asymptotic occupancy distribution (see section 4 (Singh et al., 1994) for the full definition) of the hidden POIBMDP state (s, o') given the partial observation \mathbf{o} and $V^\pi((s, o'))$ the classical state-value function (cf. definition 3.3). We abuse notation and denote both values of observations and values of states by V since the function input is not ambiguous.

220
 221
 222
 223
 224
 225
 226
 227
 The asymptotic occupancy distribution is the probability
 of a policy π to arrive in (s, o') while observing o in some
 trajectory. Now that we presented all the necessary technical
 backgrounds we move on to our contributions.
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237

4. Benchmarking RL for retrieving known decision tree policies

228
 229
 230
 231
 232
 233
 234
 235
 The goal of this section is to check if the direct approach
 described above can consistently retrieve optimal decision
 tree policies for a simple 2×2 grid world MDP. In particular,
 we use reinforcement learning to train decision tree poli-
 cies for MDPs by seeking deterministic partially observable
 policies that optimize the RL objective in POIBMDPs (cf.
 example ??).
 236
 237

4.1. Computing some decision tree policies

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
 To assess the performance of reinforcement learning, we
 identify decision tree policies that maximize the RL objec-
 tive in POIBMDPs with different trade-off rewards ζ for
 different discount factors γ . Each of those policies can be
 one of the trees illustrated in figure 1: (i) a depth-0 tree
 equivalent to always taking the same base action (π_{T_0}), (ii)
 a depth-1 tree equivalent alternating between an IGA and a
 base action (π_{T_1}), (iii) an unbalanced depth-2 tree that some-
 times takes two IGAs then a base action and sometimes a an
 IGA then a base action (π_{T_u}), (iv) a depth-2 tree that alter-
 nates between taking two IGAs and a base action (π_{T_2}), or
 (v) an infinite “tree” that only takes IGAs. Furthermore, be-
 cause from (Singh et al., 1994) we know that for POMDPs,
 stochastic partially observable policies can sometimes get
 better expected discounted rewards than deterministic par-
 tially observable policies, we also compute the value of the
 stochastic policy that randomly alternates between two base
 actions: \rightarrow and \downarrow . Taking those two base actions always
 lead to the goal state in expectation (cf. figure ??). Because
 we know all the base states, all the observations, all the
 actions, all the rewards and all the transitions of our POIB-
 MDP (cf. example 13), using definiton 3.7, we can compute
 exactly the values of those different deterministic partially
 observable policies given ζ the reward for IGAs and γ the
 discount factor.
 263
 264
 265
 266
 267
 268
 269
 270
 271

264
 265
 266
 267
 268
 269
 270
 We plot, in figure 2, the RL objective values of the decision
 tree policies as functions of ζ when we fix $\gamma = 0.99$ (stan-
 dard choice of discount in practice ??). When $\gamma = 0.99$,
 despite objective values being very similar for the depth-1
 and unbalanced depth-2 tree, we now know from the green
 shaded area that a depth-1 tree is the optimal one, w.r.t. the
 RL objective, deterministic partially observable POIBMDP
 policy for $0 < \zeta < 1$.
 271
 272
 273
 274

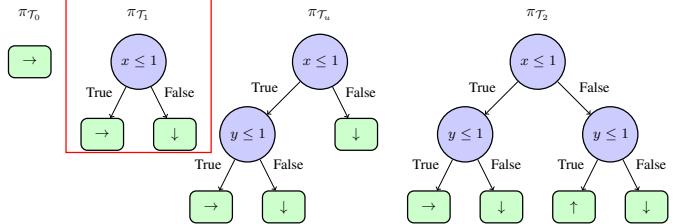


Figure 1. For each decision tree structure, e.g., depth-1 or unbal-
 anced depth-2, we illustrate a decision tree which maximizes the
 RL objective (cf. definition 3.2) in the grid world MDP.

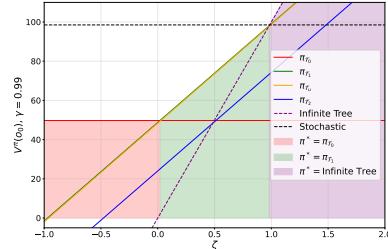


Figure 2. RL objective values (cf. definition ??) of different
 partially observable policies as functions of ζ . Shaded areas show
 the optimal *deterministic* partially observable policies in different
 ranges of ζ values.

4.2. Reinforcement learning algorithms

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

In particular, the problem of finding the optimal deterministic partially observable policies for POMDPs is NP-HARD, even with full knowledge of transitions and rewards (?)section 3.2]littman1. It means that it is impractical to enumerate all possible policies and take the best one. For even moderate-sized POMDPs, a brute-force approach would take a very long time since there are $|A|^{|O|}$ deterministic partially observable policies. Hence it is interesting to study reinforcement learning for finding the best deterministic partially observable policy since it would not search the whole solution space. However applying RL to our interpretable RL objective (cf. definition ??) is non-trivial.

In (Singh et al., 1994), the authors show that the optimal partially observable policy can be stochastic. Hence, policy gradient algorithms (Sutton et al., 1999)—that return stochastic policies—are to avoid since we seek the best *deterministic*

275 policy. Furthermore, the optimal deterministic partially ob-
 276 servable policy might not maximize all the values of all ob-
 277 servations simultaneously (Singh et al., 1994) which makes
 278 it difficult to use TD-learning (cf. algorithms 3 and ??).
 279 Indeed, doing a TD-learning update of one partially observ-
 280 able value (cf. definition 3.7) with, e.g. Q-learning, can
 281 change the value of *all* other observations in an uncontrol-
 282 lable manner because of the dependence in $P^\pi((s, o')|o)$
 283 (cf. definition 3.7). Interestingly, those two challenges of
 284 learning in POMDPs described in (Singh et al., 1994) are
 285 visible in figure 2. First, there is a whole range of ζ values
 286 for which the optimal partially observable policy is stochas-
 287 tic. Second, for e.g. $\zeta = 0.5$, while a depth-1 tree is the
 288 optimal deterministic partially observable policy, the value
 289 of state $(s_2, o_0) = (1.5, 1.5, 0, 2, 0, 2)$ is not maximized
 290 by this partially observable policy but by the sub-optimal
 291 policy that always goes down.

292 Despite those hardness results, empirical results of applying
 293 RL to POMDPs by naively replacing x by o in Q-learning or
 294 Sarsa, has already demonstrated successful in practice (Loch
 295 & Singh, 1998). More recently, the framework of Baisero
 296 et. al. called asymmetric RL (Baisero et al., 2022; Baisero &
 297 Amato, 2022) has also shown promising results to
 298 learn POMDP solutions. Asymmetric RL algorithms train
 299 a model—a policy or a value function—depending on hidden
 300 state (only available at train time) and a history dependent
 301 (or observation dependent) model. The history or observa-
 302 tion dependent model serves as target or critic to train the
 303 hidden state dependent model. The history dependent (or
 304 observation dependent) model can thus be deployed in the
 305 POMDP after training since it does not require access to the
 306 hidden state to output actions. In algorithm 2 we present
 307 asymmetric Q-learning. It is a variant of Q-learning (cf.
 308 algorithm 3) that returns a deterministic partially observ-
 309 able policy like modified DQN 4. Given a POMDP, asym-
 310 metric Q-learning trains a partially observable Q-function
 311 $Q : O \times A \rightarrow \mathbb{R}$ and a Q-function $U : X \times A \rightarrow \mathbb{R}$.
 312 The hidden state dependent Q-function U serves as a tar-
 313 get in the temporal difference learning update. We also
 314 consider an asymmetric version of Sarsa that applies sim-
 315 ilar modifications to the standard Sarsa (cf. algorithm ??).
 316 We present asymmetric Sarsa in the appendix (cf. algo-
 317 rithm ??). In (Jaakkola et al., 1994), the authors introduce a
 318 policy search algorithm A.2 that learns a (stochastic) policy
 319 $\pi : O \rightarrow \Delta(A)$ and a critic $V : X \rightarrow \mathbb{R}$ using Monte Carlo
 320 estimates to guide policy improvement. We also consider
 321 this algorithm in our experiments that we call JSJ (for the
 322 authors names Jaakkola, Singh, Jordan). We present the JSJ
 323 algorithm in the appendix (cf. algorithm ??). JSJ is equiv-
 324 alent to a tabular asymmetric policy gradient algorithm (cf.
 325 algorithm ??).

326 Until recently, the benefits of asymmetric RL over stan-
 327 dard RL was only shown empirically and only for history-
 328

329 dependent models. The work of Gaspard Lambrechts (Lam-
 330 brechts et al., 2025b) proves that some asymmetric RL algo-
 331 rithms learn better history-dependent **or** partially observable
 332 policies for solving POMDPs. This is exactly what we wish
 333 for. However, those algorithms are not practical because
 334 they require estimations of the asymptotic occupancy dis-
 335 tribution $P^\pi((s, o')|o)$ (cf. definition 3.7) for candidate
 336 policies which in turn would require to gather a lot of on-
 337 policy samples. We leave it to future work to use those
 338 algorithms that combine asymmetric RL and estimation of
 339 future visitation frequencies since those results are contem-
 340 porary to the writing of this manuscript.

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

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

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 $Q(o, a) = 0$ for all $o \in O, a \in A$

for each episode **do**

- Initialize state $x_0 \sim T_0$
- Initialize observation $o_0 \sim \Omega(x_0)$
- for** each step t **do**
- Choose action a_t using ϵ -greedy: $a_t = \text{argmax}_a Q(o_t, a)$ 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 $o_{t+1} \sim \Omega(x_{t+1})$
- $y \leftarrow r + \gamma U(\mathbf{x}_{t+1}, \text{argmax}_{a'} Q(o_{t+1}, a'))$ // TD target
- $U(\mathbf{x}_t, a_t) \leftarrow (1 - \alpha_u)U(\mathbf{x}_t, a_t) + \alpha_u y$
- $Q(o_t, a_t) \leftarrow (1 - \alpha_q)Q(o_t, a_t) + \alpha_q y$
- $x_t \leftarrow x_{t+1}$
- $o_t \leftarrow o_{t+1}$

end

end

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

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

Results

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

330 Let us understand why.

331 **4.2.1. EXPERIMENTAL SETUP**

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

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

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

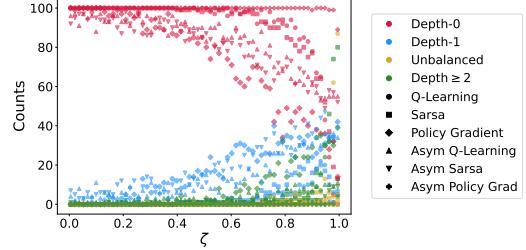
349 **Metrics:** We will consider two metrics. First, the sub-
 350 optimality gap during training, w.r.t. the interpretable RL
 351 objective, between the learned partially observable policy
 352 and the optimal deterministic partially observable policy:
 353 $|\mathbb{E}[V^{\pi^*}(s_0, o_0)|s_0 \sim T_0] - \mathbb{E}[V^\pi(s_0, o_0)|s_0 \sim T_0]|$. Because we know the whole POIBMDP model that we can
 354 represent exactly as tables, and because we know for each
 355 ζ the interpretable RL objective value of the optimal deter-
 356 ministic partially observable policy (cf. figure 2), we can
 357 report the *exact* sub-optimality gaps.

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

367 **4.3. Can (asymmetric) RL learn optimal deterministic
 368 partially observable POIBMDP policies?**

369 In figure 3, we plot the sub-optimality gaps—averaged over
 370 100 seeds—of learned policies during training. We do so
 371 for 200 different POIBMDPs where we change the reward
 372 for information gathering actions: we sample 200 ζ values
 373 uniformly in $[-1, 2]$. In figure 3, a different color represents
 374 a different POIBMDP.

375 Recall from figure 2 that for: (i) $\zeta \in [-1, 0]$, the optimal
 376



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

381 deterministic partially observable policy is a depth-0 tree,
 382 (ii) $\zeta \in]0, 1[$, the optimal deterministic partially observable
 383 policy is a depth-1 tree, and (iii) $\zeta \in [1, 2]$, the optimal de-
 384 terministic partially observable policy is a “infinite” tree that
 385 contains infinite number of internal nodes. We observe that,
 386 despite all sub-optimality gaps converging independently of
 387 the ζ values, not all algorithms in all POIBMDPs fully mini-
 388 mize the sub-optimality gap. In particular, all algorithms
 389 seem to consistently minimize the gap, i.e. learn the optimal
 390 policy or Q-function, only for $\zeta \in [1, 2]$ (all the yellow lines
 391 go to 0). However, we are interested in the range $\zeta \in]0, 1[$
 392 where the optimal decision tree policy is non-trivial, i.e. not
 393 taking the same action forever. In that range, no baseline
 394 consistently minimizes the sub-optimality gap.

395 In figure 4, we plot the distributions of the final learned trees
 396 over the 100 random seeds in function of ζ from the above
 397 runs. For example, in figure 4, in the top left plot, when
 398 learning 100 times in a POIBMDP with $\zeta = 0.5$, Q-learning
 399 returned almost 100 times a depth-0 tree. Again, on none
 400 of those subplots do we see a high rate of learned depth-
 401 trees for $\zeta \in]0, 1[$. It is alerting that the most frequent
 402 learned trees are the depth-0 trees for $\zeta \in]0, 1[$ because such
 403 trees are way more sub-optimal w.r.t. the interpretable RL
 404 objective (cf. definition ??) than e.g. the depth-2 unbalanced
 405 trees (cf. figure 2). One interpretation of this phenomenon
 406 is that the learning in POIBMDPs is very difficult and so
 407 agents tend to converge to trivial policies, e.g., repeating the
 408 same base action.

409 However, on the positive side, we observe that asymmetric
 410 versions of Q-learning and Sarsa have found the optimal
 411 deterministic partially observable policy—the depth-1 deci-
 412 sion tree—more frequently throughout the optimality range
 413 $]0, 1[$, than their symmetric counter-parts for $\zeta \in]0, 1[$. Next,
 414 we quantify how difficult it is to do RL to learn partially
 415 observable policies in POIBMDPs.

385	4.4. How difficult is it to learn in POIBMDPs?
386	In this section we run the same (asymmetric) reinforcement learning algorithms to optimize either the RL objective (cf. definition 3.2) in MDPs (cf. definition 3.1) or IBMDPs (cf. definition 3.4), or the interpretable RL objective in POIBMDPs(cf. definition ??). This essentially results in three distinct problems:
387	
388	1. Learning an optimal standard Markovian policy in an MDP, i.e. optimizing the RL objective in an MDP.
389	
390	2. Learning an optimal standard Markovian policy in an IBMDP, i.e. optimizing the RL objective in an IBMDP.
391	
392	3. Learning an optimal deterministic partially observable policy in a POIBMDP.
393	
394	
395	
396	
397	
398	
399	In order to see how difficult each of these three problems is, we can run a <i>great</i> number of experiments for each problem and compare solving rates. To make solving rates comparable we consider a unique instance for each of those problems. Problem 1 is learning one of the optimal standard Markovian deterministic policy from figure ?? for the grid world from example ?? with $\gamma = 0.99$. Problem 2 is learning one of the optimal standard Markovian deterministic for the IBMDP from figure 13 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 14b). 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 $\zeta = 0.5$.
400	
401	
402	
403	
404	
405	
406	
407	
408	
409	
410	
411	
412	
413	
414	
415	
416	
417	
418	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 ??.
419	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.
420	
421	
422	
423	
424	
425	
426	
427	
428	
429	
430	
431	
432	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
433	
434	
435	
436	
437	
438	
439	

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

Table 2. Summary of RL baselines Hyperparameters

algorithm	Problem	Total Hyperparameter Combinations
Policy Gradient	PO/IB/MDP	420
JSJ	POIBMDP	15
Q-learning	PO/IB/MDP	192
Asym Q-learning	POIBMDP	768
Sarsa	PO/IB/MDP	192
Asym Sarsa	POIBMDP	768

$\gamma = 0.99$ and $\zeta = 0.5$, the optimal policy is a depth-1 tree (e.g. figure 1) as per figure 2).

The key observations from figure 5 is that reinforcement learning a deterministic partially observable policy in a POIBMDP, is way harder than learning a standard Markovian policy. For example, Q-learning only finds the optimal solution (cf. definition ??) in only 3% of the experiments while the same algorithms to optimize the standard RL objective (cf. definition 3.2) in an MDP or IBMDP found the optimal solutions 50% of the time. Even though asymmetry seems to increase performances; learning a decision tree policy for a simple grid world directly with RL using the framework of POIBMDP originally developed in (Topin et al., 2021) seems way too difficult and costly as successes might require a million steps for such a seemingly simple problem. An other difficulty in practice that we did not cover here, is the choice of information gathering actions. For the grid world MDP, choosing good IGAs ($x \leq 1$ and $y \leq 1$) is simple but what about more complicated MDPs: how to instantiate the (PO)IBMDP action space such that internal nodes in resulting trees are useful for predictions?

To go even further, on figure 6 we re-run experiments from figure 3 and figure 4 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 6 despite higher success rates in the region $\zeta \in]0, 1[$ compared to figure 4.

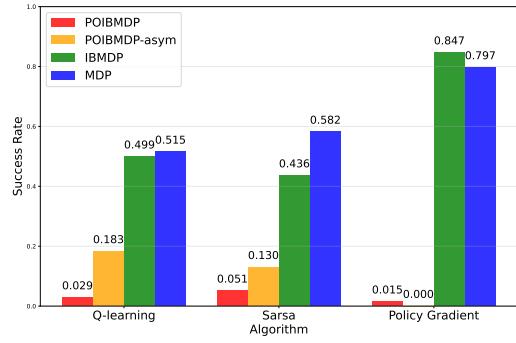


Figure 4. Success rates of different (asymmetric) RL algorithms over thousands of runs when applied to learning deterministic partially observable policies in a POIBMDP or learning deterministic policies in associated MDP and IBMDP.

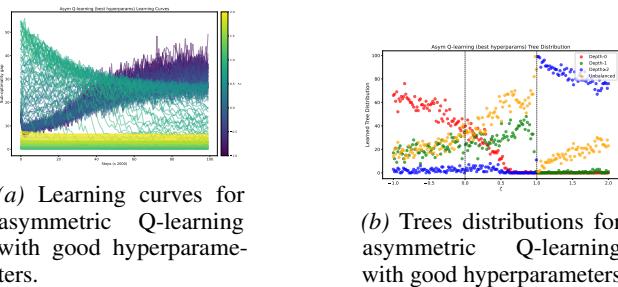


Figure 5. Analysis of the top-performing asymmetric Q-learning instantiation. (left) Learning curves, and (right) tree distributions across different POIBMDP configurations.

5. Conclusion

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

Across our experiments, we found that no algorithm could consistently learn a depth-1 decision tree policy for a grid world MDP despite it being optimal both w.r.t. the interpretable RL objective and standard RL objective (cf. definition 3.2). When compared to the results of VIPER from figure 12, direct RL is worse at retrieving decision tree policies with good interpretability-performance trade-offs. This echoes the results from the previous chapter in which we saw that direct deep RL performed worse than imitation learning to find decision tree policies for CartPole.

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

6. Classification tasks

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

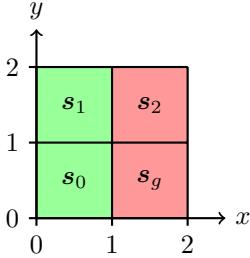


Figure 6. Classification MDP optimal actions. In this classification MDP, there are four data to which to assign either a green or red label. On the right, there is the unique optimal depth-1 tree for this particular classification MDP. This depth-1 tree also maximizes the accuracy on the corresponding classification task.

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

Definition 6.1 (Classification Markov decision process). Given a set of N examples denoted $\mathcal{E} = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^N$ 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 3.1) 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 $T_0(\mathbf{s}_0 = \mathbf{x}) = \frac{1}{N}$

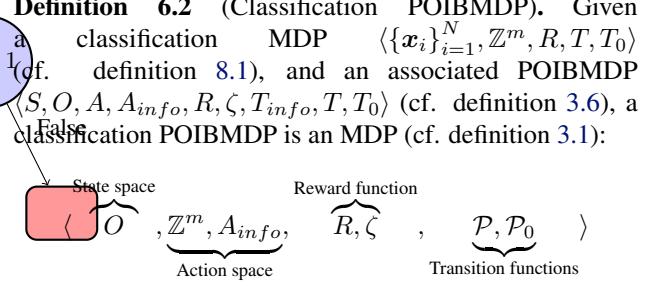
One can be convinced that policies that maximize the RL objective (cf. definition 3.2) in classification MDPs are classifiers that maximize the prediction accuracy because $\sum_{i=1}^N 1_{\pi(\mathbf{x}_i)=y_i} = \sum_{i=1}^N R(\mathbf{x}_i, \pi(\mathbf{x}_i))$. We defer the formal proof in the next part of the manuscript in which we extensively study supervised learning problems.

In figure 7 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.

10



- 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 : $R(o, a) = \frac{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall i, j\} \cap \{\mathbf{x}_i : a_i = a \forall i\}|}{|\{\mathbf{x}_i : L'_j \leq x_{ij} \leq U'_j \forall i, j\}|}$. 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 ??) $O \rightarrow A \cup A_{info}$ are in fact Markovian policy for those classification POIBMDPs. More importantly, it means that, for a given γ and ζ , if we were to know the whole POIBMDP model, we could use planning, e.g. value iteration (cf. definition ??), to compute *optimal* decision tree policies. Similarly, standard RL algorithms like Q-learning (cf. definition A.2) should work as well as for any MDP to learn optimal decision tree policies.

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

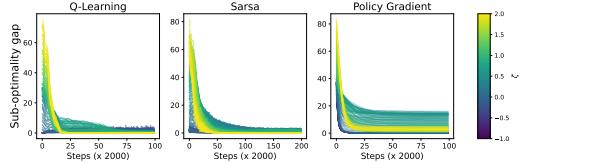


Figure 7. We reproduce the same plot as in figure 3 for classification POIBMDPs. Each individual curve is the sub-optimality gap of the learned policy during training averaged over 100 runs for a single ζ value.

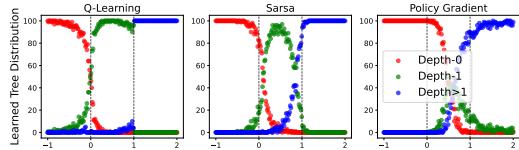


Figure 8. We reproduce the same plot as in figure 4 for classification POIBMDPs. Each colored dot is the number of final learned trees with a specific structure for a given ζ .

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

Similarly to the previous chapter, we are interested in a very simple classification POIBMDP. We study classification POIBMDPs associated with the example classification MDP from figure 7.

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

Fortunately this time, compared to general POIBMDPs, RL can be used to learn optimal deterministic partially observable policies $O : \rightarrow A \cup A_{info}$ w.r.t. the interpretable RL objective (cf. definition ??) in classification POIBMDPs. Such policies are equivalent to decision tree classifiers. We observe on figure 8 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 7) most of the time in the optimality range $\zeta \in]0, 1[$ (cf. figure 9).

8. Conclusion

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

In chapter ??, we compared the algorithms proposed in (Topin et al., 2021) that directly optimize this objective, to imitation learning algorithms that only solve a proxy problem. While those direct RL algorithms are able to *learn*, i.e. find better and better solutions with time (cf. figures 14a and 14b), the decision tree policies returned perform worse in average than imitated decision trees w.r.t. the RL objective of interest (cf. figure 15) for similar number of nodes and depth.

We further analyzed the failure mode of direct learning of decision tree policies by making connections with POMDPs (Sondik, 1978; Sigaud & Buffet, 2013). In chapter ??, we showed that learning decision tree policies for MDPs could be explicitly formulated as learning a deterministic partially observable (also known as memoryless or reactive) policy in a specific POMDP that we called POIBMDP (cf. definition 3.6). We showed that both RL and asymmetric RL, a class of algorithms specifically designed for POMDPs (Baisero & Amato, 2022; Baisero et al., 2022), were unable to consistently learn an optimal depth-1 decision tree policies for a very small grid world MDP when using the POIBMDP framework. In particular, we compared, in a very controlled experiment, the success rates of the same learning algorithms when seeking standard Markovian policies versus partially observable policies in decision processes that shared the same transitions and rewards (cf. section 6.3). We demonstrated on figure 5 that introducing partial observability greatly reduced the success rates (we also observed this implicitly on figures 14a and 14b).

Finally, in this chapter we showed that using RL to optimize the interpretable RL objective in fully observable POIBMDPs, i.e. POIBMDPs that are just MDPs, could learn optimal decision tree policies (cf. figures 8 and 9) adding new evidence that direct interpretable RL is difficult because it involves POMDPs.

This class of fully observable POIBMDPs (cf. definition 8.2) contains the decision tree induction problem for supervised learning tasks (cf. definition ??). This sparks the question: what kind of decision tree induction algorithm can we get using the MDP formalism? This is exactly what we study in the next part of this manuscript.

Those few chapters raise other interesting questions. We focused on non-parametric tree learning because RL algorithms can learn decision tree policies with potentially optimal interpretability-performance trade-offs through the reward of information gathering actions in (PO)IBMDPs (cf. definitions 3.4 and 3.6). However this comes at a cost of partial observability which makes learning difficult. Parametric tree policies on the other hand, can be computed with reinforcement learning directly in the base MDP. However existing RL algorithms for parametric decision tree policies (Silva et al., 2020; Vos & Verwer, 2024; Marton et al.,

605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
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
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
24

- 660 Glanois, C., Weng, P., Zimmer, M., Li, D., Yang, T., Hao,
 661 J., and Liu, W. A survey on interpretable reinforcement
 662 learning. *Machine Learning*, pp. 1–44, 2024.
- 663 Jaakkola, T., Singh, S. P., and Jordan, M. I. Reinforcement
 664 learning algorithm for partially observable markov deci-
 665 sion problems. In *Proceedings of the 8th International*
 666 *Conference on Neural Information Processing Systems*,
 667 NIPS’94, pp. 345–352, Cambridge, MA, USA, 1994.
 668 MIT Press.
- 669 Lambrechts, G., Bolland, A., and Ernst, D. Informed
 670 POMDP: Leveraging additional information in model-
 671 based RL. *Reinforcement Learning Journal*, 2:763–784,
 672 2025a.
- 673 Lambrechts, G., Ernst, D., and Mahajan, A. A the-
 674 oretical justification for asymmetric actor-critic algo-
 675 rithms. In *Forty-second International Conference on Ma-*
 676 *chine Learning*, 2025b. URL <https://openreview.net/forum?id=F1yANMCnAn>.
- 677 Lipton, Z. C. The mythos of model interpretability: In
 678 machine learning, the concept of interpretability is both
 679 important and slippery. *Queue*, 16(3):31–57, 2018.
- 680 Littman, M. L. Memoryless policies: theoretical limitations
 681 and practical results. In *Proceedings of the Third Inter-*
 682 *national Conference on Simulation of Adaptive Behavior:*
 683 *From Animals to Animats 3: From Animals to Animats 3*,
 684 SAB94, pp. 238–245, Cambridge, MA, USA, 1994. MIT
 685 Press. ISBN 0262531224.
- 686 Loch, J. and Singh, S. P. Using eligibility traces to find the
 687 best memoryless policy in partially observable markov
 688 decision processes. In *Proceedings of the Fifteenth Inter-*
 689 *national Conference on Machine Learning*, ICML ’98,
 690 pp. 323–331, San Francisco, CA, USA, 1998. Morgan
 691 Kaufmann Publishers Inc. ISBN 1558605568.
- 692 Lundberg, S. M. and Lee, S.-I. A unified approach to inter-
 693 pretting model predictions. In *Proceedings of the 31st Inter-*
 694 *national Conference on Neural Information Processing*
 695 *Systems*, NIPS’17, pp. 4768–4777, Red Hook, NY, USA,
 696 2017. Curran Associates Inc. ISBN 9781510860964.
- 697 Mania, H., Guy, A., and Recht, B. Simple random search
 698 of static linear policies is competitive for reinforcement
 699 learning. In *Proceedings of the 32nd International*
 700 *Conference on Neural Information Processing Systems*,
 701 NIPS’18, pp. 1805–1814, Red Hook, NY, USA, 2018.
 702 Curran Associates Inc.
- 703 Mansour, Y., Moshkovitz, M., and Rudin, C. There is no
 704 accuracy-interpretability tradeoff in reinforcement learn-
 705 ing for mazes, 2022. URL <https://arxiv.org/abs/2206.04266>.
- 706 Marton, S., Grams, T., Vogt, F., Lüdtke, S., Bartelt, C.,
 707 and Stuckenschmidt, H. Mitigating information loss in
 708 tree-based reinforcement learning via direct optimization.
 709 2025. URL <https://openreview.net/forum?id=qpXctF2aLZ>.
- 710 Milani, S., Topin, N., Veloso, M., and Fang, F. Explainable
 711 reinforcement learning: A survey and comparative review.
 712 *ACM Comput. Surv.*, 56(7), April 2024. ISSN 0360-0300.
 713 doi: 10.1145/3616864. URL <https://doi.org/10.1145/3616864>.
- 714 Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness,
 715 J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidje-
 716 land, A. K., Ostrovski, G., et al. Human-level control
 717 through deep reinforcement learning. *nature*, 518(7540):
 718 529–533, 2015.
- 719 Murthy, S. and Salzberg, S. Lookahead and pathology
 720 in decision tree induction. In *Proceedings of the 14th*
 721 *International Joint Conference on Artificial Intelligence -*
 722 *Volume 2*, IJCAI’95, pp. 1025–1031, San Francisco, CA,
 723 USA, 1995. Morgan Kaufmann Publishers Inc. ISBN
 724 1558603638.
- 725 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V.,
 726 Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P.,
 727 Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cour-
 728 napeau, D., Brucher, M., Perrot, M., and Duchesnay, E.
 729 Scikit-learn: Machine learning in Python. *Journal of*
 730 *Machine Learning Research*, 12:2825–2830, 2011.
- 731 Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W.,
 732 and Abbeel, P. Asymmetric actor critic for image-based
 733 robot learning, 2017. URL <https://arxiv.org/abs/1710.06542>.
- 734 Puri, N., Verma, S., Gupta, P., Kayastha, D., Deshmukh,
 735 S., Krishnamurthy, B., and Singh, S. Explain your move:
 736 Understanding agent actions using specific and
 737 relevant feature attribution. In *International Confer-*
 738 *ence on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SJgzLkBKPB>.
- 739 Puterman, M. L. *Markov Decision Processes: Discrete*
740 Stochastic Dynamic Programming. John Wiley & Sons,
 741 1994.
- 742 Quinlan, J. R. Induction of decision trees. *Mach. Learn.*, 1
 743 (1):81–106, 1986.
- 744 Quinlan, J. R. C4. 5: Programs for machine learning. *Mor-*
 745 *gan Kaufmann google schola*, 2:203–228, 1993.
- 746 Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus,
 747 M., and Dormann, N. Stable-baselines3: Reliable rein-
 748 force learning implementations. *Journal of Machine*
 749 *Learning Research*, 22(268):1–8, 2021.

- 715 Ribeiro, M. T., Singh, S., and Guestrin, C. "why
716 should i trust you?": Explaining the predictions of
717 any classifier. pp. 1135–1144, 2016. doi: 10.1145/
718 2939672.2939778. URL <https://doi.org/10.1145/2939672.2939778>.
- 719
- 720 Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of
721 imitation learning and structured prediction to no-regret
722 online learning. 2010.
- 723
- 724 Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and
725 Klimov, O. Proximal policy optimization algorithms.
726 *arXiv preprint arXiv:1707.06347*, 2017.
- 727
- 728 Shi, W., Huang, G., Song, S., Wang, Z., Lin, T., and Wu, C.
729 Self-supervised discovering of interpretable features for
730 reinforcement learning. *IEEE Transactions on Pattern
731 Analysis and Machine Intelligence*, 44(5):2712–2724,
732 2022. doi: 10.1109/TPAMI.2020.3037898.
- 733
- 734 Shindo, H., Delfosse, Q., Dhami, D. S., and Kersting, K.
735 Blendrl: A framework for merging symbolic and neural
736 policy learning. *arXiv*, 2025.
- 737
- 738 Sigaud, O. and Buffet, O. *Partially Observable Markov
739 Decision Processes*, chapter 7, pp. 185–228. John
740 Wiley Sons, Ltd, 2013. ISBN 9781118557426. doi:
741 <https://doi.org/10.1002/9781118557426.ch7>. URL
742 <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118557426.ch7>.
- 743
- 744 Silva, A., Gombolay, M., Killian, T., Jimenez, I., and Son,
745 S.-H. Optimization methods for interpretable differentiable
746 decision trees applied to reinforcement learning.
747 In Chiappa, S. and Calandra, R. (eds.), *Proceedings of
748 the Twenty Third International Conference on Artificial
749 Intelligence and Statistics*, volume 108 of *Proceedings of
750 Machine Learning Research*, pp. 1855–1865. PMLR, 26–
751 28 Aug 2020. URL <https://proceedings.mlr.press/v108/silva20a.html>.
- 752
- 753
- 754 Singh, S. P., Jaakkola, T. S., and Jordan, M. I. Learning
755 without state-estimation in partially observable markovian
756 decision processes. In *Proceedings of the Eleventh
757 International Conference on International Conference
758 on Machine Learning*, ICML'94, pp. 284–292, San
759 Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
760 ISBN 1558603352.
- 761
- 762 Sondik, E. J. The optimal control of partially observable
763 markov processes over the infinite horizon: Discounted
764 costs. *Operations Research*, 26(2):282–304, 1978. ISSN
765 0030364X, 15265463. URL <http://www.jstor.org/stable/169635>.
- 766
- 767 Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An
768 Introduction*. The MIT Press, Cambridge, MA, 1998.
- 769
- 770 Sutton, R. S., McAllester, D., Singh, S., and Mansour,
771 Y. Policy gradient methods for reinforcement learning
772 with function approximation. In Solla, S., Leen,
773 T., and Müller, K. (eds.), *Advances in Neural Infor-
774 mation Processing Systems*, volume 12. MIT Press,
775 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.
- 776
- Tesauro, G. Temporal difference learning and td-gammon.
Commun. ACM, 38(3):58–68, March 1995. ISSN 0001-0782. doi: 10.1145/203330.203343. URL <https://doi.org/10.1145/203330.203343>.
- 777
- Topin, N., Milani, S., Fang, F., and Veloso, M. Iterative
778 bounding mdps: Learning interpretable policies via non-
779 interpretable methods. *Proceedings of the AAAI Confer-
780 ence on Artificial Intelligence*, 35:9923–9931, 2021.
- 781
- Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola,
782 G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M.,
783 KG, A., et al. Gymnasium: A standard interface for
784 reinforcement learning environments. *arXiv preprint
785 arXiv:2407.17032*, 2024.
- 786
- van der Linden, J., de Weerdt, M., and Demirović, E. Nec-
787 essary and sufficient conditions for optimal decision trees
788 using dynamic programming. *Advances in Neural Infor-
789 mation Processing Systems*, 36:9173–9212, 2023.
- 790
- Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri,
791 S. Programmatically interpretable reinforcement learning.
792 pp. 5045–5054, 2018.
- 793
- Verwer, S. and Zhang, Y. Learning optimal classification
794 trees using a binary linear program formulation. *Proced-
795 ings of the AAAI conference on artificial intelligence*, 33:
796 1625–1632, 2019.
- 797
- Vos, D. and Verwer, S. Optimal decision tree policies
798 for markov decision processes. In *Proceedings of the
799 Thirty-Second International Joint Conference on Artifi-
800 cial Intelligence*, IJCAI '23, 2023. ISBN 978-1-956792-
801 03-4. doi: 10.24963/ijcai.2023/606. URL <https://doi.org/10.24963/ijcai.2023/606>.
- 802
- Vos, D. and Verwer, S. Optimizing interpretable decision
803 tree policies for reinforcement learning. 2024. URL
804 <https://arxiv.org/abs/2408.11632>.
- 805
- Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*,
806 8(3):279–292, 1992.
- 807
- Wu, H., Isac, O., Zeljić, A., Tagomori, T., Daggitt, M.,
808 Kokke, W., Refaeli, I., Amir, G., Julian, K., Bassan,
809 S., Huang, P., Lahav, O., Wu, M., Zhang, M., Komen-
810 dantskaya, E., Katz, G., and Barrett, C. Marabou 2.0: A

770 versatile formal analyzer of neural networks, 2024. URL
771 <https://arxiv.org/abs/2401.14461>.
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

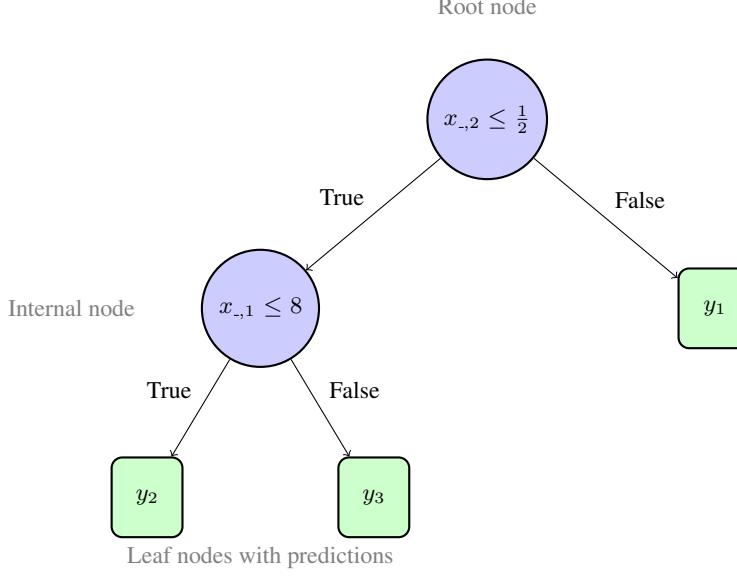


Figure 9. A generic depth 2 decision tree with 2 nodes and 3 leaves. The root node applies the test $1_{\{x_{-,1} \leq \frac{1}{2}\}}$ to check if the first features of data is below $\frac{1}{2}$. Edges represent the outcomes of the tests in each internal nodes(True/False), and leaf nodes contain predictions $y_l \in \mathcal{Y}$. 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 10.

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

A.2. Reinforcement learning of approximate solutions to MDPs

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

880 policies or values (Tesauro, 1995). In the next section, we present deep reinforcement learning algorithms designed
 881 specifically for neural networks.
 882

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

884 **Data:** MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, learning rate α , exploration rate ϵ
 885 **Result:** Policy π
 886 Initialize $Q(s, a) = 0$ for all $s \in S, a \in A$
 887 Initialize state $s_0 \sim T_0$
 888 **for** each step t **do**
 889 Choose action a_t using e.g. ϵ -greedy policy: $a_t = \text{argmax}_a Q(s_t, a)$ with prob. $1 - \epsilon$
 890 Take action a_t , observe $r_t = R(s_t, a_t)$ and $s_{t+1} \sim T(s_t, a_t)$
 891 $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$
 892 $s_t \leftarrow s_{t+1}$
 893 **end**
 894 $\pi(s) = \text{argmax}_a Q(s, a)$ // Extract greedy policy

A.3. Deep reinforcement learning

900 Reinforcement learning has also been successfully combined with function approximations to solve MDPs with large
 901 discrete state spaces or continuous state spaces ($S \subset \mathbb{R}^p$ in definition 3.1). In the rest of this manuscript, unless stated
 902 otherwise, we write s a state vector in a continuous state space¹.

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

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

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

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

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

922 **Definition A.2** (Imitation learning objective). Given an MDP \mathcal{M} (cf. definition 3.1), an expert policy π^* and a policy class
 923 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
 924 expected action disagreement with the expert:

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

926 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
 927 between the student policy's action $\pi(s)$ and the expert's action $\pi^*(s)$.
 928

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

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

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

A.5. Your first decision tree policy

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

In this section we present the first decision tree policies of this manuscript obtained using Dagger or VIPER after learning an expert Q-function for the grid world MDP from figure ?? using Q-learning (cf. algorithm 3). Recall the optimal policies for the grid world, taking the green actions in each state in figure ???. Among the optimal policies, the ones that go left or up in the goal state can be problematic for imitation learning algorithms. Indeed, we know that for this grid world MDP there exists decision tree policies with a very good interpretability-performance trade-off: depth-1 decision trees that are optimal w.r.t. the RL objective. One could even say that those trees have the *optimal* interpretability-performance trade-off because they are the shortest trees that are optimal w.r.t. the RL objective.

In figure 11, we present a depth-1 decision tree policy that is optimal w.r.t. the RL objective and a depth-1 tree that is sub-optimal. The other optimal depth-1 tree is to go right when $y \leq 1$ and down otherwise. Indeed, figure ?? shows that the optimal depth-1 tree achieves exactly the same RL objective value as the optimal policies from figure ??, independently of the discount factor γ .

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

We start by running the standard Q-learning algorithm as presented in algorithm 3 with $\epsilon = 0.3$, $\alpha = 0.1$ over 10,000 time steps. The careful reader might wonder how ties are broken in the argmax operation from algorithm 3. While Sutton and Barto break ties by index value in their book (Sutton & Barto, 1998) (the greedy action is the argmax action with smallest index), we show that the choice of tie-breaking greatly influences the performance of subsequent imitation learning algorithms. Indeed, depending on how actions are ordered in practice, Q-learning may be biased toward some optimal policies rather than others. While this does not matter for one who just wants to find an optimal policy, in our example of finding the optimal depth-1 decision tree policy, it matters *a lot*.

In the left plot of figure 12, 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 12, where we plot the proportion over 100 runs of optimal decision trees returned by Dagger or VIPER at different stages of Q-learning, we observe that imitating the optimal policy obtained by breaking ties at random consistently yields more optimal trees than breaking ties by indices. What actually happens is that the most likely output of Q-learning when ties are broken by indices is the optimal policy that goes left in the goal state, which cannot be perfectly represented by a depth-1 decision tree, because there are three different actions taken and a binary tree of depth $D = 1$ can only map to $2^D = 2$ labels.

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

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

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

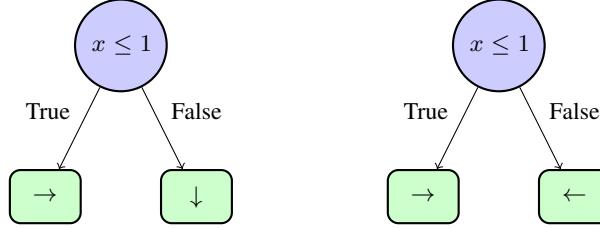


Figure 10. Left, an optimal depth-1 decision tree policy. On the right, a sub-optimal depth-1 decision tree policy.

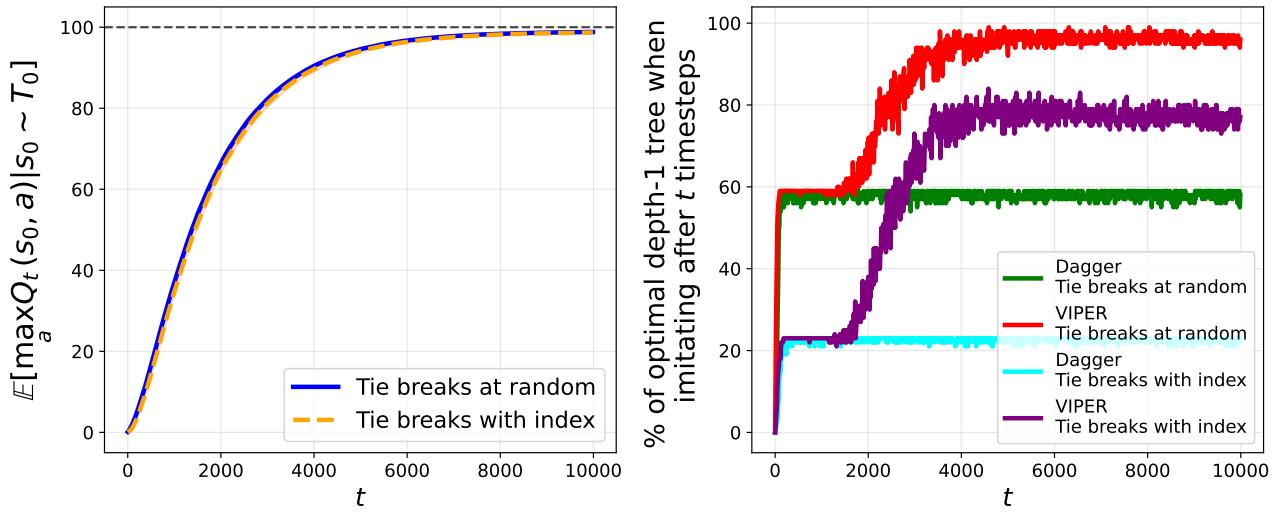


Figure 11. Left, sample complexity curve of Q-learning with default hyperparameters on the 2×2 grid world MDP over 100 random seeds. Right, performance of indirect interpretable methods when imitating the greedy policy with a tree at different Q-learning stages.

Limits of RL for decision tree policies

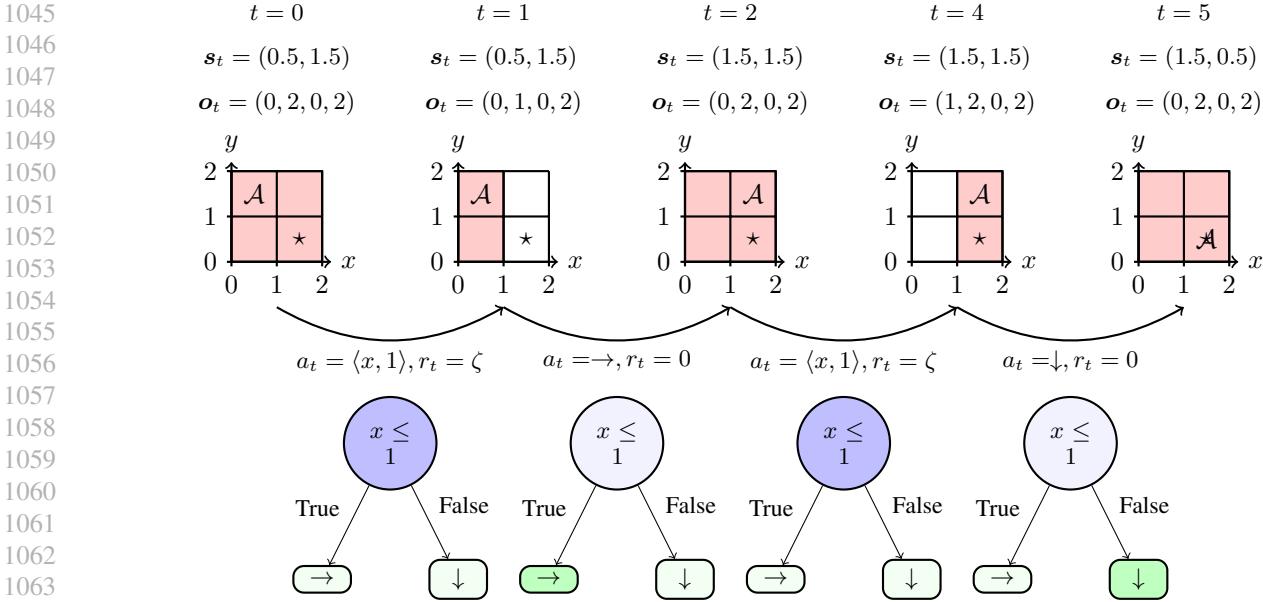


Figure 12. An IBMDP trajectory when the base MDP is 2×2 grid world. In the top row, we write the visited base state features and observations, in the middle row, we graphically represent those, and in the bottom row, we present the corresponding decision tree policy traversal. \mathcal{A} tracks the 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.

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

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

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

The initial observation is always the grid bounds $o_0 = (0, 2, 0, 2)$ 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 3.4). 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 3.4).

In figure 13 we illustrate a trajectory in this IBMDP.

1100 B. Reproducing “Iterative Bounding MDPs: Learning Interpretable Policies via 1101 Non-Interpretable Methods”

1102 We attempt to reproduce the results from (Topin et al., 2021) in which authors compare direct and indirect learning of
1103 decision tree policies of depth at most 2 for the CartPole MDP (Barto et al., 1983). In the original paper, the authors find
1104 that both direct and indirect learning yields decision tree policies with similar RL objective values (cf. definition 3.2) for the
1105 CartPole. On the other hand, we find that, imitation learning, despite not directly optimizing the RL objective for CartPole,
1106 outperforms deep RL that optimizes the interpretable RL objective (cf. definition ?? in which the objective trades off the
1107 standard RL objective and interpretability).

1108 Authors of (Topin et al., 2021) use two deep reinforcement learning baselines (cf. section ??) to which they apply some
1109 modifications in order to learn partially observable policies as required by proposition ?? and by the interpretable RL
1110 objective (cf. definition ??). Authors modify the standard DQN (cf. algorithm ??) to return a partially observable policy.
1111 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
1112 this modified DQN, the temporal difference error target for the Q -function $O \rightarrow A \cup A_{info}$ is approximated by a neural
1113 network $S \times O \rightarrow A \cup A_{info}$ that is in turn trained by bootstrapping the temporal difference error with itself. We present
1114 the modifications in algorithm 4. Similar modifications are applied to the standard PPO (cf. algorithm ??) that we present
1115 in the appendix (cf. algorithm ??). In the modified PPO, neural network policy $O \rightarrow A \cup A_{info}$ is trained using a neural
1116 network value function $S \times O \rightarrow A \cup A_{info}$ as a critic.

1117 Those two variants of DQN and PPO have first been introduced in (Pinto et al., 2017) for robotic tasks with partially
1118 observable components, under the name “asymmetric” actor-critic. Asymmetric RL algorithms that have policy and value
1119 estimates using different information from a POMDP (Sondik, 1978; Sigaud & Buffet, 2013) were later studied theoretically
1120 to solve POMDPs in Baisero’s work (Baisero et al., 2022; Baisero & Amato, 2022). The connections from Deep RL in
1121 IBMDPs for objective is absent from (Topin et al., 2021) and we defer their connections to direct interpretable reinforcement
1122 learning to the next chapter as our primary goal is to reproduce (Topin et al., 2021) *as is*. Next, we present the precise
1123 experimental setup we use to reproduce (Topin et al., 2021) in order to study direct deep reinforcement learning of decision
1124 tree policies for the CartPole MDP.

1125 B.1. Experimental setup

1126 B.1.1. (IB)MDP

1127 We use the exact same base MDP and associated IBMDPs for our experiments as (Topin et al., 2021) except when mentioned
1128 otherwise.

1129 **Base MDP** The task at hand is to optimize the RL objective (cf. definition 3.2) with a decision tree policy for the CartPole
1130 MDP (Barto et al., 1983). At each time step a learning algorithm observes the cart’s position and velocity and the pole’s
1131 angle and angular velocity, and can take action to push the CartPole left or right. While the CartPole is roughly balanced,
1132 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,
1133 the MDP transitions to an absorbing terminal state and gets 0 reward forever. Like in (Topin et al., 2021), we use the gymnasium
1134 CartPole-v0 implementation (Towers et al., 2024) of the CartPole MDP in which trajectories are truncated after 200
1135 timesteps making the maximum cumulative reward, i.e. the optimal value of the RL objective when $\gamma = 1$, to be 200. The
1136 state features of the CartPole MDP are in $[-2, 2] \times [-2, 2] \times [-0.14, 0.14] \times [-1.4, 1.4]$.

1137 **IBMDP** Authors define the associated IBMDP (cf. definition 3.4) with $\zeta = -0.01$ and 4 information gathering actions. In
1138 appendix ??, we give more details about how the authors of the original IBMDP paper chose the information gathering
1139 actions. In addition to the original IBMDP paper, we also try $\zeta = 0.01$ and 3 information gathering actions. We use the
1140 same discount factor as the authors: $\gamma = 1$. We try two different approaches to limit the depth of decision tree policies to be
1141 at most 2: terminating trajectories if the agent takes too many information gathering actions in a row or simply giving a
1142 reward of -1 to the agent every time it takes an information gathering action past the depth limit. In practice, we could have
1143 tried an action masking approach, i.e. having a state dependent-action set, but we want to abide to the MDP formalism in
1144 order to properly understand direct interpretable approaches. We will also try IBMDPs where we do not limit the maximum
1145 depth for completeness.

1155
 1156
 1157
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167 **Algorithm 4** Modified Deep Q-Network. We highlight in green the changes to the standard DQN (cf. algorithm ??).
 1168 **Data:** IBMDP $\mathcal{M}_{IB}(S \times O, A \cup A_{info}, (R, \zeta), (T, T_0, T_{info}))$, learning rate α , exploration rate ϵ , partially observable
 1169 Q-network parameters θ , Q-network parameters ϕ , replay buffer \mathcal{B} , update frequency C
 1170 **Result:** Partially observable deterministic policy π_{po}
 1171 Initialize partially observable Q-network parameters θ
 1172 Initialize Q-network parameters ϕ and target network parameters $\phi^- = \phi$
 1173
 1174 Initialize replay buffer $\mathcal{B} = \emptyset$
 1175 **for each episode do**
 1176 Initialize base state features $s_0 \sim T_0$
 1177 Initialize observation $\mathbf{o}_0 = (L_1, U_1, \dots, L_p, U_p)$
 1178
 1179 **for each step t do**
 1180 Choose action a_t using ϵ -greedy: $a_t = \text{argmax}_a Q_\theta(\mathbf{o}_t, a)$ with prob. $1 - \epsilon$
 1181 Take action a_t , observe r_t
 1182 Store transition $(s_t, \mathbf{o}_t, a_t, r_t, s_{t+1})$ in \mathcal{B}
 1183 Sample random batch $(\mathbf{s}_i, \mathbf{o}_i, a_i, r_i, s_{i+1}) \sim \mathcal{B}$
 1184 $a' = \underset{a}{\text{argmax}} Q_\theta(\mathbf{o}_i, a)$
 1185 $y_i = r_i + \gamma Q_{\phi^-}(\mathbf{s}_{i+1}, a')$ // Compute target
 1186 $\phi \leftarrow \phi - \alpha \nabla_\phi (Q_\phi(\mathbf{s}_i, a_i) - y_i)^2$ // Update Q-network
 1187 $\theta \leftarrow \theta - \alpha \nabla_\theta (Q_\theta(\mathbf{o}_i, a_i) - y_i)^2$ // Update partially observable Q-network
 1188 **if** $t \bmod C = 0$ **then**
 1189 $\theta^- \leftarrow \theta$ // Update target network
 1190 **end**
 1191 $s_t \leftarrow s_{t+1}$
 1192 $\mathbf{o}_t \leftarrow \mathbf{o}_{t+1}$
 1193
 1194 **end**
 1195 **end**
 1196 $\pi_{po}(\mathbf{o}) = \text{argmax}_a Q_\theta(\mathbf{o}, a)$ // Extract greedy policy

1197
 1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209

1210 B.1.2. BASELINES
 1211

1212 **Modified DQN and Modified PPO** as mentioned above, the authors use the modified version of DQN from algorithm 4.
 1213 We use the exact same hyperparameters for modified DQN as the authors when possible. We use the same layers width
 1214 (128) and number of hidden layers (2), the same exploration strategy (ϵ -greedy with linearly decreasing value ϵ between
 1215 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
 1216 to be collected randomly before doing value updates (10^5). We also try to use more exploration during training (change
 1217 the initial ϵ value to 0.9). We use the same optimizer (RMSprop with hyperparameter 0.95 and learning rate 2.5×10^{-4})
 1218 to update the Q -networks. Authors did not share which DQN implementation they used so we use the stable-baselines3
 1219 one (Raffin et al., 2021). Authors did not share which activation functions they used so we try both tanh and relu. For the
 1220 modified PPO algorithm (cf. algorithm ??), we can exactly match the authors hyperparameters since they use the open
 1221 source stable-baselines3 implementation of PPO. We match training budgets: we train modified DQN on 1 million timesteps
 1222 and modified PPO on 4 million timesteps.
 1223

1224 **DQN and PPO** We also benchmark the standard DQN and PPO when learning standard Markovian IBMDP policies
 1225 $\pi : S \times O \rightarrow A \cup A_{info}$ and when learning standard $\pi : S \rightarrow A$ policies directly in the CartPole MDP. We summarize
 1226 hyperparameters for the IBMDP and for the learning algorithms in appendices ??, ?? and ??.
 1227

1228 **Indirect methods** We also compare modified RL algorithm to imitation learning (cf. section A.4). To do so, we use VIPER
 1229 or Dagger (cf. algorithms ?? and ??) to imitate greedy neural network policies obtained with standard DQN learning directly
 1230 on CartPole. We use Dagger to imitate neural network policies obtained with the standard PPO learning directly on CartPole.
 1231 For each indirect method, we imitate the neural network experts by fitting decision trees on 10000 expert transitions using
 1232 the CART (cf. algorithm ??) implementation from scikit-learn (Pedregosa et al., 2011) with default hyperparameters and
 1233 maximum depth of 2 like in (Topin et al., 2021).
 1234

 1235 B.1.3. METRICS
 1236

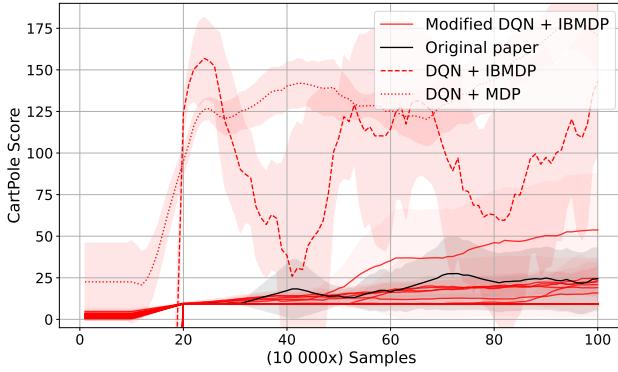
1237 The key metric of this section is performance when controlling the CartPole, i.e, the average *undiscounted* cumulative reward
 1238 of a policy on 100 trajectories (RL objective with $\gamma = 1$). For modified RL algorithms that learn a partially observable
 1239 policy (or Q -function) in an IBMDP, we periodically extract the policy (or Q -function) and use algorithm 1 to extract a
 1240 decision tree for the CartPole MDP. We then evaluate the tree on 100 independent trajectories in the MDP and report the
 1241 mean undiscounted cumulative reward. For RL applied to IBMDPs, since we can't deploy learned policies directly to
 1242 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
 1243 periodically evaluate those IBMDP policies in a copy of the IBMDP in which we fix $\zeta = 0$ ensuring that the copied IBMDP
 1244 undiscounted cumulative rewards only account rewards from the CartPole MDP (non-zero rewards in the IBMDP only occur
 1245 when a reward from the base MDP is given, i.e. when $a_t \in A$ in the IBMDP (cf. definition 3.4)). Similarly, we do 100
 1246 trajectories of the extracted policies in the copied IBMDP and report the average undiscounted cumulative reward. For RL
 1247 applied directly to the base MDP we can just periodically extract the learned policies and evaluate them on 100 CartPole
 1248 trajectories.
 1249

1250 Since imitation learning baselines train offline, i.e, on a fixed dataset, their performances cannot directly be reported on
 1251 the same axis as RL baselines. For that reason, during the training of a standard RL baseline, we periodically extract the
 1252 trained neural policy/ Q -function that we consider as the expert to imitate. Those experts are then imitated with VIPER or
 1253 Dagger using 10 000 newly generated transitions and then fitted decision tree policies are then evaluated on 100 CartPole
 1254 trajectories. We do not report the imitation learning objective values during VIPER or Dagger training. Every single
 1255 combination of IBMDP and Modified RL hyperparameters is run 20 times. For standard RL on either an IBMDP or an MDP,
 1256 we use the paper original hyperparameters when they were specified, with depth control using negative rewards, tanh()
 1257 activations. We use 20 individual random seeds for every experiment in this chapter. Next, we present our results when
 1258 reproducing (Topin et al., 2021).
 1259

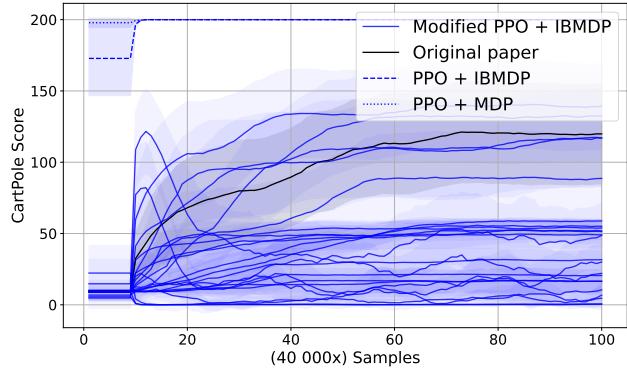
 1260 B.2. Results
 1261

 1262 B.2.1. HOW WELL DO MODIFIED DEEP RL BASELINES LEARN IN IBMDPs?
 1263

1264 On figure 14a, 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
 1265



(a) DQN variants



(b) PPO variants

Figure 13. Comparison of modified reinforcement learning algorithms on different CartPole IBMDPs. (a) Shows variations of modified DQN and DQN (cf. table ??), while (b) shows variations of modified PPO and PPO (cf. table ??). For both algorithms, we give different line-styles for the learning curves when applied directly on the CartPole MDP versus when applied on the IBMDP to learn standard Markovian policies. We color the modified RL algorithm variant from the original paper in black. Shaded areas represent the confidence interval at 95% at each measure on the y-axis.

x-axis and have low y-values. In particular, the highest final y-value, among all the learning curves that could possibly correspond to the original paper modified DQN, correspond to poor performances on the CartPole MDP. On figure 14b, 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 14a and 14b, we observe that standard RL baselines (RL + IBMDP and RL + MDP), learn better CartPole policies in average than their modified counterparts that learn partially observable policies (cf. proposition ??). On figure 14b, it is clear that for the standard PPO baselines, learning is super efficient and algorithms learn optimal policies with reward 200 in few thousands steps.

B.2.2. WHICH DECISION TREE POLICIES DOES DIRECT REINFORCEMENT LEARNING RETURN FOR THE CARTPOLE MDP?

On figure 15, we isolate the best performing algorithms instantiations that learn decision tree policies for the CartPole MDP. We compare the best modified DQN and modified PPO to imitation learning baselines that use the surrogate imitation objective (cf. definition A.2) to find CartPole decision tree policies. We find that despite having poor performances in *average*, the modified deep reinforcement learning baselines can find very good decision tree policies as shown by the min-max shaded areas on the left of figure 15 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 16, we present the best decision tree policies for CartPole returned by modified DQN and modified PPO. We used algorithm 1 to extract 20 trees from the 20 partially observable policies returned by the modified deep reinforcement learning algorithms over the 20 training seeds. We then plot the best tree for each baseline. Those trees get an average RL objective of roughly 175. Similarly, we plot a representative tree for imitation learning baseline as well as a tree that is optimal for CartPole w.r.t. the RL objective obtained with VIPER. Unlike for direct methods, the trees returned by imitation learning are extremely similar across seeds. In particular they often only vary in the scalar value used in the root node but in general have the same structure and test the angular velocity. On the other hand the most frequent trees across seeds returned by modified RL baselines are “trivial” decision tree policies that either repeat the same base action forever or repeat the same IGA (cf. definition 3.4) forever.

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

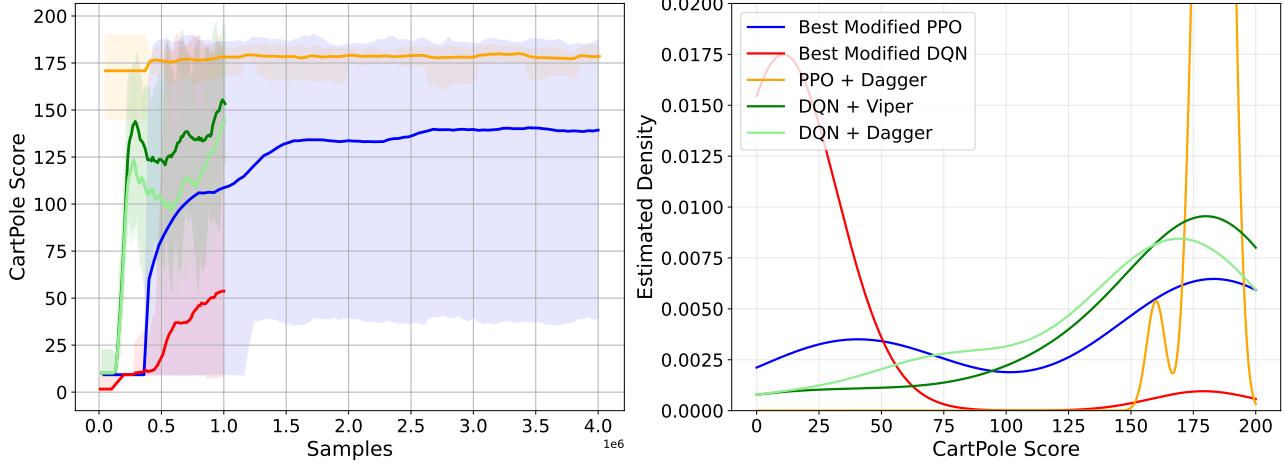


Figure 14. (left) Mean performance of the best-w.r.t. the RL objective for CartPole-modified RL + IBMDP combination. Shaded areas represent the min and max performance over the 20 seeds during training. (right) Corresponding score distribution of the final decision tree policies w.r.t. the RL objective for CartPole.

figures 14a and 14b). 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 15). In the next chapter, we highlight the connections between direct interpretable RL (cf. definition ??) and POMDPs to get insights on the hardness of direct reinforcement learning of decision trees.

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

C. RL objective values

Optimal depth-1 decision tree policy $\pi_{\mathcal{T}_1}$ has one root node that tests $x \leq 1$ (respectively $y \leq 1$) and two leaf nodes \rightarrow and \downarrow . To compute $V_{\mathcal{T}_1}^{\pi}(o_0)$, we compute the values of $\pi_{\mathcal{T}_1}$ in each of the possible starting states $(s_0, o_0), (s_1, o_0), (s_2, o_0), (s_g, o_0)$ and compute the expectation over those. At initialization, when the base state is $s_g = (1.5, 0.5)$, the depth-1 decision tree policy cycles between taking an information gathering action $x \leq 1$ and moving down to get a positive reward for which it gets the returns:

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(s_g, o_0) &= \zeta + \gamma + \gamma^2 \zeta + \gamma^3 \dots \\ &= \sum_{t=0}^{\infty} \gamma^{2t} \zeta + \sum_{t=0}^{\infty} \gamma^{2t+1} \\ &= \frac{\zeta + \gamma}{1 - \gamma^2} \end{aligned}$$

At initialization, in either of the base states $s_0 = (0.5, 0.5)$ and $s_2 = (1.5, 1.5)$, the value of the depth-1 decision tree policy is the return when taking one information gathering action $x \leq 1$, then moving right or down, then following the policy from the goal state s_g :

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(s_0, o_0) &= \zeta + \gamma 0 + \gamma^2 V^{\pi_{\mathcal{T}_1}}(s_g, o_0) \\ &= \zeta + \gamma^2 V^{\pi_{\mathcal{T}_1}}(s_g, o_0) \\ &= V^{\pi_{\mathcal{T}_1}}(s_2, o_0) \end{aligned}$$



Similarly, the value of the best depth-1 decision tree policy in state $s_1 = (0.5, 1.5)$ is the value of taking one information gathering action then moving right to s_2 then following the policy in s_2 :

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(s_1, o_0) &= \zeta + \gamma^2 V^{\pi_{\mathcal{T}_1}}(s_2, o_0) \\ &= \zeta + \gamma^2 V^{\pi_{\mathcal{T}_1}}(s_2, o_0) \\ &= \zeta + \gamma^2(\zeta + \gamma^2 V^{\pi_{\mathcal{T}_1}}(s_g, o_0)) \\ &= \zeta + \gamma^2 \zeta + \gamma^4 V^{\pi_{\mathcal{T}_1}}(s_g, o_0) \end{aligned}$$

Since the probability of being in any base states at initialization given that the observation is o_0 is simply the probability of being in any base states at initialization, we can write:

$$\begin{aligned} V^{\pi_{\mathcal{T}_1}}(o_0) &= \frac{1}{4} V^{\pi_{\mathcal{T}_1}}(s_g, o_0) + \frac{2}{4} V^{\pi_{\mathcal{T}_1}}(s_2, o_0) + \frac{1}{4} V^{\pi_{\mathcal{T}_1}}(s_1, o_0) \\ &= \frac{1}{4} \frac{\zeta + \gamma}{1 - \gamma^2} + \frac{2}{4} (\zeta + \gamma^2 \frac{\zeta + \gamma}{1 - \gamma^2}) + \frac{1}{4} (\zeta + \gamma^2 \zeta + \gamma^4 \frac{\zeta + \gamma}{1 - \gamma^2}) \\ &= \frac{1}{4} \frac{\zeta + \gamma}{1 - \gamma^2} + \frac{2}{4} \left(\frac{\zeta + \gamma^3}{1 - \gamma^2} \right) + \frac{1}{4} \left(\frac{\zeta + \gamma^5}{1 - \gamma^2} \right) \\ &= \frac{4\zeta + \gamma + 2\gamma^3 + \gamma^5}{4(1 - \gamma^2)} \end{aligned}$$

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

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

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

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

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

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

Unbalanced depth-2 decision tree: the unbalanced depth-2 decision tree takes an information gathering action $x \leq 0.5$ then either takes the \downarrow action or takes a second information $y \leq 0.5$ followed by \rightarrow or \downarrow . In states G and S_2 , the value of the unbalanced tree is the same as for the depth-1 tree. In states S_0 and S_1 , the policy takes two information gathering actions before taking a base action and so on:

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

$$\begin{aligned}
 V_{S_1}^{\mathcal{T}_u} &= \zeta + \gamma\zeta + \gamma^20 + \gamma^3V_{S_0}^{\mathcal{T}_u} \\
 &= \zeta + \gamma\zeta + \gamma^20 + \gamma^3(\zeta + \gamma\zeta + \gamma^20 + \gamma^3V_G^{\mathcal{T}_1}) \\
 &= \zeta + \gamma\zeta + \gamma^3\zeta + \gamma^4\zeta + \gamma^6V_G^{\mathcal{T}_1}
 \end{aligned}$$

We get:

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

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

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

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

$$\begin{aligned}
 J(\mathcal{T}_2) &= \frac{1}{4}V_G^{\mathcal{T}_2} + \frac{2}{4}V_{S_2}^{\mathcal{T}_2} + \frac{1}{4}V_{S_1}^{\mathcal{T}_2} \\
 &= \frac{1}{4}V_G^{\mathcal{T}_2} + \frac{2}{4}(\zeta + \gamma\zeta + \gamma^20 + \gamma^3V_G^{\mathcal{T}_2}) + \frac{1}{4}(\zeta + \gamma\zeta + \gamma^20 + \gamma^3\zeta + \gamma^4\zeta + \gamma^50 + \gamma^6V_G^{\mathcal{T}_2}) \\
 &= \frac{\zeta(3 + 3\gamma) + \gamma^2 + \gamma^5 + \gamma^8}{4(1 - \gamma^3)}
 \end{aligned}$$

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

Stochastic policy: the other non-trivial policy that can be learned by solving a partially observable IBMDP is the stochastic policy that guarantees to reach G after some time: fifty percent chance to do \rightarrow and fifty percent chance to do \downarrow . This stochastic policy has objective value:

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

1540 Solving these equations:

1541
1542
1543
1544
1545
1546

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

$$V_{S_1}^{\text{stoch}} - \frac{1}{4}\gamma^2 V_{S_1}^{\text{stoch}} = \frac{1}{4}\gamma^2 V_G^{\text{stoch}} + \frac{1}{2}\gamma V_G^{\text{stoch}}$$

$$V_{S_1}^{\text{stoch}}(1 - \frac{1}{4}\gamma^2) = (\frac{1}{4}\gamma^2 + \frac{1}{2}\gamma)V_G^{\text{stoch}}$$

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

1565

1566

1567

1568

1569

1570

1571

1572

1573

1574

1575

1576

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

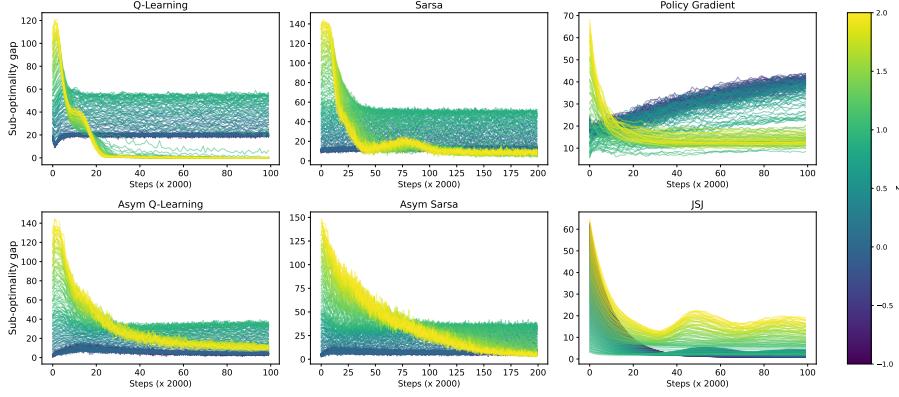


Figure 16. (Asymmetric) reinforcement learning in POIBMDPs. In each subplot, each single line is colored by the value of ζ in the corresponding POIBMDP in which learning occurs. Each single learning curve represent the sub-optimality gap averaged over 100 seeds.

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

D. Training curves