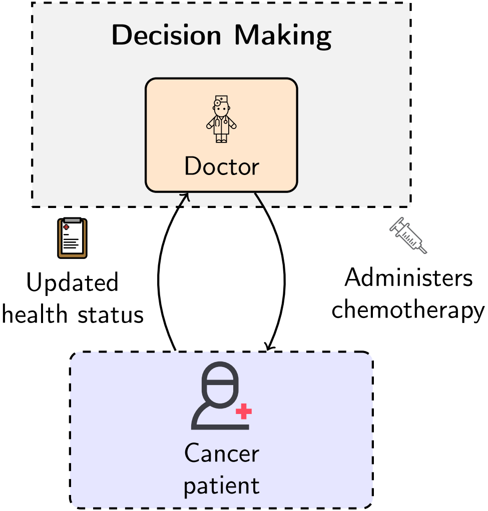
Interpretability, Decision Trees, and Sequential Decision Making

Hector Kohler

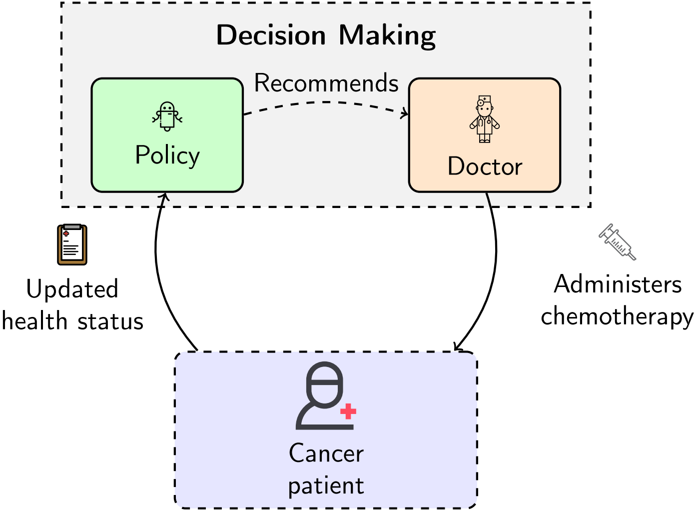
Supervised by Dr. Riad Akrour (HdR) and Prof. Philippe Preux (HdR)
Université de Lille, CNRS, Inria, UMR CRIStAL 9189, France

December 4, 2025

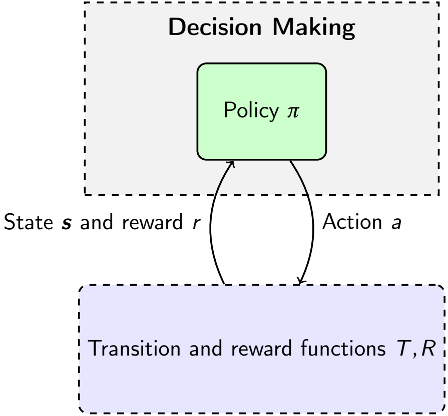
Sequential decision making (SDM)



Sequential decision making (SDM)

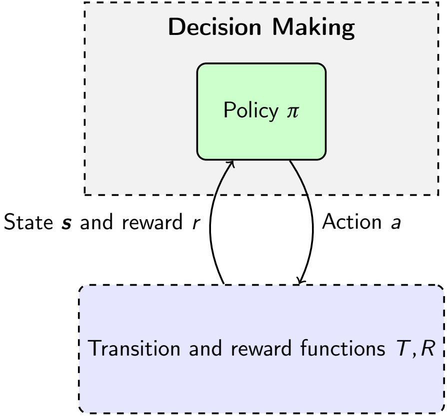


Markov decision processes (MDPs) and reinforcement learning (RL)



Markov decision processes (Puterman 1994).

Markov decision processes (MDPs) and reinforcement learning (RL)

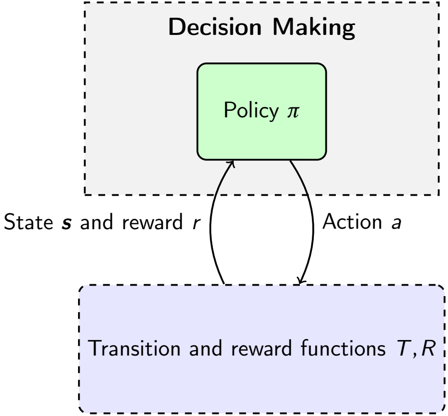


Markov decision processes (Puterman 1994).

- RL (Sutton and Barto 1998) aims to find a policy, $\pi : S \to A$ that maximizes:

$$J(\pi) = \mathop{\mathbb{E}}_{\boldsymbol{s}_t \sim T}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right]$$

Markov decision processes (MDPs) and reinforcement learning (RL)



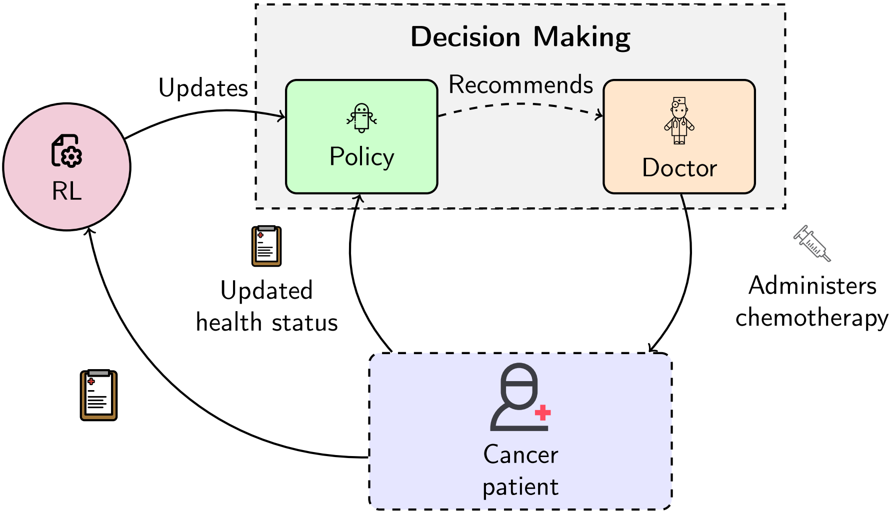
Markov decision processes (Puterman 1994).

- RL (Sutton and Barto 1998) aims to find a policy, $\pi : S \rightarrow A$ that maximizes:

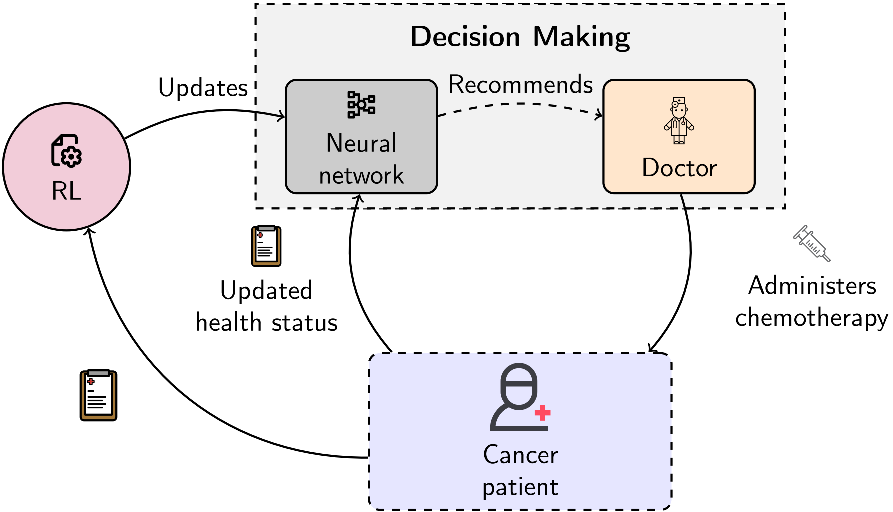
$$J(\pi) = \mathop{\mathbb{E}}_{\boldsymbol{s}_t \sim T} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \right]$$

- Lots of succesful (deep) RL algorithms (Schulman et al. 2017).

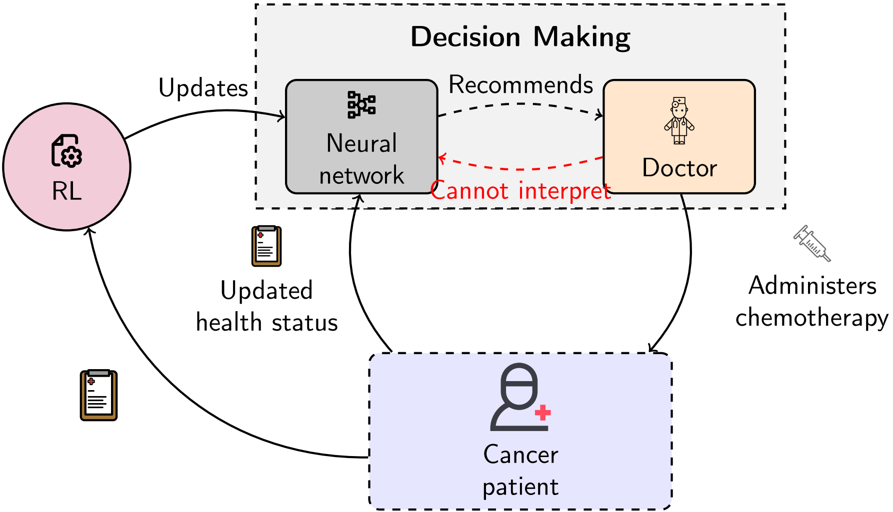
Sequential decision making and reinforcement learning



Sequential decision making and reinforcement learning

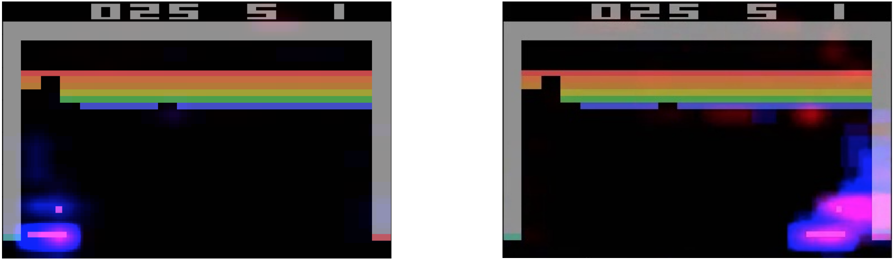


Sequential decision making and reinforcement learning



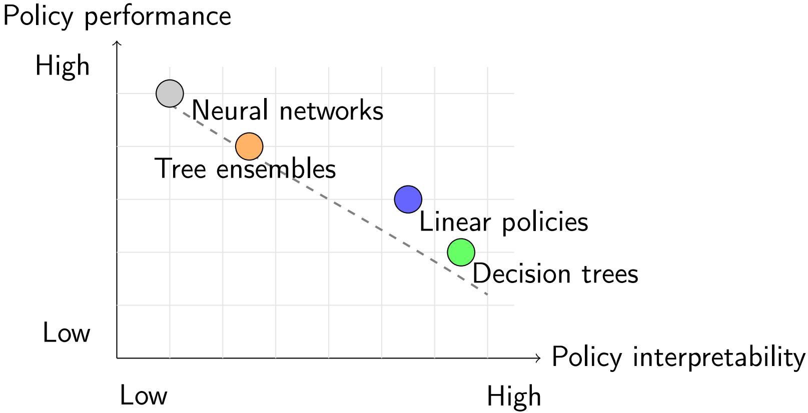
Interpretability in SDM (Glanois et al. 2024).

⚠**No concensus on definition (Lipton 2018)**.



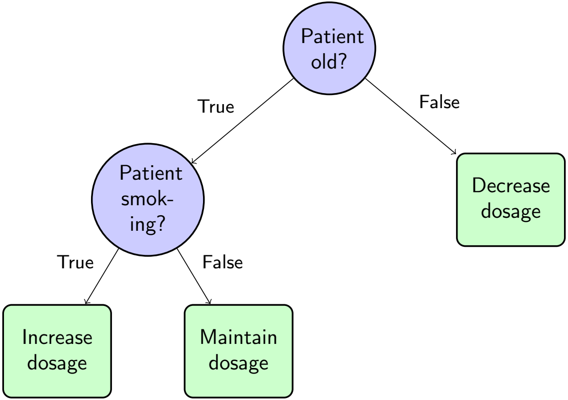
Local interpretability with saliency maps of different MDP states (Greydanus et al. 2018).

Interpretability in SDM (Glanois et al. 2024).



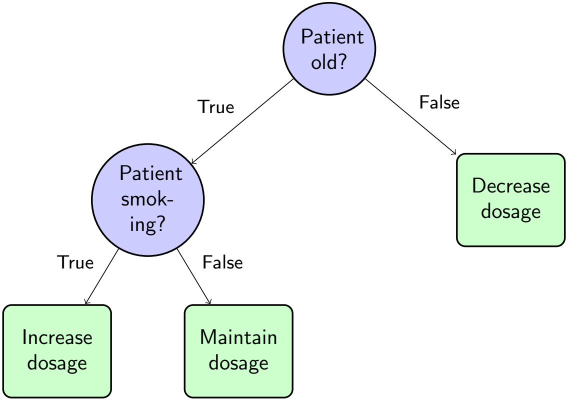
Global interpretability with intrinsically interpretable policies.

Decision trees



A generic decision tree of depth $D = 2$.

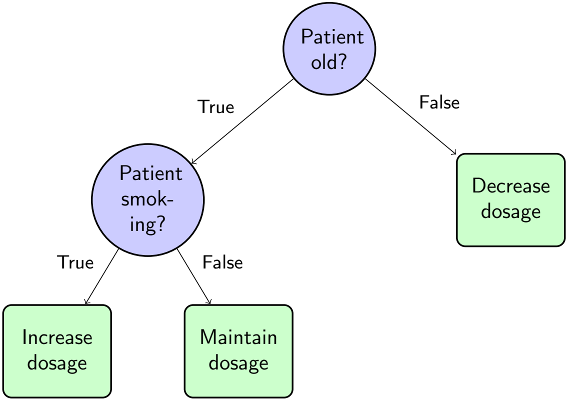
Decision trees



A generic decision tree of depth $D = 2$.

Successful algorithms for classification/regression (Breiman et al. 1984).

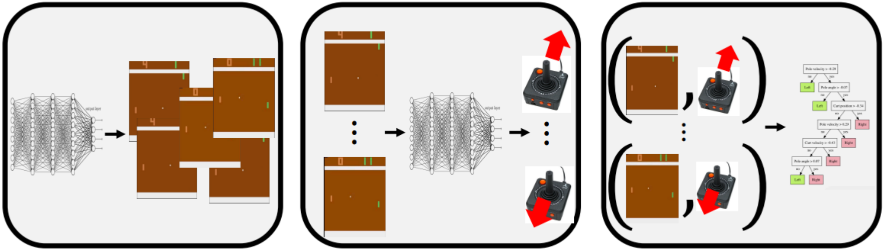
Decision trees



A generic decision tree of depth $D = 2$.

Successful algorithms for classification/regression (Breiman et al. 1984).
What about SDM?

Imitation learning

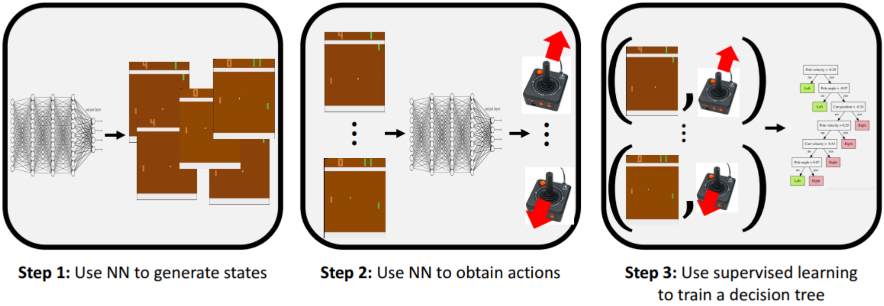


Step 1: Use NN to generate states

Step 2: Use NN to obtain actions

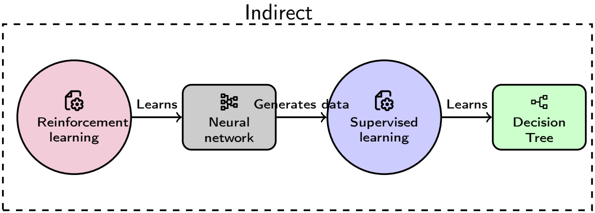
Step 3: Use supervised learning to train a decision tree

Imitation learning

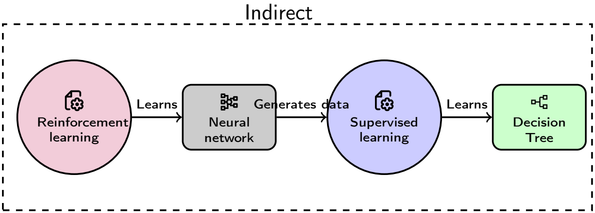


Most research focused on indirect learning of interpretable policies (Bastani, Pu, and Solar-Lezama 2018).

Two ways to get interpretable policies for SDM (Glanois et al. 2024)



Two ways to get interpretable policies for SDM (Glanois et al. 2024)

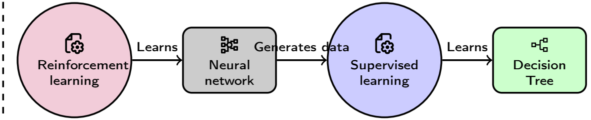


⚠**Policies obtained indirectly optimize a surrogate objective rather than an MDP cumulative rewards.**

Two ways to get interpretable policies for SDM (Glanois et al. 2024)

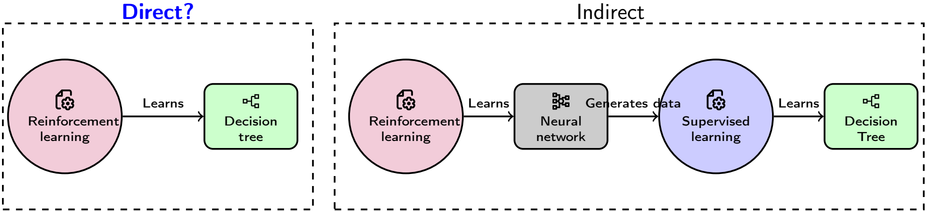
Direct?

Indirect



⚠**Policies obtained indirectly optimize a surrogate objective rather than an MDP cumulative rewards.**

Two ways to get interpretable policies for SDM (Glanois et al. 2024)



⚠**Policies obtained indirectly optimize a surrogate objective rather than an MDP cumulative rewards.**

Contributions

1. Can we directly train decision tree policies that trade off interpretability and performances for SDM?
2. Can we leverage SDM to learn decision trees for classification/regression?
3. How to measure policy interpretability in SDM?

Contributions

1. Can we directly train decision tree policies that trade off interpretability and performances for SDM?
2. Can we leverage SDM to learn decision trees for classification/regression?
3. How to measure policy interpretability in SDM?

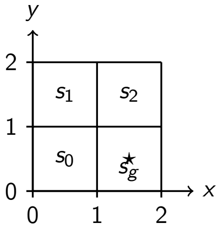
Contributions

1. Can we directly train decision tree policies that trade off interpretability and performances for SDM?
2. Can we leverage SDM to learn decision trees for classification/regression?
3. How to measure policy interpretability in SDM?

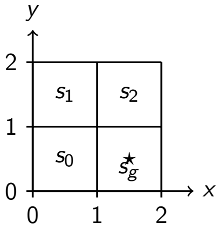
Contributions

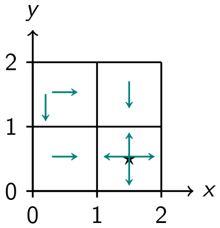
1. Can we directly train decision tree policies that trade off interpretability and performances for SDM?
2. Can we leverage SDM to learn decision trees for classification/regression?
3. How to measure policy interpretability in SDM?

Grid world MDP and decision tree policies



Grid world MDP and decision tree policies



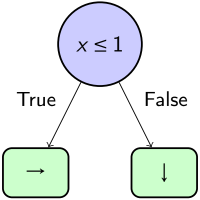


Grid world MDP and decision tree policies

Grid world MDP and optimal actions.

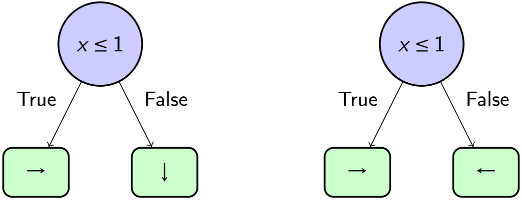
Grid world MDP and decision tree policies

Grid world MDP and optimal actions.



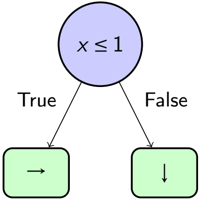
Grid world MDP and decision tree policies

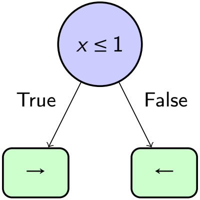
Grid world MDP and optimal actions.

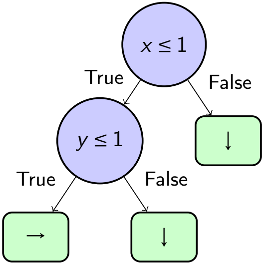


Grid world MDP and decision tree policies

Grid world MDP and optimal actions.

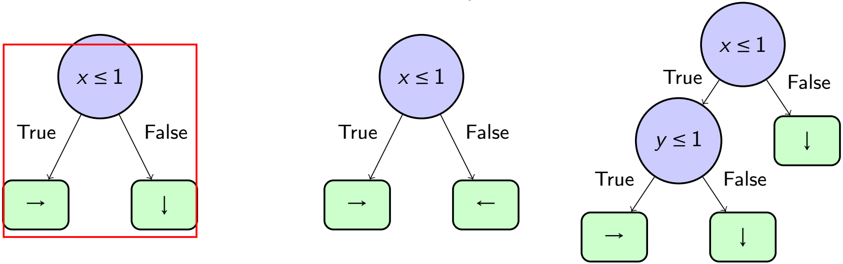






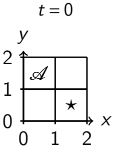
Grid world MDP and decision tree policies

Grid world MDP and optimal actions.



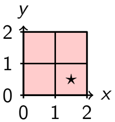
Decision tree policies with different interpretability-performance trade-offs.

Iterative bounding Markov decision processes (Topin et al. 2021)

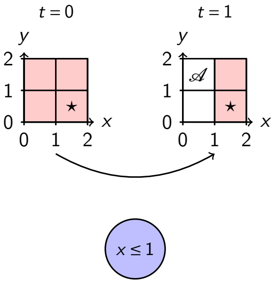


Iterative bounding Markov decision processes (Topin et al. 2021)

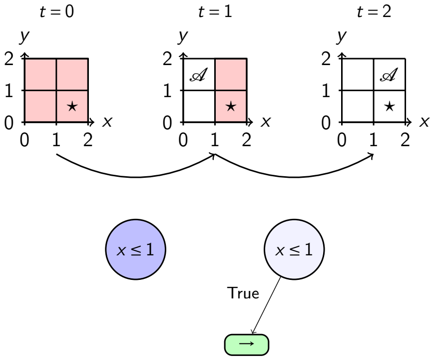
$t = 0$



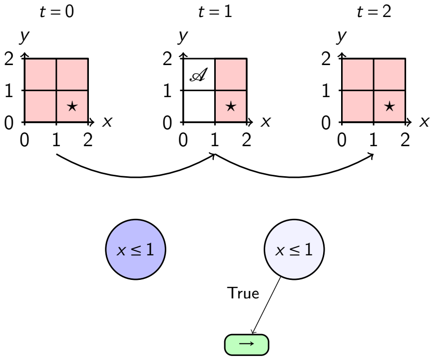
Iterative bounding Markov decision processes (Topin et al. 2021)



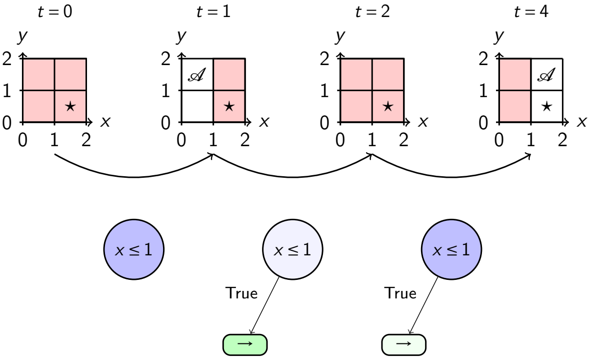
Iterative bounding Markov decision processes (Topin et al. 2021)



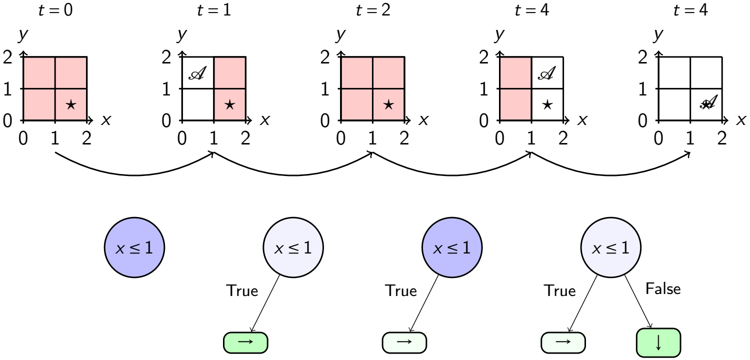
Iterative bounding Markov decision processes (Topin et al. 2021)



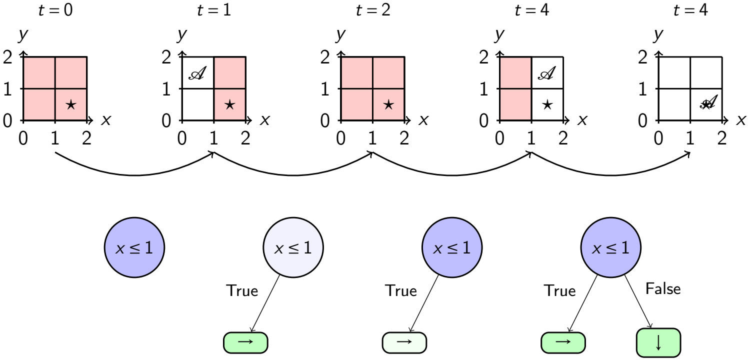
Iterative bounding Markov decision processes (Topin et al. 2021)



Iterative bounding Markov decision processes (Topin et al. 2021)



Iterative bounding Markov decision processes (Topin et al. 2021)



Iterative bounding Markov decision processes (Topin et al. 2021)

Given an MDP $\mathscr{M}$ $\langle S, A, R, T \rangle$

Iterative bounding Markov decision processes (Topin et al. 2021)

Given an MDP $\mathscr{M}$ $\langle S, A, R, T\rangle$, an IBMDP is an MDP

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

where:

Iterative bounding Markov decision processes (Topin et al. 2021)

Given an MDP $\mathscr{M}\ \langle S, A, R, T \rangle$, an IBMDP is an MDP

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

where:

- O: observations of some state features.

Iterative bounding Markov decision processes (Topin et al. 2021)

Given an MDP $\mathscr{M}$ $\langle S, A, R, T\rangle$, an IBMDP is an MDP

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

where:

- O: observations of some state features.
- A_{info}: actions that gather informations about some state features.

Iterative bounding Markov decision processes (Topin et al. 2021)

Given an MDP $\mathscr{M}\ \langle S, A, R, T\rangle$, an IBMDP is an MDP

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

where:

- O: observations of some state features.
- A_{info}: actions that gather informations about some state features.
- ζ: reward for taking $a \in A_{info}$.

Iterative bounding Markov decision processes (Topin et al. 2021)

Given an MDP $\mathscr{M}\ \langle S, A, R, T \rangle$, an IBMDP is an MDP

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

where:

- O: observations of some state features.
- A_{info}: actions that gather informations about some state features.
- ζ: reward for taking $a \in A_{info}$.
- T_{info}: transitions following $a \in A_{info}$ that update the partial observation with gathered info.

Iterative bounding Markov decision processes (Topin et al. 2021)

Given an MDP $\mathscr{M}$ $\langle S, A, R, T \rangle$, an IBMDP is an MDP

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

where:

- O: observations of some state features.
- A_{info}: actions that gather informations about some state features.
- ζ: reward for taking $a \in A_{info}$.
- T_{info}: transitions following $a \in A_{info}$ that update the partial observation with gathered info.
- ⚠**IBMDP policies $\pi_{po} : O \rightarrow A \cup A_{info}$ are decision tree policies for $\mathscr{M}$.**

RL for memoryless policies in POMDPs

RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

RL for memoryless policies in POMDPs

RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

RL for memoryless policies in POMDPs

RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

RL for memoryless policies in POMDPs

RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

RL for memoryless policies in POMDPs

RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

RL for memoryless policies in POMDPs

RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

RL for memoryless policies in POMDPs

RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

RL for memoryless policies in POMDPs

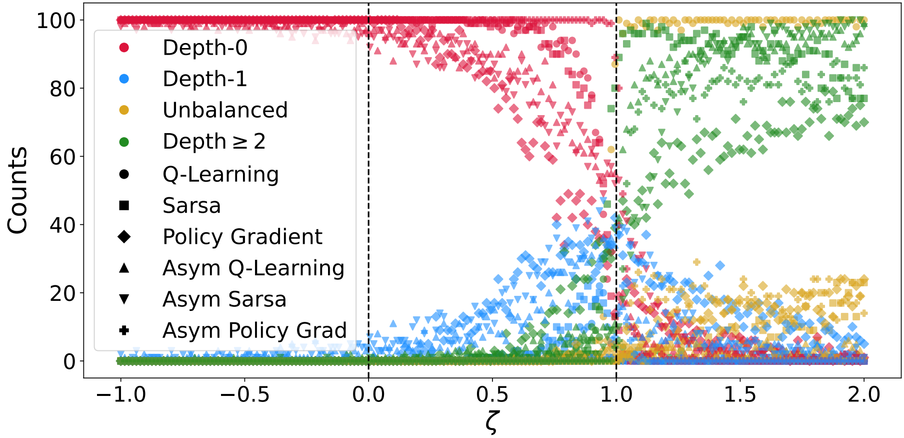
RL for memoryless policies

- Finding the best deterministic and memoryless policy in a POMDP is NP-hard (Littman 1994)!
- The best memoryless policy can be stochastic (Singh, Jaakkola, and Jordan 1994).
- Value-based RL converges to sub-optimal solutions.

Asymmetric RL (Pinto et al. 2017)

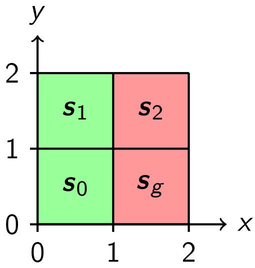
- Value-based → learns $Q(o, a)$ with TD targets $U(s, a)$ (Baisero, Daley, and Amato 2022).
- Actor-critic → policy gradient on $\pi(o, a)$ using a critic $V(s)$ (Baisero and Amato 2022).
- Supposed to work better for our problem (Lambrechts, Ernst, and Mahajan 2025).

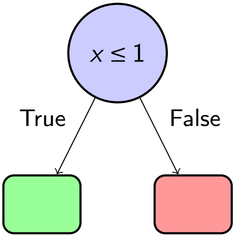
Result: RL cannot retrieve optimal depth-1 trees for the grid world MDP



Distributions of tree policies learned with (asymmetric) RL algorithms as a function of the interpretability reward ζ.

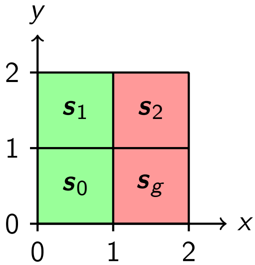
Direct RL of decision trees for classification tasks does not involve partial observability

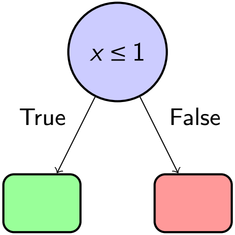




Classification MDP and the unique optimal depth-1 tree.

Direct RL of decision trees for classification tasks does not involve partial observability

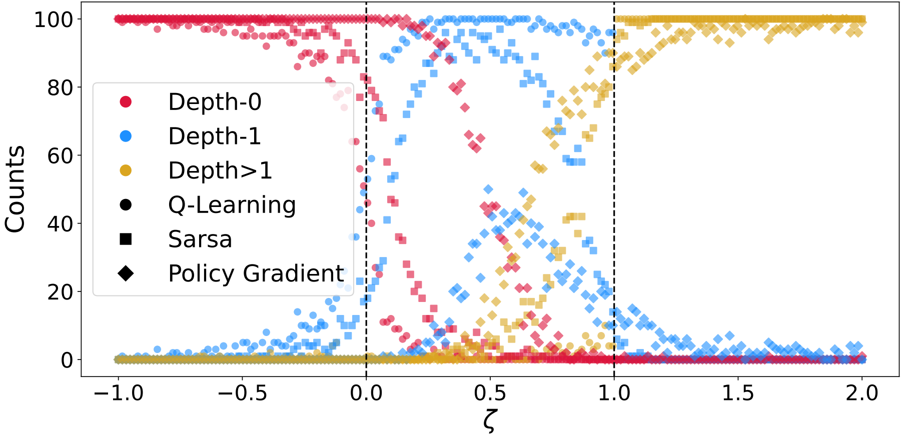




Classification MDP and the unique optimal depth-1 tree.

Partial observations are sufficient statistics of the full states in IBMDPs for classification tasks.

Result: RL can retrieve optimal depth-1 trees for the toy classification MDP



Distributions of tree policies learned with various RL algorithms.

Perspectives for direct RL of decision tree policies.

- It seems that directly learning decision tree trading off interpretability and performances in MDPs can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches (Wu et al. 2020)?
- Fixing the policy tree structure a priori (parametric trees, (Marton et al. 2025))?

RL can train good decision trees for classification MDPs

Q: Can we leverage SDM to design new decision tree induction algorithms for classification/regression?

Perspectives for direct RL of decision tree policies.

- It seems that directly learning decision tree trading off interpretability and performances in MDPs can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches (Wu et al. 2020)?
- Fixing the policy tree structure a priori (parametric trees, (Marton et al. 2025))?

RL can train good decision trees for classification MDPs

Q: Can we leverage SDM to design new decision tree induction algorithms for classification/regression?

Perspectives for direct RL of decision tree policies.

- It seems that directly learning decision tree trading off interpretability and performances in MDPs can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches (Wu et al. 2020)?
- Fixing the policy tree structure a priori (parametric trees, (Marton et al. 2025))?

RL can train good decision trees for classification MDPs

Q: Can we leverage SDM to design new decision tree induction algorithms for classification/regression?

Perspectives for direct RL of decision tree policies.

- It seems that directly learning decision tree trading off interpretability and performances in MDPs can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches (Wu et al. 2020)?
- Fixing the policy tree structure a priori (parametric trees, (Marton et al. 2025))?

RL can train good decision trees for classification MDPs

Q: Can we leverage SDM to design new decision tree induction algorithms for classification/regression?

Perspectives for direct RL of decision tree policies.

- It seems that directly learning decision tree trading off interpretability and performances in MDPs can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches (Wu et al. 2020)?
- Fixing the policy tree structure a priori (parametric trees, (Marton et al. 2025))?

RL can train good decision trees for classification MDPs

Q: Can we leverage SDM to design new decision tree induction algorithms for classification/regression?

Perspectives for direct RL of decision tree policies.

- It seems that directly learning decision tree trading off interpretability and performances in MDPs can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches (Wu et al. 2020)?
- Fixing the policy tree structure a priori (parametric trees, (Marton et al. 2025))?

RL can train good decision trees for classification MDPs

Q: Can we leverage SDM to design new decision tree induction algorithms for classification/regression?

Decision trees in supervised learning

- N data points $\{\boldsymbol{x}_i, y_i\}$. Each $\boldsymbol{x}_i$ is described by p features and has a label $y_i \in \mathscr{Y}$. We want to find a tree of depth at most D, $T \in \mathscr{T}_D$ that minimizes:

$$\mathscr{L}_\alpha(T) = \frac{1}{N}\sum_{i=1}^{N}\ell(y_i, T(x_i)) + \alpha C(T)$$

- Trees **interpretable** and **competitive with neural nets** (Grinsztajn, Oyallon, and Varoquaux 2022).
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O(2^D)}$ operations (Breiman et al. 1984) .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O((2Np)^D)}$ operations (NP-hard) (Bertsimas and Dunn 2017).
- In between optimal and greedy?

Decision trees in supervised learning

- N data points $\{\boldsymbol{x}_i, y_i\}$. Each $\boldsymbol{x}_i$ is described by p features and has a label $y_i \in \mathscr{Y}$. We want to find a tree of depth at most D, $T \in \mathscr{T}_D$ that minimizes:

$$\mathscr{L}_\alpha(T) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, T(x_i)) + \alpha C(T)$$

- Trees **interpretable** and **competitive with neural nets** (Grinsztajn, Oyallon, and Varoquaux 2022).
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O}(2^{\boldsymbol{D}})$ operations (Breiman et al. 1984) .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O}((2\boldsymbol{Np})^{\boldsymbol{D}})$ operations (NP-hard) (Bertsimas and Dunn 2017).
- In between optimal and greedy?

Decision trees in supervised learning

- N data points $\{\boldsymbol{x}_i, y_i\}$. Each $\boldsymbol{x}_i$ is described by p features and has a label $y_i \in \mathscr{Y}$. We want to find a tree of depth at most D, $T \in \mathscr{T}_D$ that minimizes:

$$\mathscr{L}_\alpha(T) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, T(x_i)) + \alpha C(T)$$

- Trees **interpretable** and **competitive with neural nets** (Grinsztajn, Oyallon, and Varoquaux 2022).
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O}(2^{\boldsymbol{D}})$ operations (Breiman et al. 1984) .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O}((2\boldsymbol{Np})^{\boldsymbol{D}})$ operations (NP-hard) (Bertsimas and Dunn 2017).
- In between optimal and greedy?

Decision trees in supervised learning

- N data points $\{\boldsymbol{x}_i, y_i\}$. Each $\boldsymbol{x}_i$ is described by p features and has a label $y_i \in \mathscr{Y}$. We want to find a tree of depth at most D, $T \in \mathscr{T}_D$ that minimizes:

$$\mathscr{L}_\alpha(T) = \frac{1}{N}\sum_{i=1}^{N}\ell(y_i, T(x_i)) + \alpha C(T)$$

- Trees **interpretable** and **competitive with neural nets** (Grinsztajn, Oyallon, and Varoquaux 2022).
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O(2^D)}$ operations (Breiman et al. 1984) .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O((2Np)^D)}$ operations (NP-hard) (Bertsimas and Dunn 2017).
- In between optimal and greedy?

Decision trees in supervised learning

- N data points $\{\boldsymbol{x}_i, y_i\}$. Each $\boldsymbol{x}_i$ is described by p features and has a label $y_i \in \mathcal{Y}$. We want to find a tree of depth at most D, $T \in \mathcal{T}_D$ that minimizes:

$$\mathcal{L}_\alpha(T) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, T(x_i)) + \alpha C(T)$$

- Trees **interpretable** and **competitive with neural nets** (Grinsztajn, Oyallon, and Varoquaux 2022).
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O}(2^{\boldsymbol{D}})$ operations (Breiman et al. 1984) .
- Optimal algorithms, optimal accuracy, but $\boldsymbol{O}((2\boldsymbol{Np})^{\boldsymbol{D}})$ operations (NP-hard) (Bertsimas and Dunn 2017).
- In between optimal and greedy?

Decision trees in supervised learning

- N data points $\{\boldsymbol{x}_i, y_i\}$. Each $\boldsymbol{x}_i$ is described by p features and has a label $y_i \in \mathscr{Y}$. We want to find a tree of depth at most D, $T \in \mathscr{T}_D$ that minimizes:

$$\mathscr{L}_\alpha(T) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, T(x_i)) + \alpha C(T)$$

- Trees **interpretable** and **competitive with neural nets** (Grinsztajn, Oyallon, and Varoquaux 2022).
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O}(2^{\boldsymbol{D}})$ operations (Breiman et al. 1984) .
- Optimal algorithms, optimal accuracy, but $\boldsymbol{O}((2\boldsymbol{Np})^{\boldsymbol{D}})$ operations (NP-hard) (Bertsimas and Dunn 2017).
- In between optimal and greedy?

Decision tree induction as solving MDPs

Intuition

The induction of a decision tree is made of a sequence of decisions: at each node, we must decide whether it is better to split (a subset of) the training data, or to create a leaf node.

- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: interpretability term $-\alpha$ and misclassifications.
- T: node traversals.

Proposition (Objective Equivalence)

Let π be a deterministic policy of the MDP. Then $J_\alpha(\pi) = -\mathscr{L}_\alpha(E(\pi, s_0))$ where E is an algorithm that extracts a decision tree from π (Topin et al. 2021).

Decision tree induction as solving MDPs

Intuition

The induction of a decision tree is made of a sequence of decisions: at each node, we must decide whether it is better to split (a subset of) the training data, or to create a leaf node.

- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: interpretability term $-\alpha$ and misclassifications.
- T: node traversals.

Proposition (Objective Equivalence)

Let π be a deterministic policy of the MDP. Then $J_\alpha(\pi) = -\mathcal{L}_\alpha(E(\pi, s_0))$ where E is an algorithm that extracts a decision tree from π (Topin et al. 2021).

Decision tree induction as solving MDPs

Intuition

The induction of a decision tree is made of a sequence of decisions: at each node, we must decide whether it is better to split (a subset of) the training data, or to create a leaf node.

- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: interpretability term $-\alpha$ and misclassifications.
- T: node traversals.

Proposition (Objective Equivalence)

Let π be a deterministic policy of the MDP. Then $J_\alpha(\pi) = -\mathscr{L}_\alpha(E(\pi, s_0))$ where E is an algorithm that extracts a decision tree from π (Topin et al. 2021).

Decision tree induction as solving MDPs

Intuition

The induction of a decision tree is made of a sequence of decisions: at each node, we must decide whether it is better to split (a subset of) the training data, or to create a leaf node.

- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: interpretability term $-\alpha$ and misclassifications.
- T: node traversals.

Proposition (Objective Equivalence)

Let π be a deterministic policy of the MDP. Then $J_\alpha(\pi) = -\mathscr{L}_\alpha(E(\pi, s_0))$ where E is an algorithm that extracts a decision tree from π (Topin et al. 2021).

Decision tree induction as solving MDPs

Intuition

The induction of a decision tree is made of a sequence of decisions: at each node, we must decide whether it is better to split (a subset of) the training data, or to create a leaf node.

- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: interpretability term $-\alpha$ and misclassifications.
- T: node traversals.

Proposition (Objective Equivalence)

Let π be a deterministic policy of the MDP. Then $J_\alpha(\pi) = -\mathscr{L}_\alpha(E(\pi, s_0))$ where E is an algorithm that extracts a decision tree from π (Topin et al. 2021).

Decision tree induction as solving MDPs

Intuition

The induction of a decision tree is made of a sequence of decisions: at each node, we must decide whether it is better to split (a subset of) the training data, or to create a leaf node.

- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: interpretability term $-\alpha$ and misclassifications.
- T: node traversals.

Proposition (Objective Equivalence)

Let π be a deterministic policy of the MDP. Then $J_\alpha(\pi) = -\mathscr{L}_\alpha(E(\pi, s_0))$ where E is an algorithm that extracts a decision tree from π (Topin et al. 2021).

Decision tree induction as solving MDPs

Intuition

The induction of a decision tree is made of a sequence of decisions: at each node, we must decide whether it is better to split (a subset of) the training data, or to create a leaf node.

- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: interpretability term $-\alpha$ and misclassifications.
- T: node traversals.

Proposition (Objective Equivalence)

Let π be a deterministic policy of the MDP. Then $J_\alpha(\pi) = -\mathscr{L}_\alpha(E(\pi, s_0))$ where E is an algorithm that extracts a decision tree from π (Topin et al. 2021).

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random. . .

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random. . .

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random...

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random...

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- Dynamic Programming Decision Trees (DPDT):
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random...

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random...

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random...

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random...

Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $O(2^D)$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $O((2Np)^D)$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

Top-B greedy splits (Blanc et al. 2023), quantiles, random...

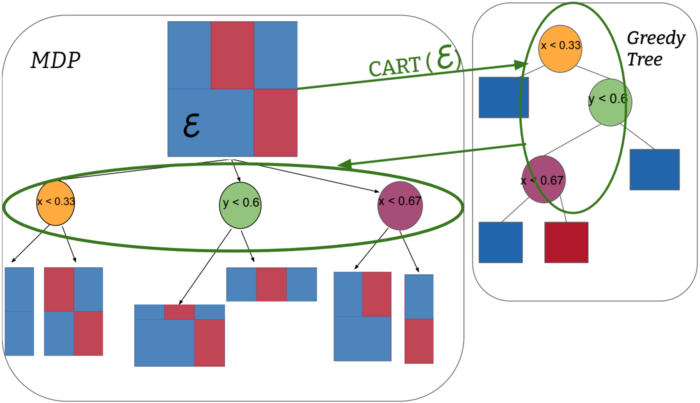
Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion
  → **MDP state space size is $\boldsymbol{O(2^D)}$**.
- Optimal algorithms consider all possible actions in each state
  → **MDP state space size is $\boldsymbol{O((2Np)^D)}$**.
- **Dynamic Programming Decision Trees (DPDT)**:
  → for each MDP state consider B actions: **state space size is $\boldsymbol{O((2B)^D)}$**.
  → solve the MDP exactly with DP.

How to choose the B candidate actions/splits?

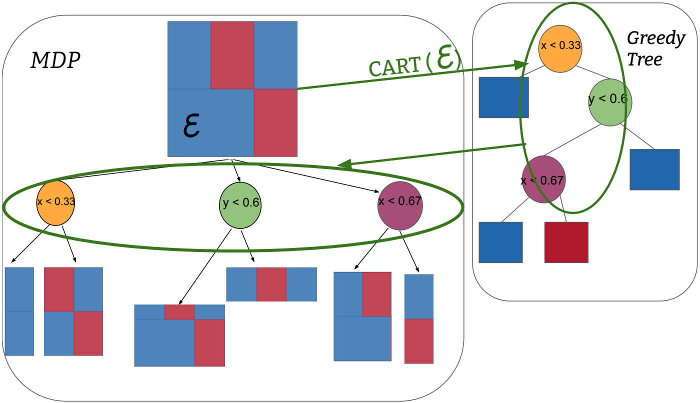
Top-B greedy splits (Blanc et al. 2023), quantiles, random...

Practical implemenataion of DPDT



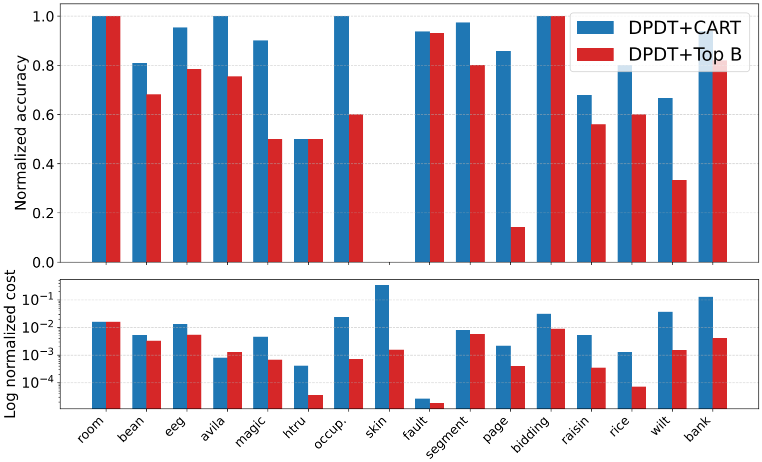
We can use greedy trees nodes as candidate actions.

Practical implemenataion of DPDT



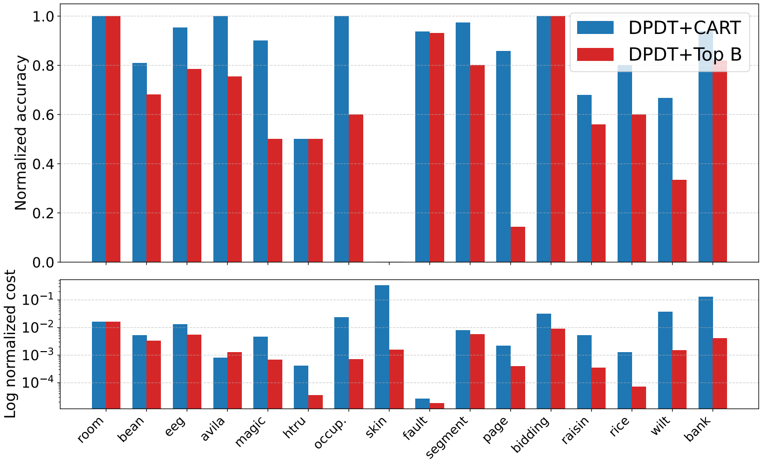
We can use greedy trees nodes as candidate actions.

Fast like greedy trees, accurate like optimal trees

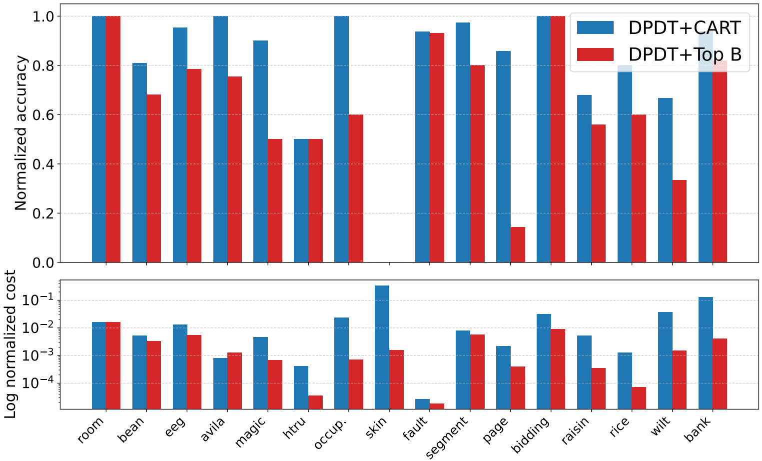


Train accuracies against cost for detph-3 trees.

Fast like greedy trees, accurate like optimal trees

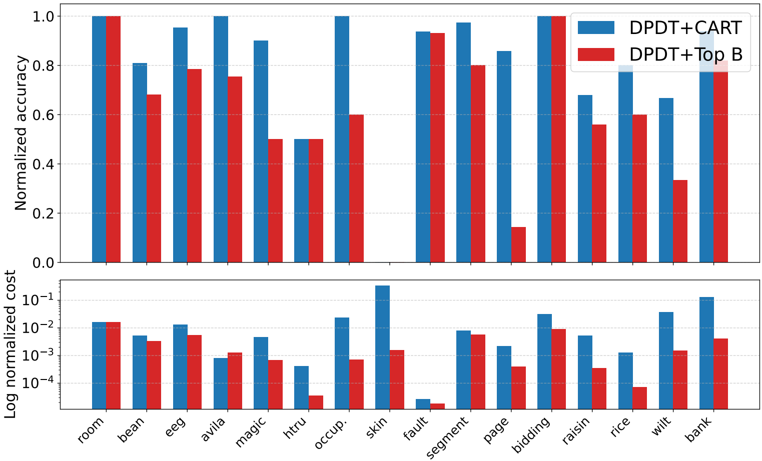


- DPDT trees can are not worse than greedy trees.



- DPDT trees can are not worse than greedy trees.
- DPDT trees can be strictly better than greedy trees.

Fast like greedy trees, accurate like optimal trees



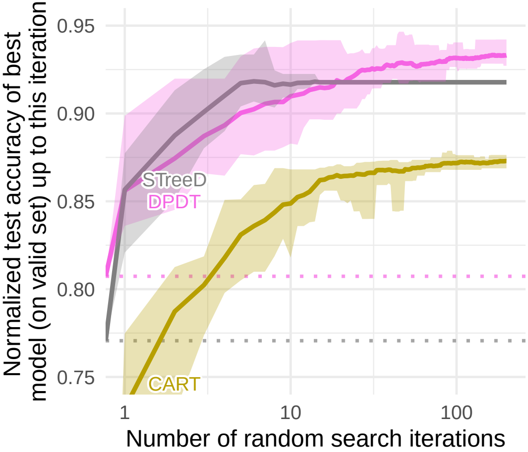
- DPDT trees can are not worse than greedy trees.
- DPDT trees can be strictly better than greedy trees.
- CART generates more diverse splits than Top B for DPDT.

Large scale evaluation of DPDT trees generalization

(Grinsztajn, Oyallon, and Varoquaux 2022)

Large scale evaluation of DPDT trees generalization

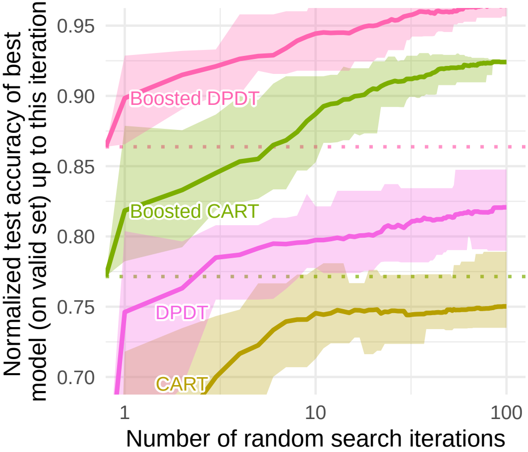
(Grinsztajn, Oyallon, and Varoquaux 2022)



DPDT depth-5 trees vs. other detph-5 trees

Large scale evaluation of DPDT trees generalization

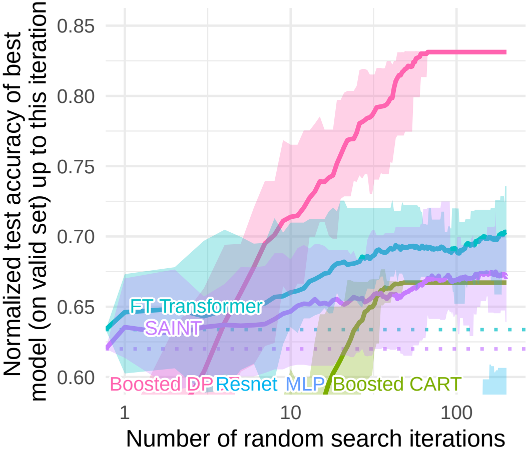
(Grinsztajn, Oyallon, and Varoquaux 2022)



Boosted DPDT vs. Boosted CART

Large scale evaluation of DPDT trees generalization

(Grinsztajn, Oyallon, and Varoquaux 2022)



Boosted DPDT vs. other classifiers

Perspectives

Perspectives

- New SOTA decision tree induction with dynamic programming in MDPs.

Perspectives

- New SOTA decision tree induction with dynamic programming in MDPs.
- What about using DPDT for indirect decision tree policy learning for SDM?

Perspectives

- New SOTA decision tree induction with dynamic programming in MDPs.
- What about using DPDT for indirect decision tree policy learning for SDM?
- What performances could we reach with an industry-grade implementation of XGboost+DPDT?

Perspectives

- New SOTA decision tree induction with dynamic programming in MDPs.
- What about using DPDT for indirect decision tree policy learning for SDM?
- What performances could we reach with an industry-grade implementation of XGboost+DPDT?

Let us take a step back

Q: Are decision trees really the most interpretable model?

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

- Interpretability $\simeq$ time to reproduce the computations.

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

- Interpretability $\simeq$ time to reproduce the computations.
- Interpretability $\simeq$ effort to read through the entire policy.

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

- Interpretability $\simeq$ **runtime in seconds?**
- Interpretability $\simeq$ effort to read through the entire policy.

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

- Interpretability $\simeq$ **runtime in seconds?**
- Interpretability $\simeq$ **size in bytes?**

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

- Interpretability $\simeq$ **runtime in seconds?**
- Interpretability $\simeq$ **size in bytes?**

How to compare policy of different classes?

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

- Interpretability $\simeq$ **runtime in seconds?**
- Interpretability $\simeq$ **size in bytes?**

How to compare policy of different classes?

- Different hardwares (CPUs vs GPUs).

How to measure policy interpretability?

The notion of *simulatability* (Lipton 2018)

- Interpretability $\simeq$ **runtime in seconds?**
- Interpretability $\simeq$ **size in bytes?**

How to compare policy of different classes?

- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) (Luo et al. 2024)

We propose policy unfolding

```python
# Small ReLU MLP for Pendulum
def play(x):
    h_layer_0_0 = 1.68 * x[0] + -0.69 * x[1] + -0.74 * x[2] + -1.40
    h_layer_0_0 = max(0.0, h_layer_0_0)
    h_layer_0_1 = 0.20 * x[0] + 0.29 * x[1] + -0.021 * x[2] + 1.25
    h_layer_0_1 = max(0.0, h_layer_0_1)
    h_layer_0_2 = 0.33 * x[0] + -0.57 * x[1] + 0.47 * x[2] + 1.94
    h_layer_0_2 = max(0.0, h_layer_0_2)
    h_layer_0_3 = 1.39 * x[0] + 0.94 * x[1] + 0.50 * x[2] + -1.13
    h_layer_0_3 = max(0.0, h_layer_0_3)
    h_layer_1_0 = 1.16 * h_layer_0_0 + -1.59 * h_layer_0_1 + 0.95 * h_layer_0_2 +
      -1.22 * h_layer_0_3 + -0.54
    h_layer_1_0 = max(0.0, h_layer_1_0)
    h_layer_1_1 = -0.55 * h_layer_0_0 + 1.13 * h_layer_0_1 + -0.58 * h_layer_0_2
     + -0.72 * h_layer_0_3 + 1.56
    h_layer_1_1 = max(0.0, h_layer_1_1)
    h_layer_1_2 = 1.10 * h_layer_0_0 + -1.01 * h_layer_0_1 + 0.96 * h_layer_0_2 +
      -2.84 * h_layer_0_3 + -0.02
    h_layer_1_2 = max(0.0, h_layer_1_2)
    h_layer_1_3 = 0.27 * h_layer_0_0 + 0.44 * h_layer_0_1 + 0.39 * h_layer_0_2 +
     0.15 * h_layer_0_3 + -1.24
    h_layer_1_3 = max(0.0, h_layer_1_3)
    h_layer_2_0 = -2.80 * h_layer_1_0 + -0.60 * h_layer_1_1 + 3.07 * h_layer_1_2
     + -1.63 * h_layer_1_3 + -0.36
    y_0 = h_layer_2_0

    return [y_0]
```

Is time/size of unfolded policies a good proxy?

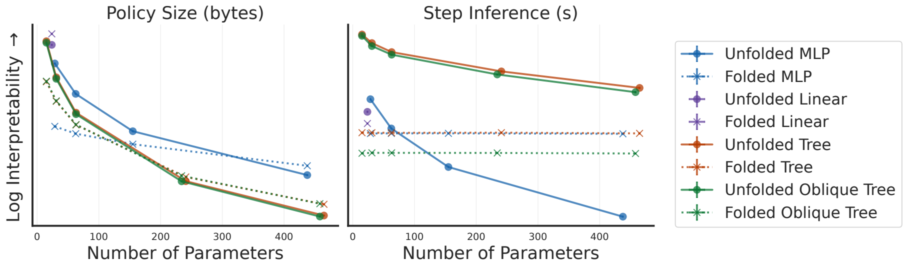
Setup

We imitate ~ 40000 expert policies from `stable-baselines3` using various policy classes/nb parameters on various environments.

Is time/size of unfolded policies a good proxy?

Setup

We imitate ~40000 expert policies from `stable-baselines3` using various policy classes/nb parameters on various environments.

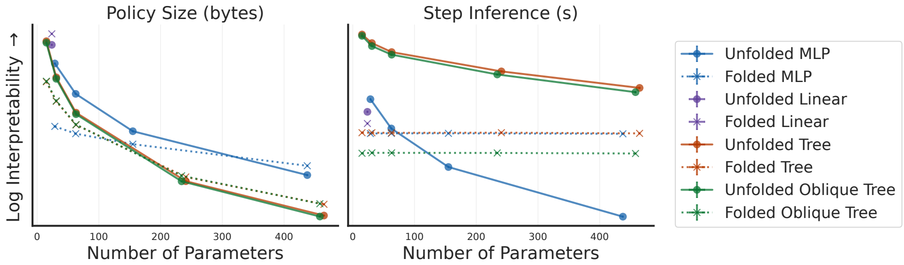


Aggregated policies interpretability on classic control environments

Is time/size of unfolded policies a good proxy?

Setup

We imitate ~ 40000 expert policies from `stable-baselines3` using various policy classes/nb parameters on various environments.



Aggregated policies interpretability on classic control environments

Less parameters means more interpretability (Freitas 2014).

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- What about floating points precision?
- What about the cost of basic operations ($\times$ vs. $+$ vs. $\geq$)?

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- What about floating points precision?
- What about the cost of basic operations ($\times$ vs. $+$ vs. $\geq$)?

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- What about floating points precision?
- What about the cost of basic operations ($\times$ vs. $+$ vs. $\geq$)?

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- What about floating points precision?
- What about the cost of basic operations ($\times$ vs. $+$ vs. $\geq$)?

Conclusion: interpretable SDM is a difficult research topic

- **Technical challenges: Learning interpretable policies for SDM involves partial observability.**
  → Focus on indirect approaches and/or on POMDP research first?
  → **Created opportunities for new decision tree algos for classif/regression.**
- **Fundamental challenges: No concensus on interpretability definition.**
  → Keep exploring benchmarks for policy interpretability.
  → Discuss with the community (InterpPol workshop).

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary (Nagendran et al. 2024).

Conclusion: interpretable SDM is a difficult research topic

- **Technical challenges: Learning interpretable policies for SDM involves partial observability.**
  → Focus on indirect approaches and/or on POMDP research first?
  → **Created opportunities for new decision tree algos for classif/regression.**
- **Fundamental challenges: No concensus on interpretability definition.**
  → Keep exploring benchmarks for policy interpretability.
  → Discuss with the community (InterpPol workshop).

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary (Nagendran et al. 2024).

Conclusion: interpretable SDM is a difficult research topic

- **Technical challenges: Learning interpretable policies for SDM involves partial observability.**
  → Focus on indirect approaches and/or on POMDP research first?
  → **Created opportunities for new decision tree algos for classif/regression.**
- **Fundamental challenges: No concensus on interpretability definition.**
  → Keep exploring benchmarks for policy interpretability.
  → Discuss with the community (InterpPol workshop).

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary (Nagendran et al. 2024).

Conclusion: interpretable SDM is a difficult research topic

- **Technical challenges:** **Learning interpretable policies for SDM involves partial observability.**
  → Focus on indirect approaches and/or on POMDP research first?
  → **Created opportunities for new decision tree algos for classif/regression.**
- **Fundamental challenges:** **No concensus on interpretability definition**.
  → Keep exploring benchmarks for policy interpretability.
  → Discuss with the community (InterpPol workshop).

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary (Nagendran et al. 2024).

Conclusion: interpretable SDM is a difficult research topic

- **Technical challenges: Learning interpretable policies for SDM involves partial observability.**
  → Focus on indirect approaches and/or on POMDP research first?
  → **Created opportunities for new decision tree algos for classif/regression.**
- **Fundamental challenges: No concensus on interpretability definition.**
  → Keep exploring benchmarks for policy interpretability.
  → Discuss with the community (InterpPol workshop).

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary (Nagendran et al. 2024).

Conclusion: interpretable SDM is a difficult research topic

- **Technical challenges:** **Learning interpretable policies for SDM involves partial observability.**
  → Focus on indirect approaches and/or on POMDP research first?
  → **Created opportunities for new decision tree algos for classif/regression.**
- **Fundamental challenges:** **No concensus on interpretability definition**.
  → Keep exploring benchmarks for policy interpretability.
  → Discuss with the community (InterpPol workshop).

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary (Nagendran et al. 2024).

Broader perspectives

- **Deep learning:** Can we design deep learning layers that take datasets and output candidate splits?
- **Combinatorial optimization:** Can we formulate other combinatorial/NP-hard problems as MDPs and design other DPDT-like algorithms?
- **Human-computer interaction:** Can we do large scale human study of the ~40K programs interpertability?

Broader perspectives

- **Deep learning:** Can we design deep learning layers that take datasets and output candidate splits?
- **Combinatorial optimization:** Can we formulate other combinatorial/NP-hard problems as MDPs and design other DPDT-like algorithms?
- **Human-computer interaction:** Can we do large scale human study of the ~40K programs interpertability?

Broader perspectives

- **Deep learning:** Can we design deep learning layers that take datasets and output candidate splits?
- **Combinatorial optimization:** Can we formulate other combinatorial/NP-hard problems as MDPs and design other DPDT-like algorithms?
- **Human-computer interaction:** Can we do large scale human study of the ~40K programs interpertability?

Broader perspectives

- **Deep learning:** Can we design deep learning layers that take datasets and output candidate splits?
- **Combinatorial optimization:** Can we formulate other combinatorial/NP-hard problems as MDPs and design other DPDT-like algorithms?
- **Human-computer interaction:** Can we do large scale human study of the ~40K programs interpertability?

- Baisero, Andrea and Christopher Amato (2022). “Unbiased Asymmetric Reinforcement Learning under Partial Observability”. In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS ’22. Virtual Event, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, pp. 44–52. ISBN: 9781450392136.
- Baisero, Andrea, Brett Daley, and Christopher Amato (Jan. 2022). “Asymmetric DQN for partially observable reinforcement learning”. In: *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*. Ed. by James Cussens and Kun Zhang. Vol. 180. Proceedings of Machine Learning Research. PMLR, pp. 107–117. URL: `https://proceedings.mlr.press/v180/baisero22a.html`.
- Barceló, Pablo et al. (2020). “Model interpretability through the lens of computational complexity”. In: *Advances in neural information processing systems*.
- Bastani, Osbert, Yewen Pu, and Armando Solar-Lezama (2018). “Verifiable Reinforcement Learning via Policy Extraction”. In.
- Bertsimas, Dimitris and Jack Dunn (2017). “Optimal classification trees”. In: *Machine Learning* 106, pp. 1039–1082.

Blanc, Guy et al. (2023). “Harnessing the power of choices in decision tree learning”. In: *Advances in Neural Information Processing Systems* 36, pp. 80220–80232.

Breiman, L et al. (1984). *Classification and Regression Trees*. Wadsworth.

Doshi-Velez, Finale and Been Kim (2017). “Towards A Rigorous Science of Interpretable Machine Learning”. In: arXiv: 1702.08608 [stat.ML]. URL: https://arxiv.org/abs/1702.08608.

Freitas, Alex A. (Mar. 2014). “Comprehensible classification models: a position paper”. In: *SIGKDD Explor. Newsl.* 15.1, pp. 1–10. ISSN: 1931-0145. DOI: 10.1145/2594473.2594475. URL: https://doi.org/10.1145/2594473.2594475.

Glanois, Claire et al. (2024). “A survey on interpretable reinforcement learning”. In: *Machine Learning*, pp. 1–44.

Greydanus, Sam et al. (2018). *Visualizing and Understanding Atari Agents*.

Grinsztajn, Léo, Edouard Oyallon, and Gaël Varoquaux (2022). “Why do tree-based models still outperform deep learning on typical tabular data?" In: *Advances in neural information processing systems* 35, pp. 507–520.

Lambrechts, Gaspard, Damien Ernst, and Aditya Mahajan (2025). "A Theoretical Justification for Asymmetric Actor-Critic algorithms". In: *Forty-second International Conference on Machine Learning*. URL: https://openreview.net/forum?id=F1yANMCnAn.

Lipton, Zachary C. (2018). "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery.". In: *Queue* 16.3, pp. 31–57.

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

Luo, Lirui et al. (2024). "End-to-End Neuro-Symbolic Reinforcement Learning with Textual Explanations". In: *International Conference on Machine Learning (ICML)*.

Marton, Sascha et al. (2025). "Mitigating Information Loss in Tree-Based Reinforcement Learning via Direct Optimization". In: URL: https://openreview.net/forum?id=qpXctF2aLZ.

Nagendran, Myura et al. (2024). "Eye tracking insights into physician behaviour with safe and unsafe explainable AI recommendations". In: *NPJ Digital Medicine* 7.1, p. 202.

Pinto, Lerrel et al. (2017). *Asymmetric Actor Critic for Image-Based Robot Learning*. arXiv: 1710.06542 [cs.RO]. URL: https://arxiv.org/abs/1710.06542.

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

Schulman, John et al. (2017). "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347*.

Singh, Satinder P., Tommi S. Jaakkola, and Michael I. Jordan (1994). "Learning without state-estimation in partially observable Markovian decision processes". In: *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*. ICML'94. New Brunswick, NJ, USA: Morgan Kaufmann Publishers Inc., pp. 284–292. ISBN: 1558603352.

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

Topin, Nicholay et al. (2021). "Iterative bounding mdps: Learning interpretable policies via non-interpretable methods". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35, pp. 9923–9931.

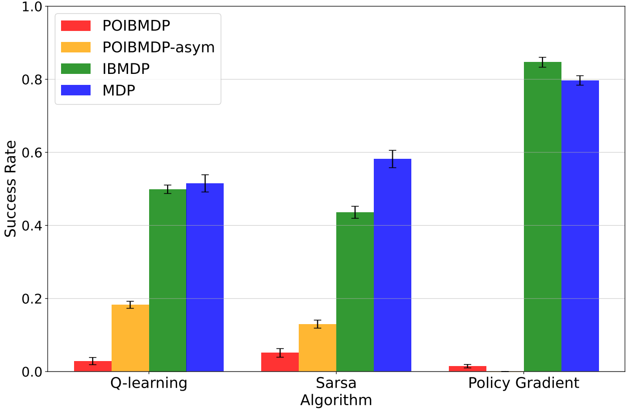
Wu, Mike et al. (Apr. 2020). "Regional Tree Regularization for Interpretability in Deep Neural Networks". In: 34, pp. 6413–6421. DOI: 10.1609/aaai.v34i04.6112. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6112.

Result: for similar problems, RL struggles more when there is partial observability

Success rates over thousands of RL runs with varying hyperparameters when learning different policies in the same IBMDP[1].

[1] We also observed similar results on classic controls and variants of the grid world MDP.

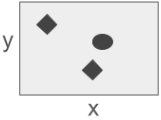
Result: for similar problems, RL struggles more when there is partial observability



Success rates over thousands of RL runs with varying hyperparameters when learning different policies in the same IBMDP[1].

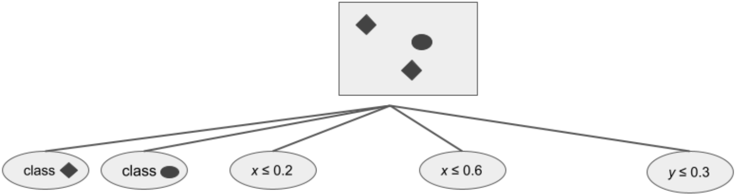
[1] We also observed similar results on classic controls and variants of the grid world MDP.

Decision tree induction as solving MDPs



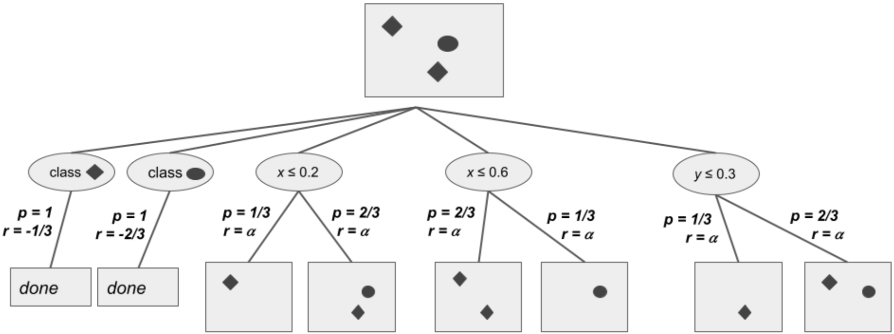
Example of decision tree induction as an MDP.

Decision tree induction as solving MDPs



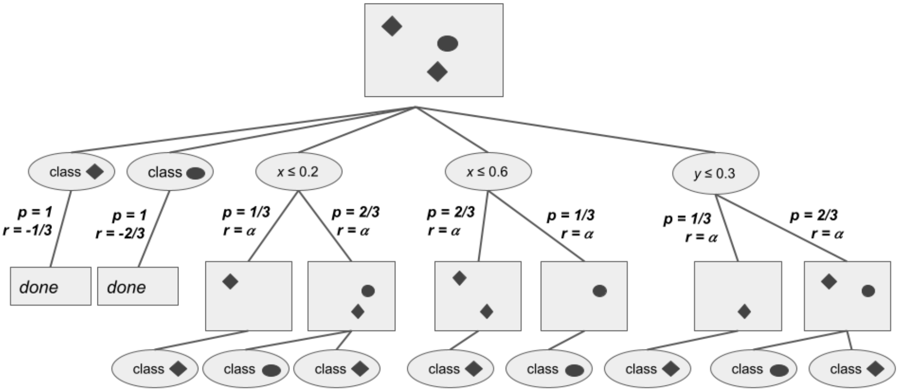
Example of decision tree induction as an MDP.

Decision tree induction as solving MDPs



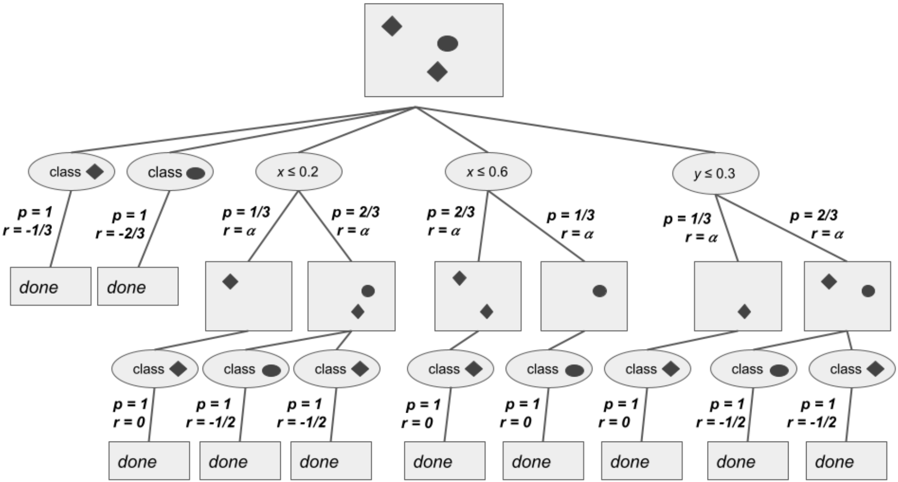
Example of decision tree induction as an MDP.

Decision tree induction as solving MDPs



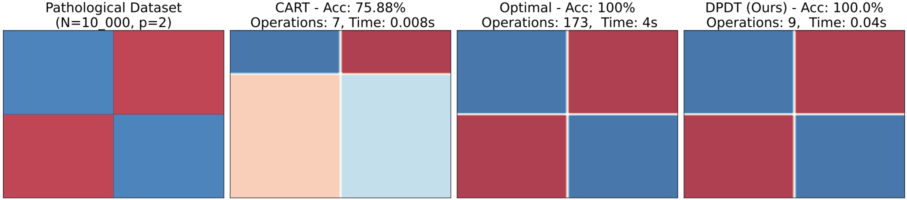
Example of decision tree induction as an MDP.

Decision tree induction as solving MDPs



Example of decision tree induction as an MDP.

Fast like greedy trees, accurate like optimal trees



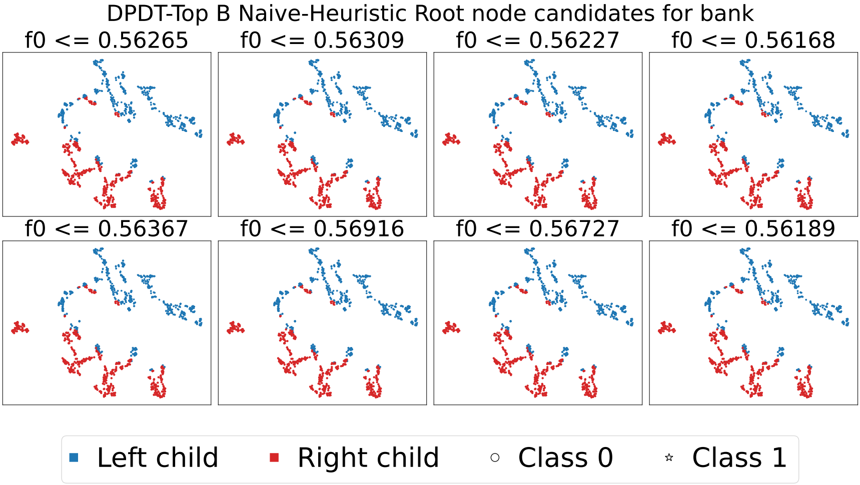
Comparison of greedy, optimal, and DPDT depth-2 trees on the checkersboard dataset.

Fast like greedy trees, accurate like optimal trees

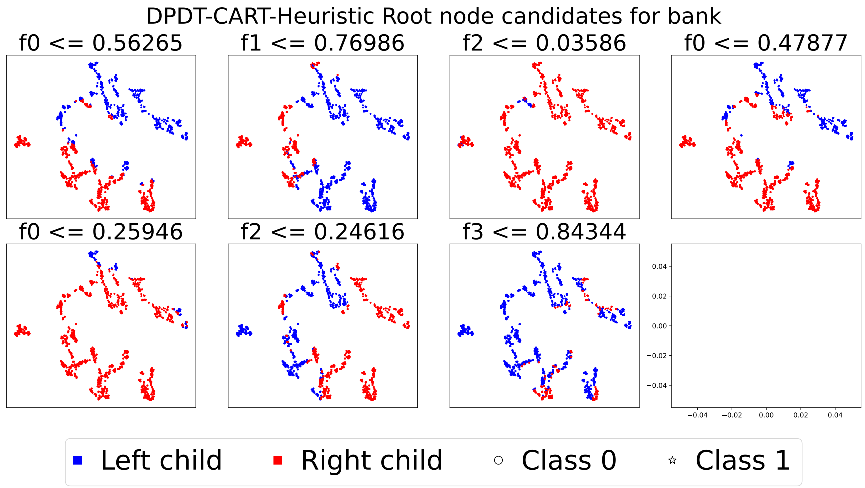
Comparison of accuracies and operations for depth-3 trees.

	Accuracy						Operations					
Dataset	Opt	Greedy	DPDT				Opt	Greedy	DPDT			
			CART$^-$	CART$^+$	TopB$^-$	TopB$^+$			CART$^-$	CART$^+$	TopB$^-$	TopB$^+$
room	**0.992**	0.968	0.991	**0.992**	0.990	**0.992**	10^6	15	286	16100	111	16100
bean	**0.871**	0.777	0.812	0.853	0.804	0.841	$5 \cdot 10^6$	15	295	25900	112	16800
eeg	**0.708**	0.666	0.689	0.706	0.684	0.699	$2 \cdot 10^6$	13	289	26000	95	11000
avila	**0.585**	0.532	0.574	**0.585**	0.563	0.572	$3 \cdot 10^7$	9	268	24700	60	38900
magic	**0.831**	0.801	0.822	0.828	0.807	0.816	$6 \cdot 10^6$	15	298	28000	70	4190
htru	**0.981**	0.979	0.979	0.980	0.979	0.980	$6 \cdot 10^7$	15	295	25300	55	2180
occup.	**0.994**	0.989	0.991	**0.994**	0.990	0.992	$7 \cdot 10^5$	13	280	16300	33	510
skin	**0.969**	0.966	0.966	0.966	0.966	0.966	$7 \cdot 10^4$	15	301	23300	20	126
fault	**0.682**	0.553	0.672	0.674	0.672	0.673	$9 \cdot 10^8$	13	295	24200	111	16800
segment	**0.887**	0.574	0.812	0.879	0.786	0.825	$2 \cdot 10^6$	7	220	16300	68	11400
page	**0.971**	0.964	0.970	0.970	0.964	0.965	10^7	15	298	22400	701	4050
bidding	**0.993**	0.981	0.985	**0.993**	0.985	**0.993**	$3 \cdot 10^5$	13	256	9360	58	2700
raisin	**0.894**	0.869	0.879	0.886	0.875	0.883	$4 \cdot 10^6$	15	295	20900	48	1440
rice	**0.938**	0.933	0.934	0.937	0.933	0.936	$2 \cdot 10^7$	15	298	25500	49	1470
wilt	**0.996**	0.993	0.994	0.995	0.994	0.994	$3 \cdot 10^5$	13	274	11300	33	465
bank	**0.983**	0.933	0.971	0.980	0.951	0.974	$6 \cdot 10^4$	13	271	7990	26	256

CART generates more diverse splits than Top B



CART generates more diverse splits than Top B



Why generating candidate splits with CART?

Theorem (DPDT trees are not worse than greedy trees)

The greedy tree is always a solution of the MDPs we solve. Because we solve the MDPs exactly with DP, if the greedy tree is the best solution, DPDT will find it.

Theorem (DPDT trees can be strictly better than greedy trees)

There exist a depth budget D and a dataset for which DPDT trees are strictly better than greedy trees.[a]

[a] cf. checkersboard dataset.

Why generating candidate splits with CART?

Theorem (DPDT trees are not worse than greedy trees)

The greedy tree is always a solution of the MDPs we solve. Because we solve the MDPs exactly with DP, if the greedy tree is the best solution, DPDT will find it.

Theorem (DPDT trees can be strictly better than greedy trees)

There exist a depth budget D and a dataset for which DPDT trees are strictly better than greedy trees.[a]

[a] cf. checkersboard dataset.

Why generating candidate splits with CART?

Theorem (DPDT trees are not worse than greedy trees)

The greedy tree is always a solution of the MDPs we solve. Because we solve the MDPs exactly with DP, if the greedy tree is the best solution, DPDT will find it.

Theorem (DPDT trees can be strictly better than greedy trees)

There exist a depth budget D and a dataset for which DPDT trees are strictly better than greedy trees.[a]

[a] cf. checkersboard dataset.

Why generating candidate splits with CART?

Theorem (DPDT trees are not worse than greedy trees)

The greedy tree is always a solution of the MDPs we solve. Because we solve the MDPs exactly with DP, if the greedy tree is the best solution, DPDT will find it.

Theorem (DPDT trees can be strictly better than greedy trees)

There exist a depth budget D and a dataset for which DPDT trees are strictly better than greedy trees.[a]

[a] cf. checkersboard dataset.

Why generating candidate splits with CART?

Theorem (DPDT trees are not worse than greedy trees)

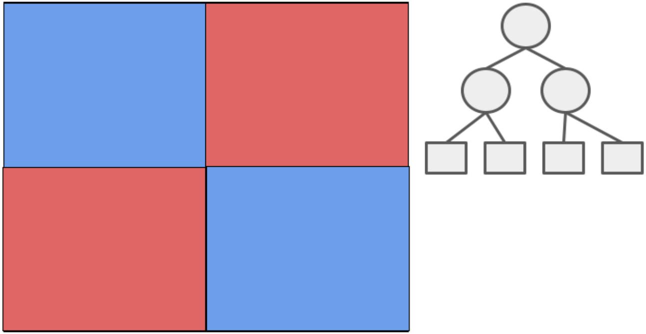
The greedy tree is always a solution of the MDPs we solve. Because we solve the MDPs exactly with DP, if the greedy tree is the best solution, DPDT will find it.

Theorem (DPDT trees can be strictly better than greedy trees)

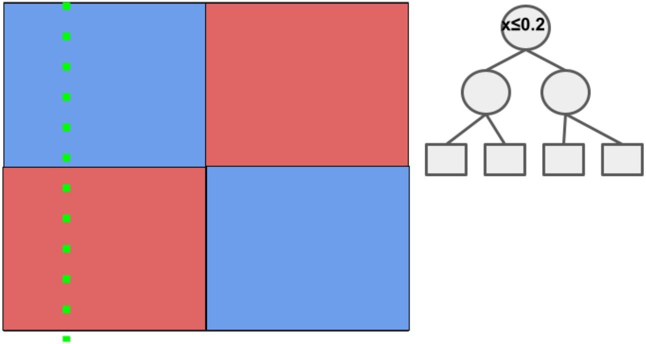
There exist a depth budget D and a dataset for which DPDT trees are strictly better than greedy trees.[a]

[a] cf. checkersboard dataset.

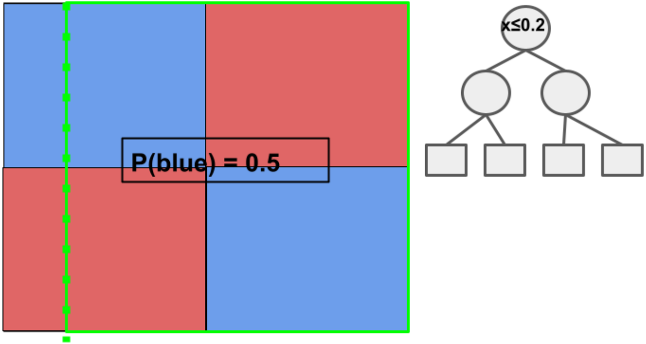
DPDT trees can be strictly better than greedy trees



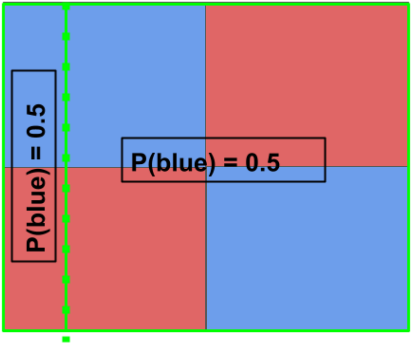
DPDT trees can be strictly better than greedy trees

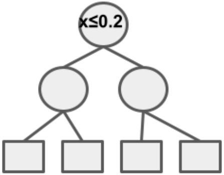


DPDT trees can be strictly better than greedy trees

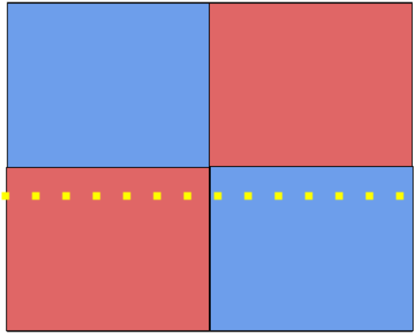


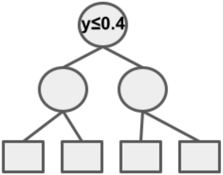
DPDT trees can be strictly better than greedy trees



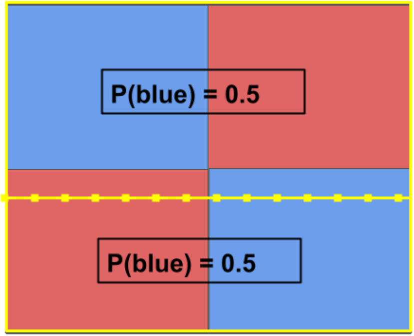


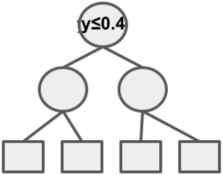
DPDT trees can be strictly better than greedy trees



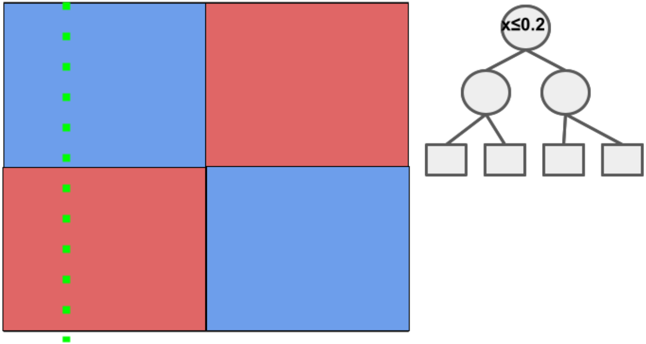


DPDT trees can be strictly better than greedy trees

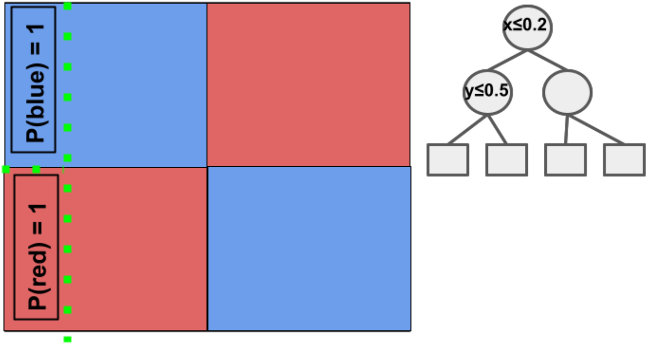




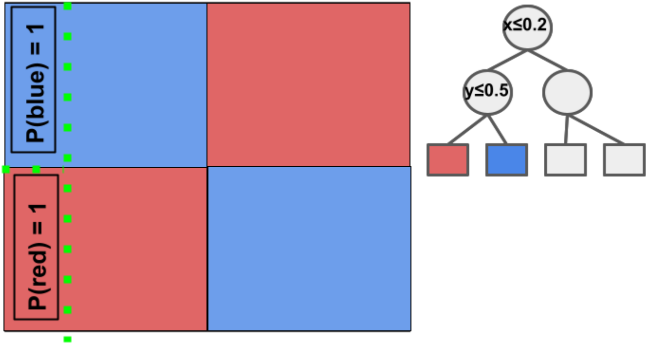
DPDT trees can be strictly better than greedy trees



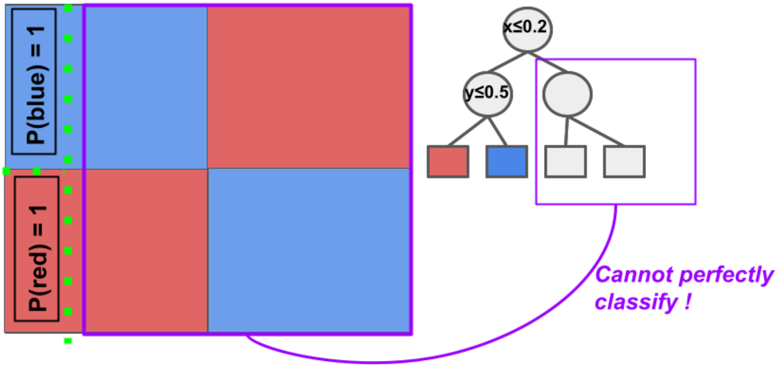
DPDT trees can be strictly better than greedy trees



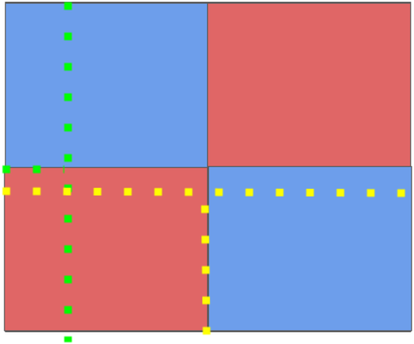
DPDT trees can be strictly better than greedy trees

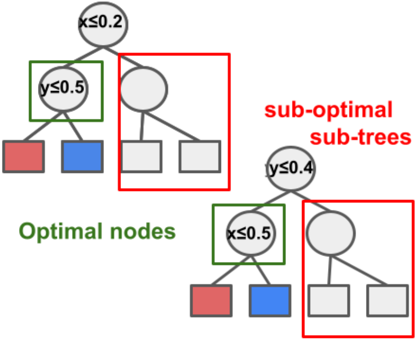


DPDT trees can be strictly better than greedy trees

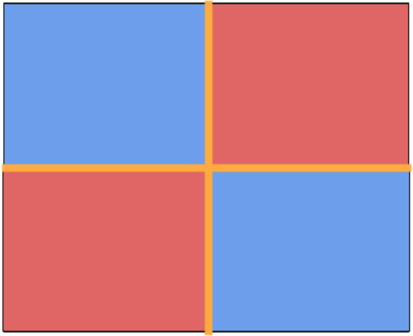


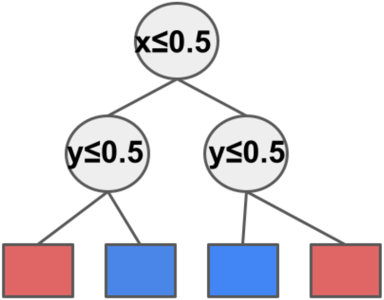
DPDT trees can be strictly better than greedy trees



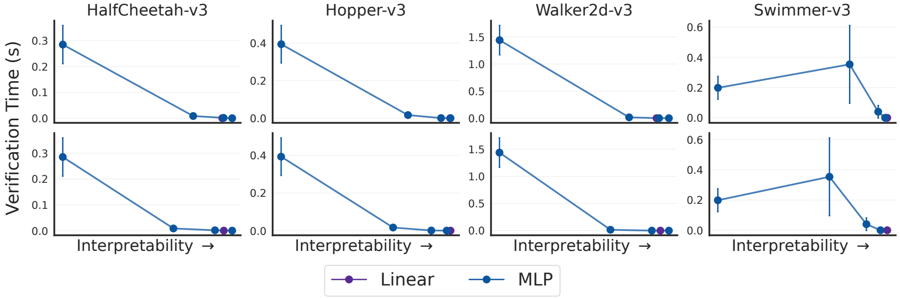


DPDT trees can be strictly better than greedy trees



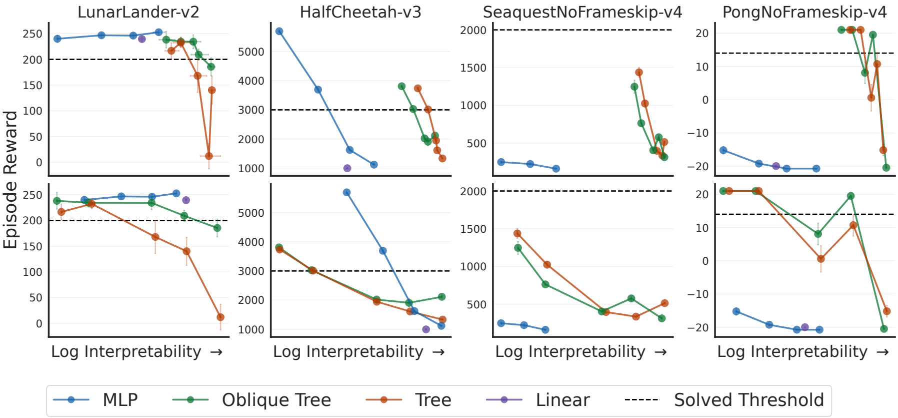


Result: verification time does scale with step inference time



Verification time as a function of policy interpretability. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size.

Result: there is no dominating policy class for all environments



Interpretability-Performance trade-offs. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size.

We propose policy unfolding

```python
# Decision tree for Mountain Car
def play(x):
    if x[1] <= -0.2597:
        if x[1] <= -0.6378:
            return 0
        else:
            if x[0] <= -1.0021:
                return 2
            else:
                return 0
    else:
        if x[1] <= -0.0508:
            if x[0] <= 0.2979:
                if x[0] <= 0.0453:
                    return 2
                else:
                    if x[1] <=
   -0.2156:
                        return 0
                    else:
                        return 2
            else:
                return 0
        else:
            return 2
```

```python
# Small ReLU MLP for Pendulum
def play(x):
    h_layer_0_0 = 1.238*x[0]+0.971*x
      [1]
                  +0.430*x[2]+0.933
    h_layer_0_0 = max(0, h_layer_0_0
     )
    h_layer_0_1 = -1.221*x[0]+1.001
                  *x[1]-0.423*x[2]
                  +0.475
    h_layer_0_1 = max(0, h_layer_0_1
     )
    h_layer_1_0 = -0.109*h_layer_0_0
                  -0.377*h_layer_0_1
                  +1.694
    h_layer_1_0 = max(0, h_layer_1_0
     )
    h_layer_1_1 = -3.024*h_layer_0_0
                  -1.421*h_layer_0_1
                  +1.530
    h_layer_1_1 = max(0, h_layer_1_1
     )

    h_layer_2_0 = -1.790*h_layer_1_0
                  +2.840*h_layer_1_1
                  +0.658
    y_0 = h_layer_2_0
    return [y_0]
```