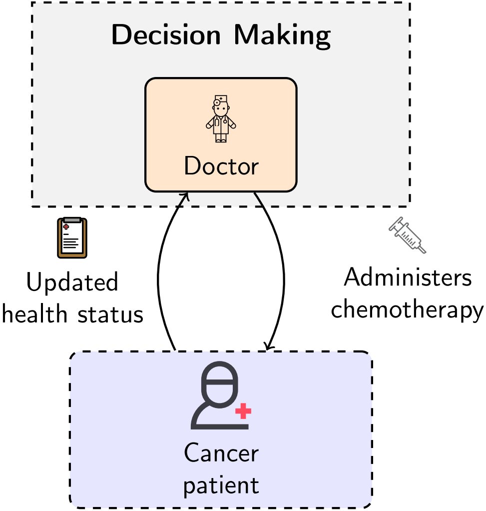
Interpretability, Decision Trees, and Sequential Decision Making

Hector Kohler

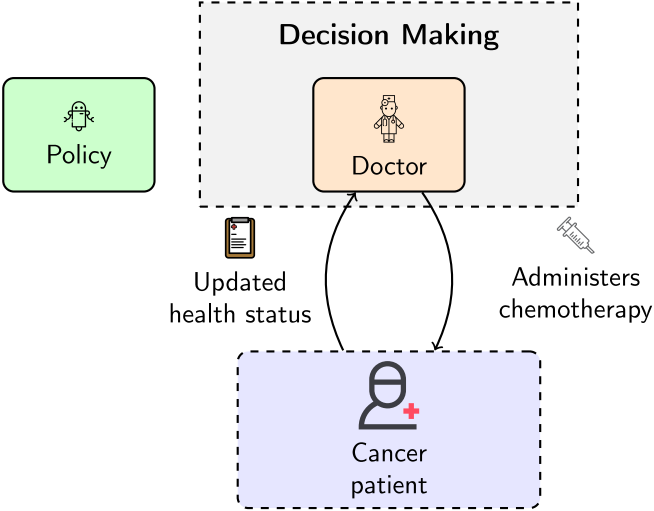
Supervised by Dr. Riad Akrour (HdR) and Prof. Philippe Preux (HdR)
Université de Lille, CNRS, Inria, UMR CRIStAL 9189, France

November 27, 2025

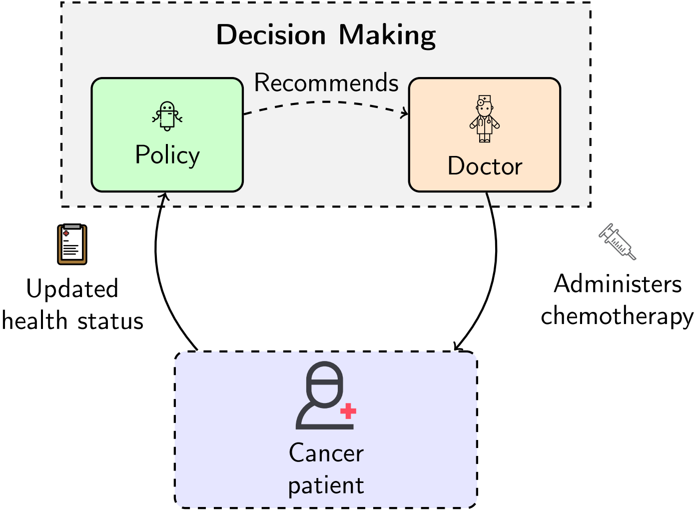
Sequential decision making (SDM) and machine learning (ML)



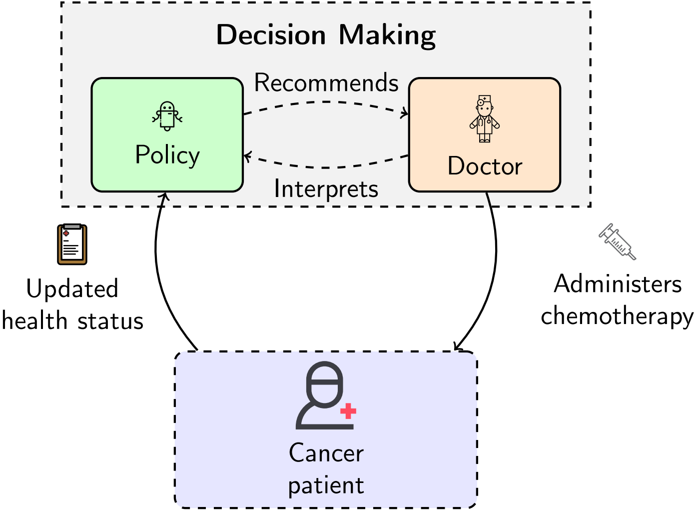
Sequential decision making (SDM) and machine learning (ML)



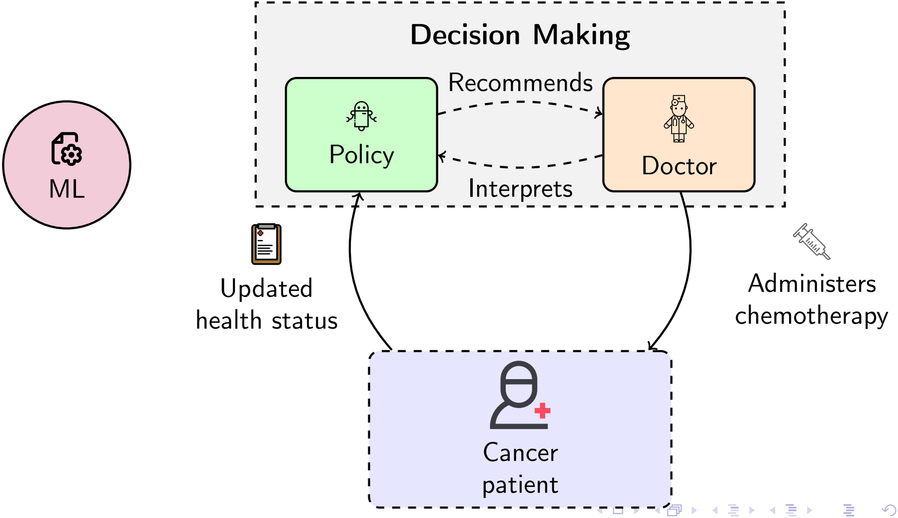
Sequential decision making (SDM) and machine learning (ML)



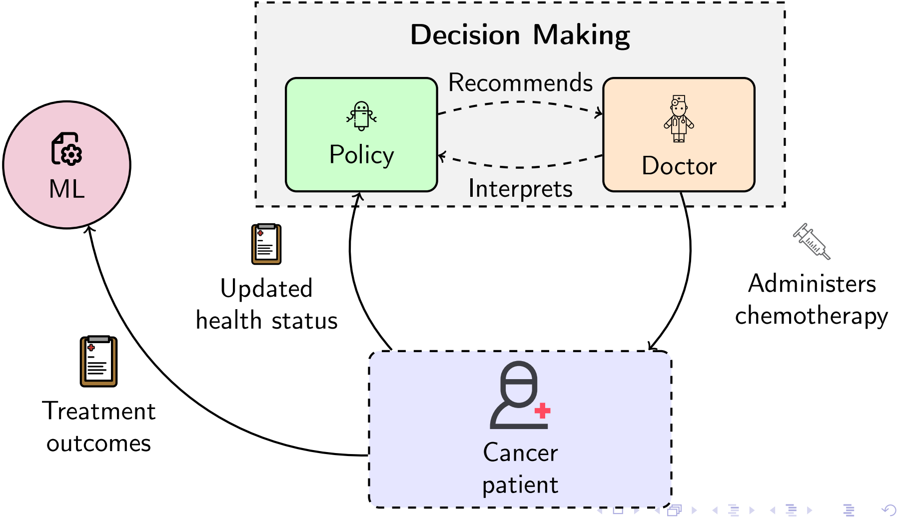
Sequential decision making (SDM) and machine learning (ML)



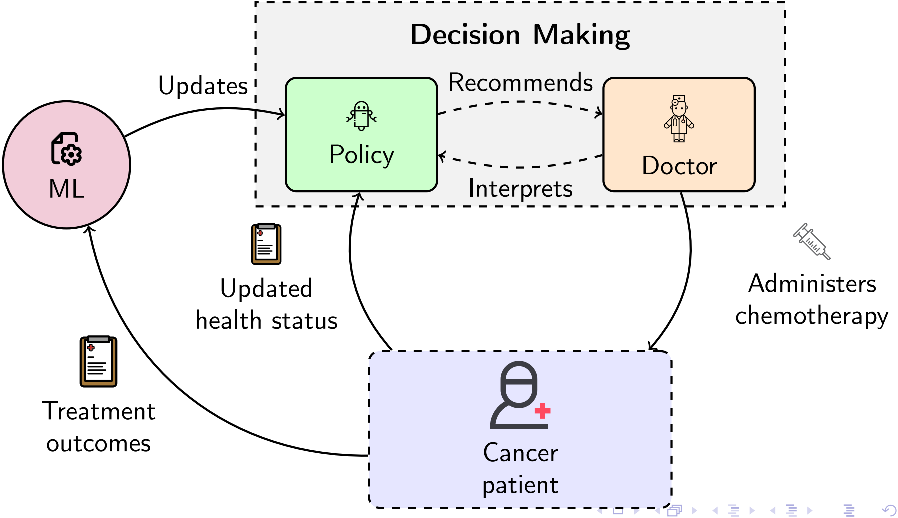
Sequential decision making (SDM) and machine learning (ML)



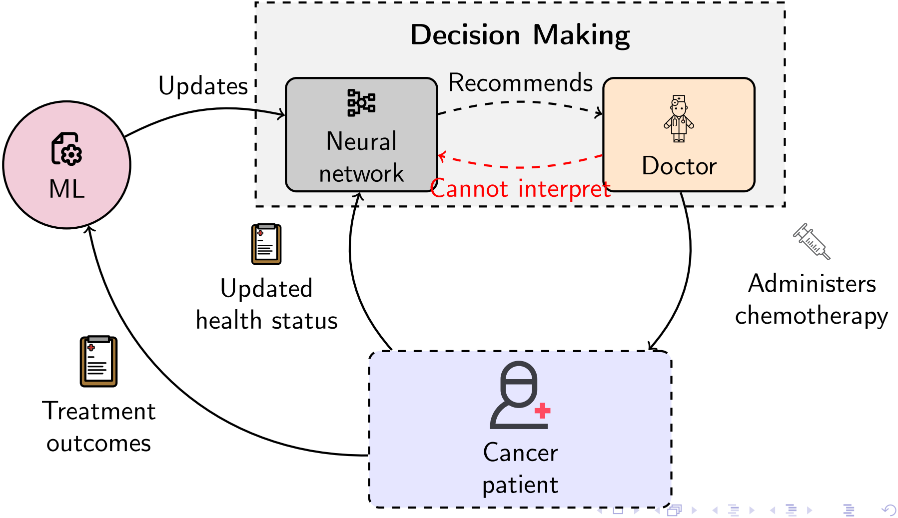
Sequential decision making (SDM) and machine learning (ML)



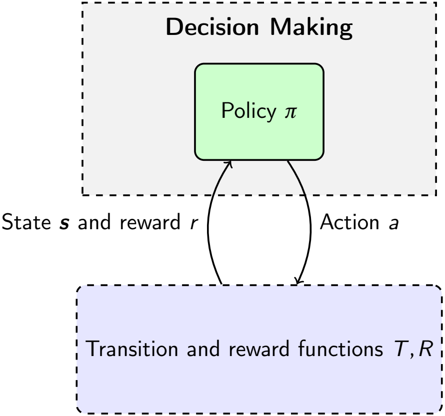
Sequential decision making (SDM) and machine learning (ML)



Sequential decision making (SDM) and machine learning (ML)

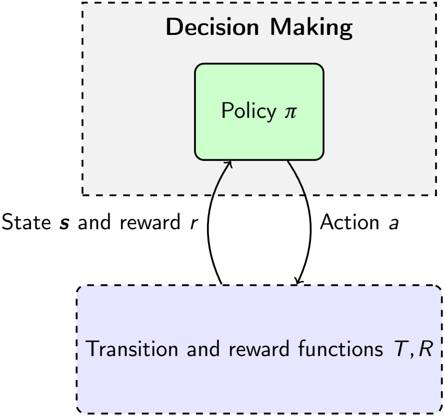


Markov decision processes (MDPs) and reinforcement learning (RL)



Markov decision processes [Put94].

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

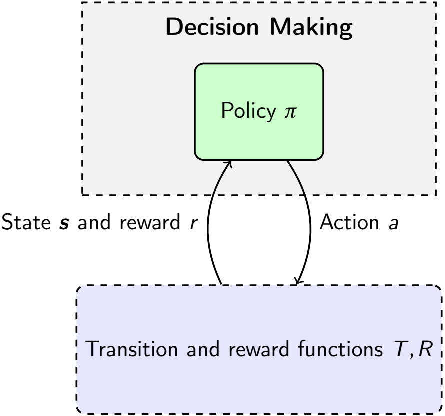


Markov decision processes [Put94].

- RL [SB98] aims to find a policy, $\pi : S \to A$ that maximizes:

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

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



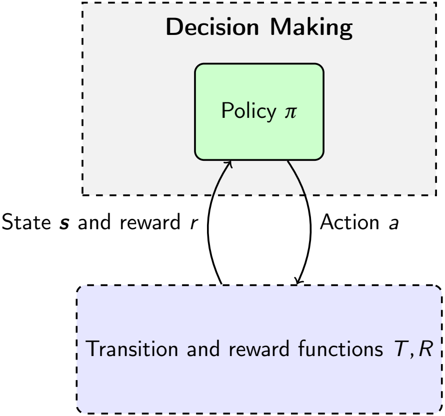
Markov decision processes [Put94].

- RL [SB98] aims to find a policy, $\pi : S \to A$ that maximizes:

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

- Lots of succesful RL algorithms [SB98; Mni+15; Sch+17].

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



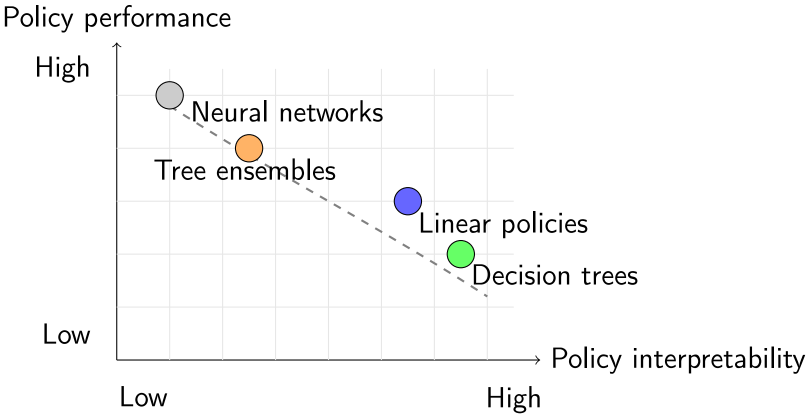
Markov decision processes [Put94].

- RL [SB98] aims to find a policy, $\pi : S \to A$ that maximizes:

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

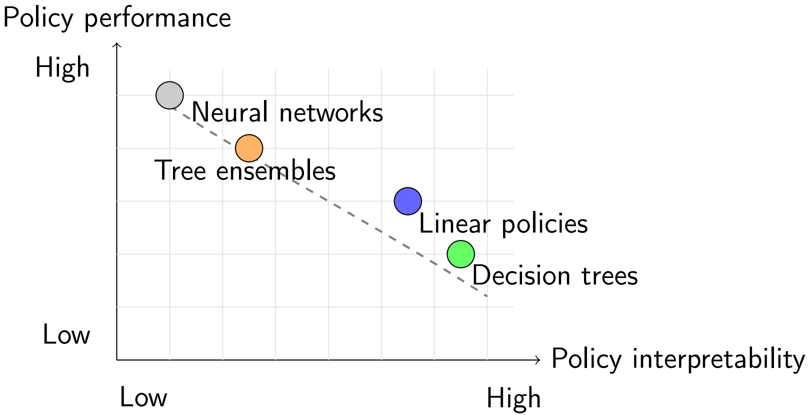
- Lots of succesful RL algorithms [SB98; Mni+15; Sch+17].
- Few interpretability concerns.

Policy interpretability



Heuristic interpretability-performance trade-offs of different policy classes.

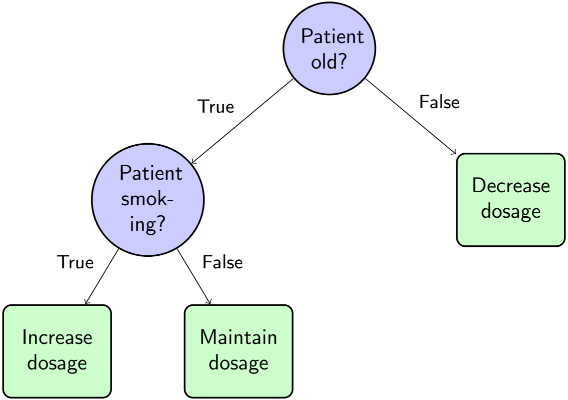
Policy interpretability



Heuristic interpretability-performance trade-offs of different policy classes.

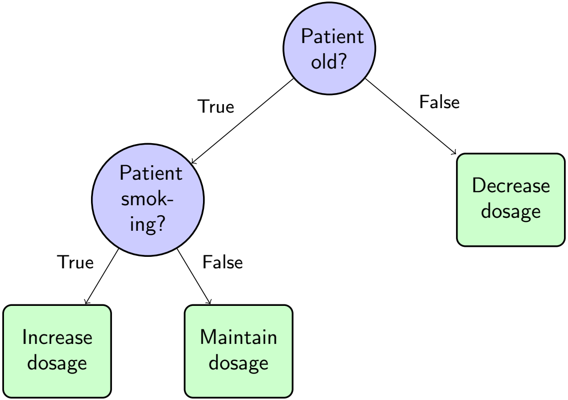
⚠ **No definition of interpretability for machine learning models!**

Decision trees



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

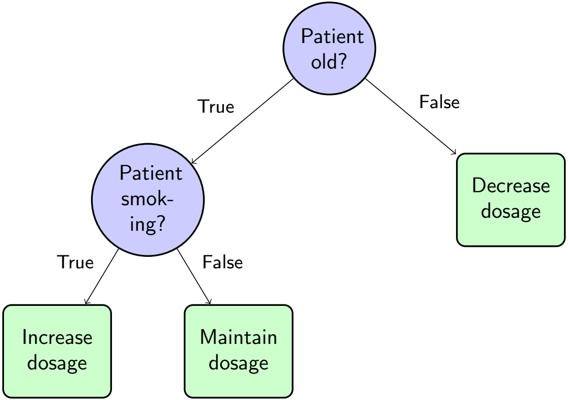
Decision trees



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

Successful algorithms for classification/regression: [Bre+84; BD17; Dem+22; VZ19; MMW22] . . .

Decision trees

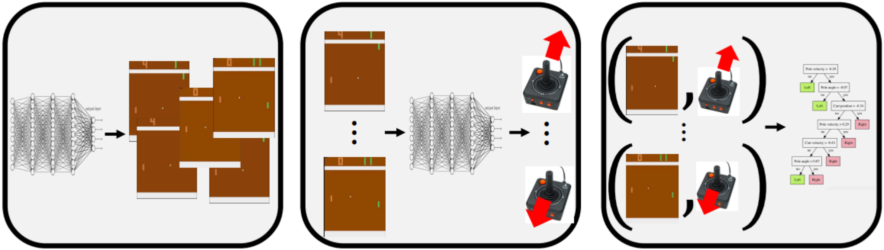


A generic decision tree of depth $D = 2$.

Successful algorithms for classification/regression: [Bre+84; BD17; Dem+22; VZ19; MMW22] . . .

What about SDM?

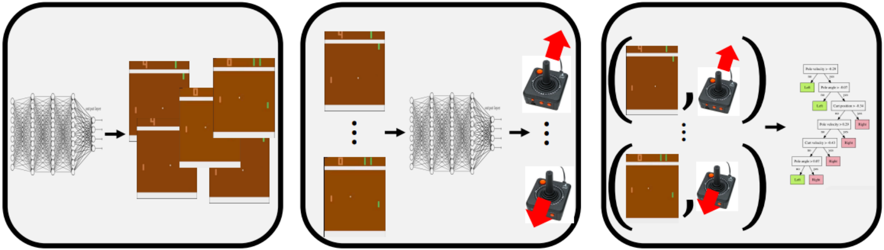
Indirect approach: imitation learning



Step 1: Use NN to generate states

Step 2: Use NN to obtain actions

Step 3: Use supervised learning to train a decision tree



Step 1: Use NN to generate states

Step 2: Use NN to obtain actions

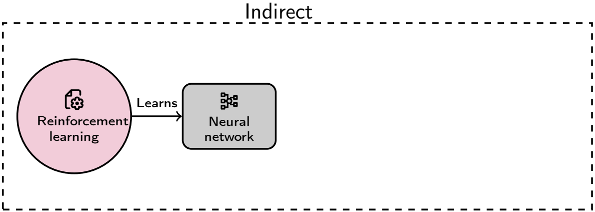
Step 3: Use supervised learning to train a decision tree

Most research focused on indirect learning of interpretable policies [RGB10; BPS18; Ver+18; Mil+24].

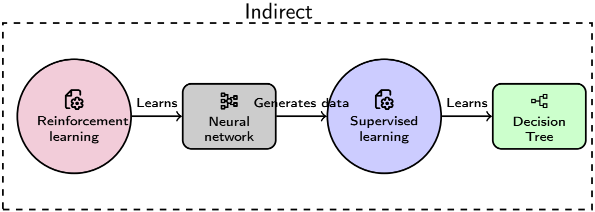
Two ways to get interpretable policies for SDM

Indirect

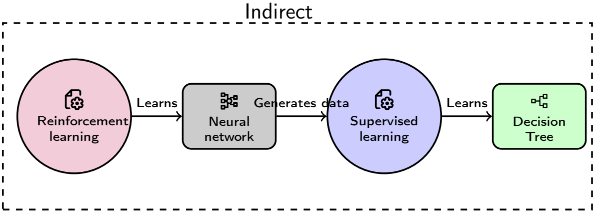
Two ways to get interpretable policies for SDM



Two ways to get interpretable policies for SDM

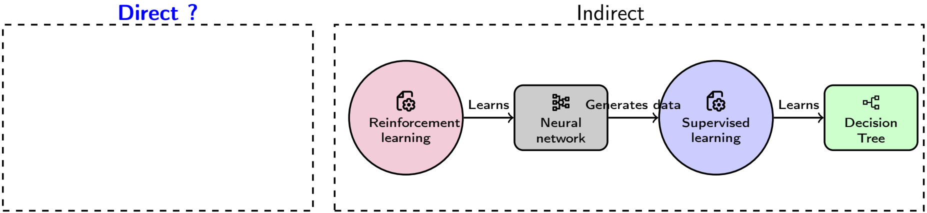


Two ways to get interpretable policies for SDM



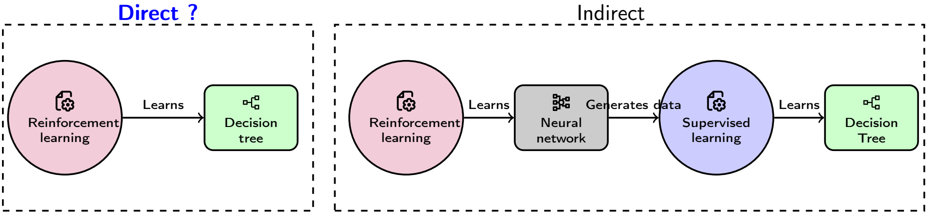
⚠**Policies obtained indirectly optimize a surrogate objective rather than an MDP cumulative rewards.**

Two ways to get interpretable policies for SDM



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

Two ways to get interpretable policies for SDM



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

Contributions

1. Can we directly optimize a trade-off of interpretability and performance in SDM?
2. Can we leverage SDM to learn interpretable classifiers for supervised learning?
3. How to measure policy interpretability in SDM?

Contributions

1. Can we directly optimize a trade-off of interpretability and performance in SDM?
2. Can we leverage SDM to learn interpretable classifiers for supervised learning?
3. How to measure policy interpretability in SDM?

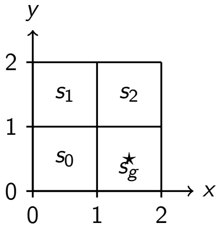
Contributions

1. Can we directly optimize a trade-off of interpretability and performance in SDM?
2. Can we leverage SDM to learn interpretable classifiers for supervised learning?
3. How to measure policy interpretability in SDM?

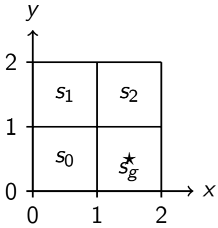
Contributions

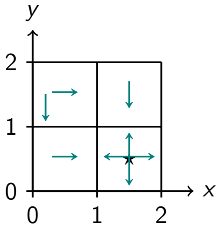
1. Can we directly optimize a trade-off of interpretability and performance in SDM?
2. Can we leverage SDM to learn interpretable classifiers for supervised learning?
3. How to measure policy interpretability in SDM?

Grid world MDP and decision tree policies



Grid world MDP and decision tree policies

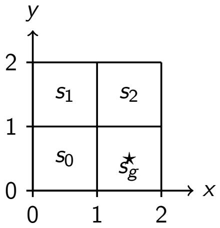




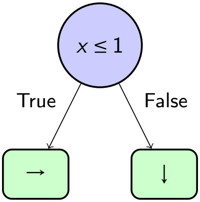
Grid world MDP and decision tree policies

Grid world MDP and optimal actions.

Grid world MDP and decision tree policies

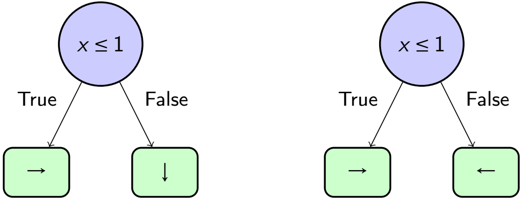


Grid world MDP and optimal actions.



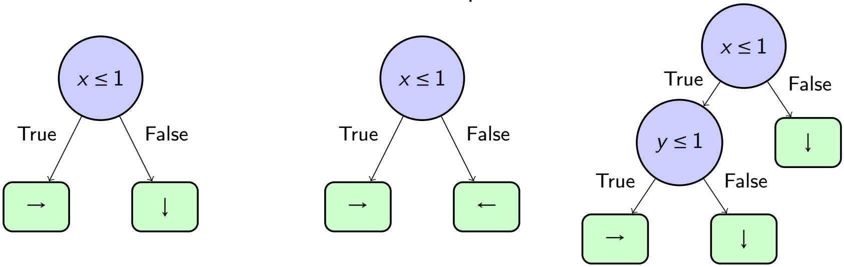
Grid world MDP and decision tree policies

Grid world MDP and optimal actions.



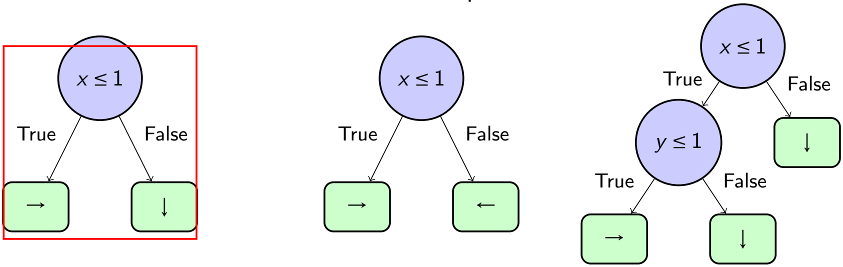
Grid world MDP and decision tree policies

Grid world MDP and optimal actions.



Grid world MDP and decision tree policies

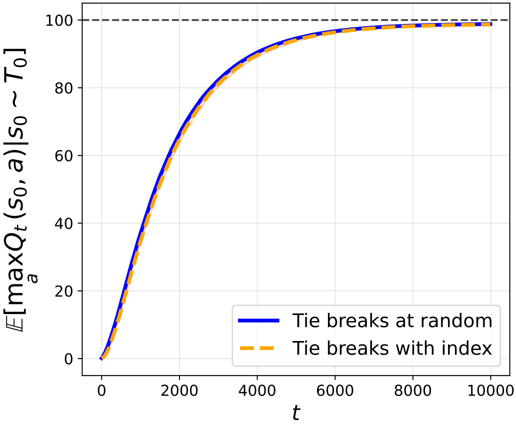
Grid world MDP and optimal actions.



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

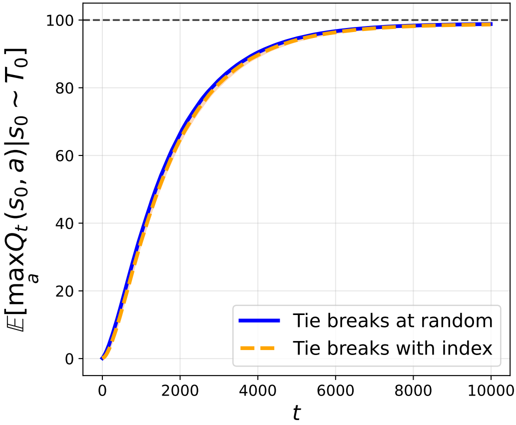
Grid world MDP and decision tree policies: indirect approach

Grid world MDP and decision tree policies: indirect approach

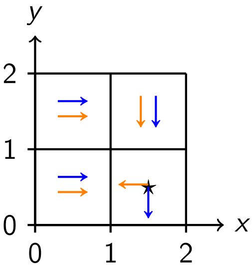


Sample complexity curve of Q-learning over 100 random seeds.

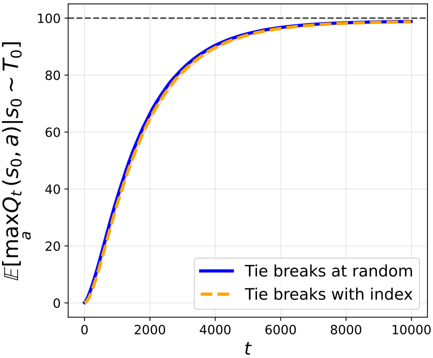
Grid world MDP and decision tree policies: indirect approach

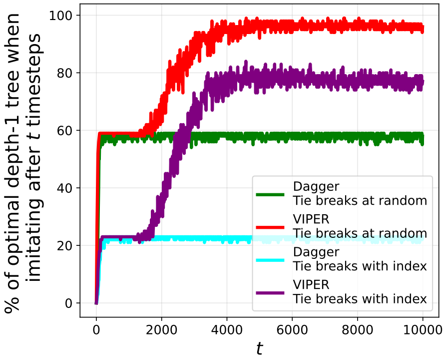


Sample complexity curve of Q-learning over 100 random seeds.



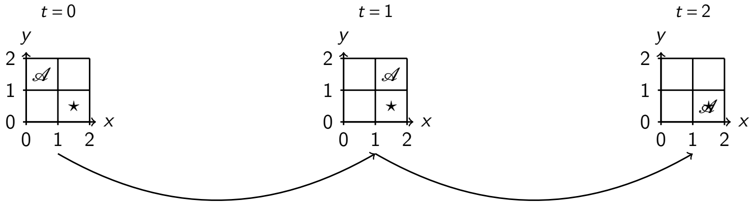
Expert policies.



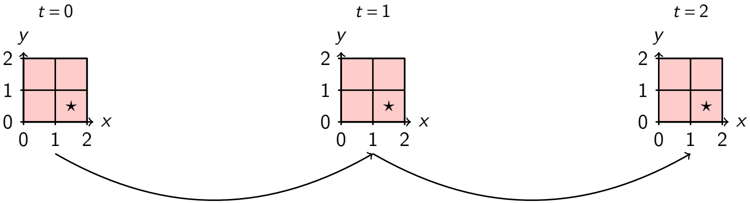


Sample complexity curve of Q-learning over 100 random seeds and performance of indirect interpretable methods when imitating the greedy policy with a tree at different Q-learning stages.

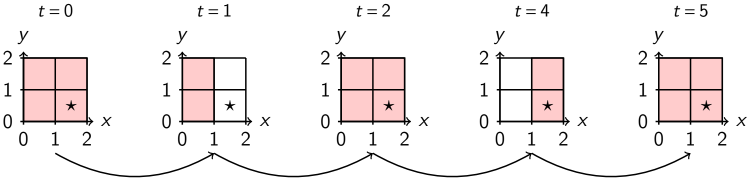
Iterative bounding Markov decision processes [Top+21]



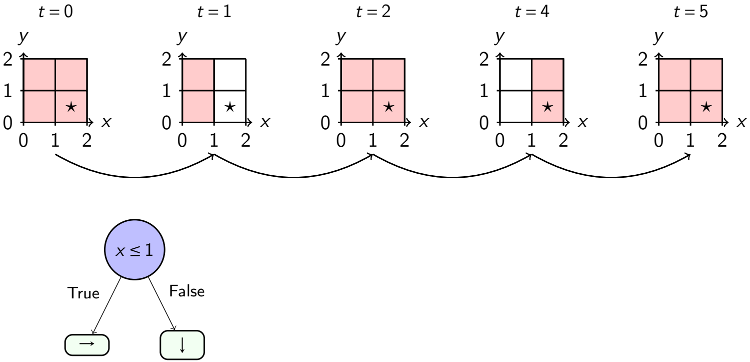
Iterative bounding Markov decision processes [Top+21]



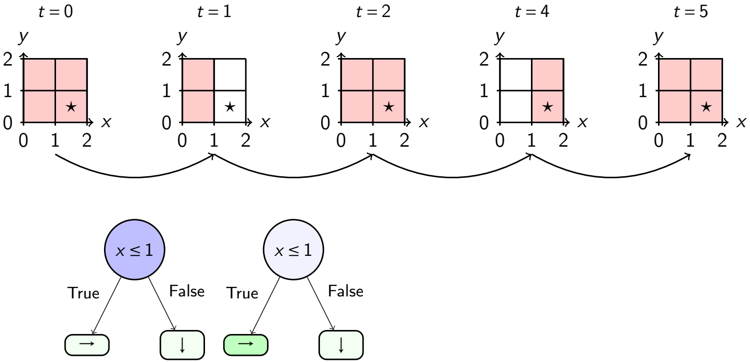
Iterative bounding Markov decision processes [Top+21]



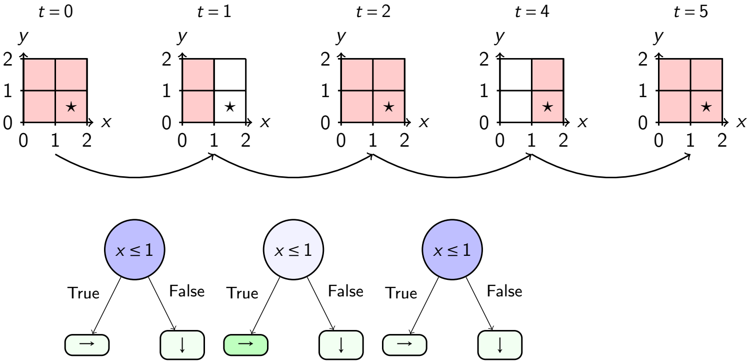
Iterative bounding Markov decision processes [Top+21]



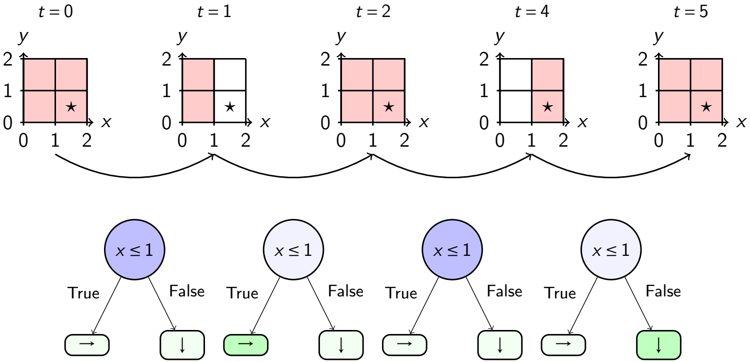
Iterative bounding Markov decision processes [Top+21]



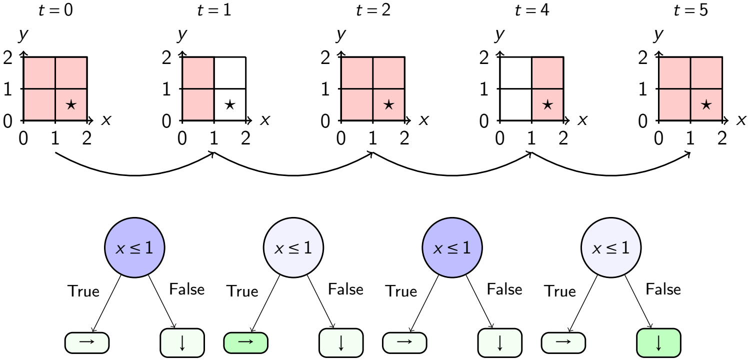
Iterative bounding Markov decision processes [Top+21]



Iterative bounding Markov decision processes [Top+21]



Iterative bounding Markov decision processes [Top+21]



⚠ **To learn decision tree policies for an MDP we need to hide and allow agents to query information about state features.**

Deterministic partially observable policies in IBMDPs

IBMDPs promises

- No need to design new algorithm: we can use RL.
- IBMDP rewards trade-off naturally interpretability and performances.

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LFM25].

Deterministic partially observable policies in IBMDPs

IBMDPs promises

- No need to design new algorithm: we can use RL.
- IBMDP rewards trade-off naturally interpretability and performances.

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LFM25].

Deterministic partially observable policies in IBMDPs

IBMDPs promises

- No need to design new algorithm: we can use RL.
- IBMDP rewards trade-off naturally interpretability and performances.

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LFM25].

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o,a)$ with TD targets $Q(s,a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o,a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

Deterministic partially observable policies in IBMDPs

RL for partially observable policies

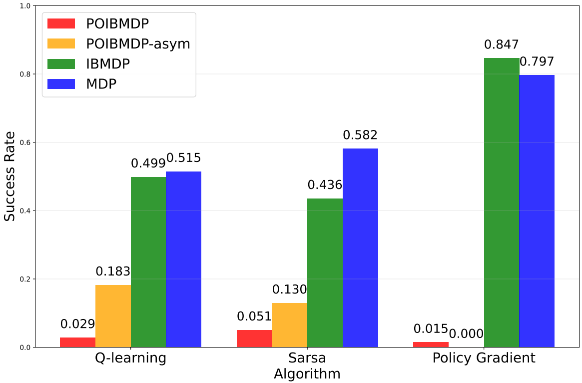
- Finding the best **deterministic** and **partially observable** policy is NP-hard [Lit94]!
- The best partially observable policy can be stochastic [SJJ94].
- Value-based RL converges to sub-optimal solutions [SJJ94].

Asymmetric RL

- Access to hidden states during training but not at execution [Pin+17].
- Value-based → learns $Q(o, a)$ with TD targets $Q(s, a)$ [BDA22].
- Actor-critic[a] → policy gradient on $\pi(o, a)$ using a critic $V(s)$ [BA22].
- Supposed to work better for our problem [LEM25].

[a] Although those return stochastic policies, we can be greedy.

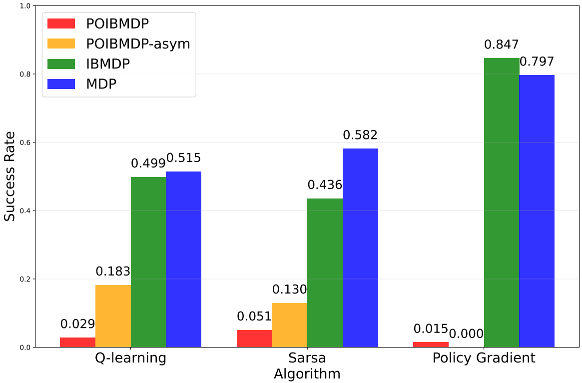
Result: for similar problems, RL struggles more when there is partial observability



Success rates over thousands of RL runs with varying hyperparameters when learning different policies in the same IBMDP[1]. Is it all for nothing?

[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]. **Is it all for nothing?**

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

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? A: Yes!

Perspectives for direct RL of decision tree policies.

- It seems that interpretability for SDM problems can be difficult to achieve because of **partial observability**.
- Should we focus on indirect approaches? Hybrid approaches [Wu+20]?
- Fixing the policy tree structure a priori (paramteric trees, [Mar+25])?
- Can other policies (programs, oblique trees, algebraic expressions. . .) be directly optimized with RL in IBMDPs?
- Design algorithms that learn deterministic partially observable policies [LBE25; LEM25]?

We show that decision tree policies for supervised learning tasks are fully observable IBMDP policies.

Q: Can we leverage SDM to design new decision tree induction algorithms for the supervised learning setting? **A: Yes!**

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** [GOV22].
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O(2^D)}$ operations [Bre+84; Qui86; Qui93] .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O((2Np)^D)}$ operations (NP-hard) [BD17; Dem+22; LWD23; CRB24; HR76].
- 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** [GOV22].
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O}(2^{\boldsymbol{D}})$ operations [Bre+84; Qui86; Qui93] .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O}((2\boldsymbol{Np})^{\boldsymbol{D}})$ operations (NP-hard) [BD17; Dem+22; LWD23; CRB24; HR76].
- 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** [GOV22].
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O(2^D)}$ operations [Bre+84; Qui86; Qui93] .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O((2Np)^D)}$ operations (NP-hard) [BD17; Dem+22; LWD23; CRB24; HR76].
- 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** [GOV22].
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O(2^D)}$ operations [Bre+84; Qui86; Qui93] .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O((2Np)^D)}$ operations (NP-hard) [BD17; Dem+22; LWD23; CRB24; HR76].
- 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** [GOV22].
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O(2^D)}$ operations [Bre+84; Qui86; Qui93] .
- Optimal algorithms, **optimal accuracy**, but $\boldsymbol{O((2Np)^D)}$ operations (NP-hard) [BD17; Dem+22; LWD23; CRB24; HR76].
- 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** [GOV22].
- Greedy algorithms **sub-optimal accuracy**, but $\boldsymbol{O(2^D)}$ operations [Bre+84; Qui86; Qui93] .
- Optimal algorithms, optimal accuracy, but $\boldsymbol{O((2Np)^D)}$ operations (NP-hard) [BD17; Dem+22; LWD23; CRB24; HR76].
- 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: penalty or accuracies.
- T: node traversals.

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: penalty or accuracies.
- T: node traversals.

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: penalty or accuracies.
- T: node traversals.

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: penalty or accuracies.
- T: node traversals.

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: penalty or accuracies.
- T: node traversals.

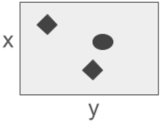
Decision tree induction as solving MDPs

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.

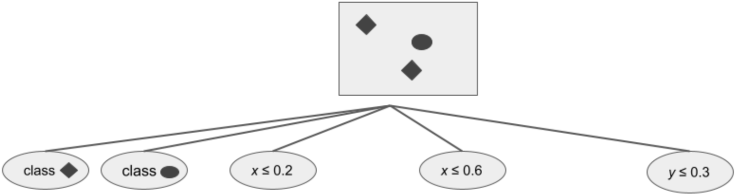
- S: data subsets.
- A: test or leaf nodes that can be added to the tree.
- R: penalty or accuracies.
- T: node traversals.

Decision tree induction as solving MDPs



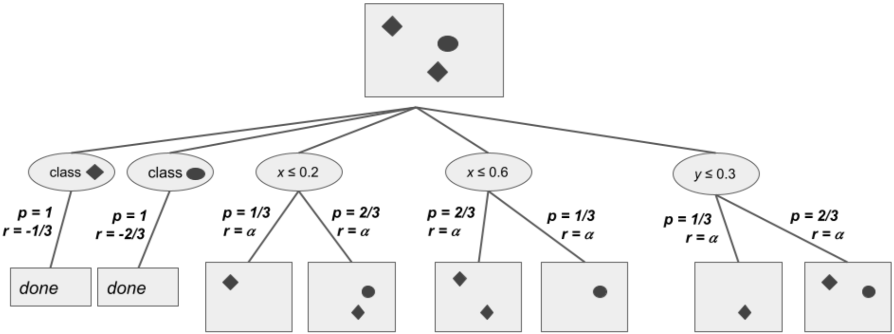
Example of decision tree induction as an MDP.

Decision tree induction as solving MDPs



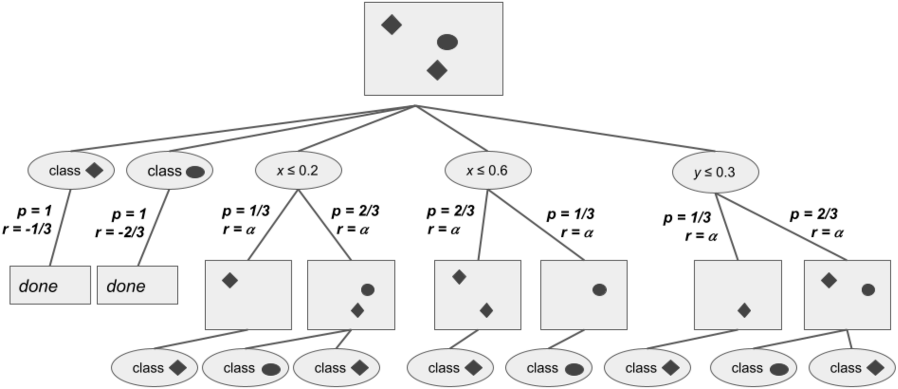
Example of decision tree induction as an MDP.

Decision tree induction as solving MDPs



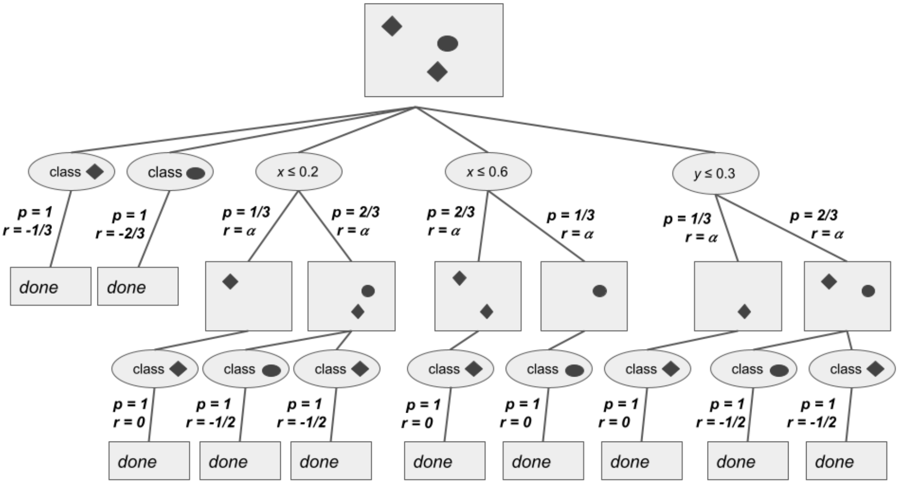
Example of decision tree induction as an MDP.

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.

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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$.**

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$.**

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$.**

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$.**

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: state space size is $O((2B)^D)$.

How to choose the B candidate actions/splits?

Top-B greedy splits [Bla+23], quantiles, random. . .

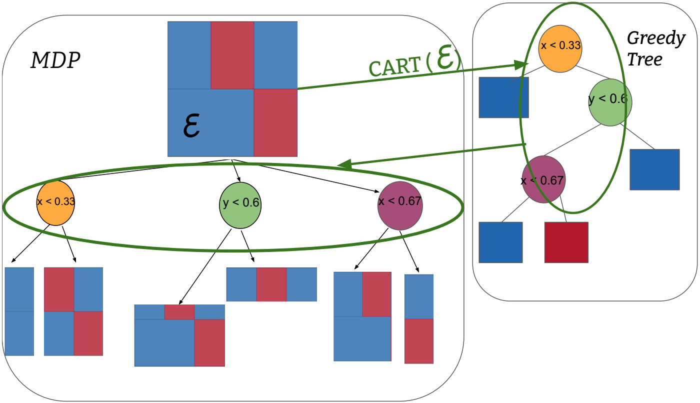
Controlling the time complexity of decision tree induction

- 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)**: Let's choose candidate actions adaptively
  → for each MDP state consider B actions: **state space size is $O((2B)^D)$**.

How to choose the B candidate actions/splits?

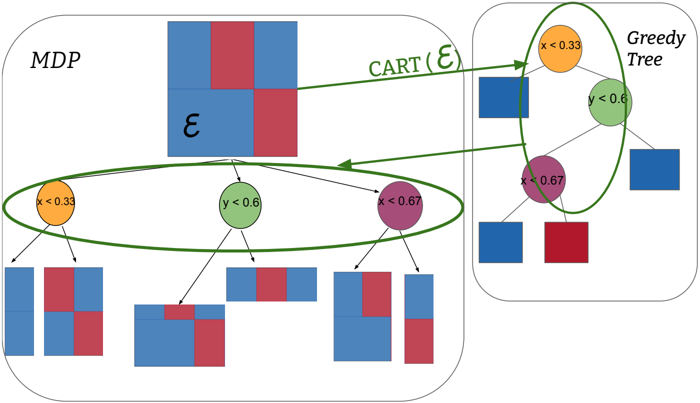
Top-B greedy splits [Bla+23], quantiles, random...

Practical implemenataion of DPDT



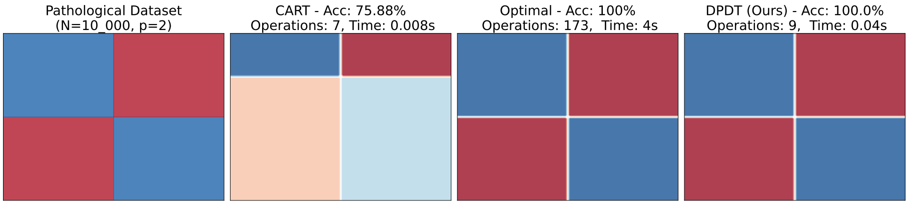
We can use greedy trees nodes as candidate actions.

Practical implemenataion of DPDT



We can use greedy trees nodes as candidate actions.

Fast like greedy trees, accurate like optimal trees



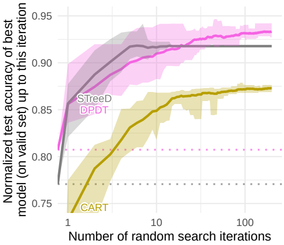
Comparison of greedy, optimal, and DPDT depth-2 trees on the checkersboard dataset.

Fast like greedy trees, accurate like optimal trees

Comparison of accuracies and operations for depth-3 trees.

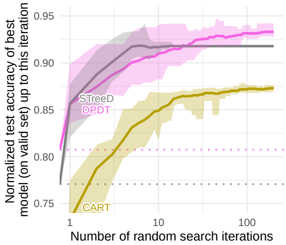
	Accuracy						Operations					
Dataset	Opt	Greedy	DPDT CART$^-$	DPDT CART$^+$	DPDT TopB$^-$	DPDT TopB$^+$	Opt	Greedy	DPDT CART$^-$	DPDT CART$^+$	DPDT TopB$^-$	DPDT TopB$^+$
room	**0.992**	0.968	0.991	**0.992**	0.990	**0.992**	10^6	15	286	16100	111	16100
bean	**0.871**	0.777	0.812	0.853	0.804	0.841	$5 \cdot 10^6$	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

DPDT trees generalization

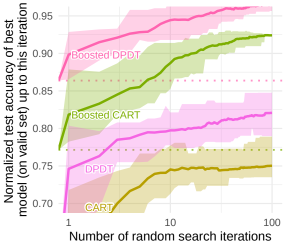


DPDT depth-5 trees vs. other detph-5 trees

DPDT trees generalization

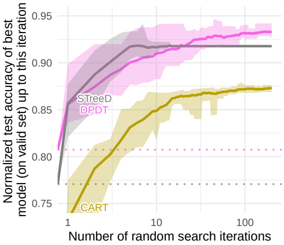


DPDT depth-5 trees vs. other detph-5 trees

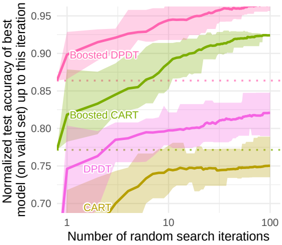


Boosted DPDT vs. Boosted CART

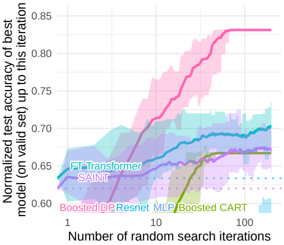
DPDT trees generalization



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



Boosted DPDT vs. Boosted CART



Boosted DPDT vs. other classifiers

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.

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.

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.

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.

Why generating candidate splits with CART?

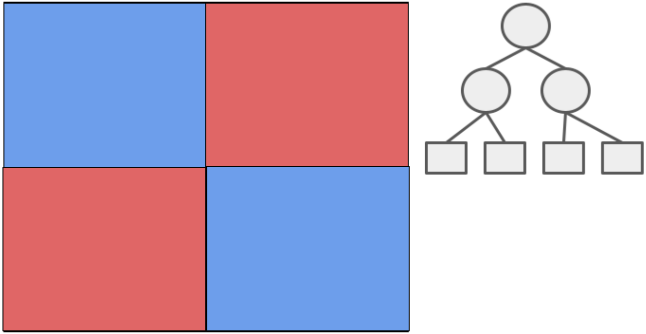
Theorem (DPDT trees are not worse than greedy trees)

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.

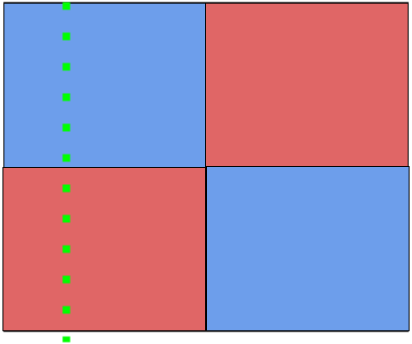
Theorem (DPDT trees can be strictly better than greedy trees)

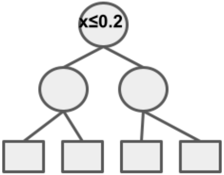
There exist a depth budget D and a dataset for which DPDT trees are strictly better than greedy trees.

DPDT trees can be strictly better than greedy trees

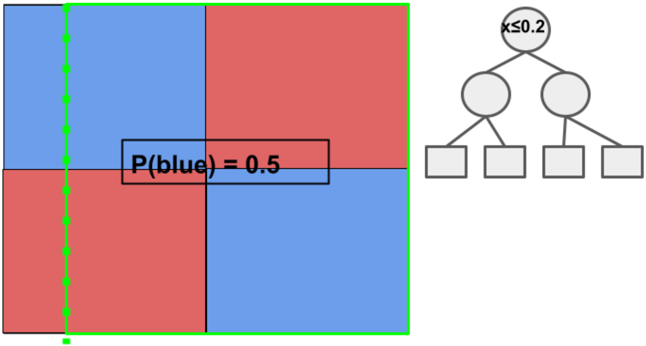


DPDT trees can be strictly better than greedy trees

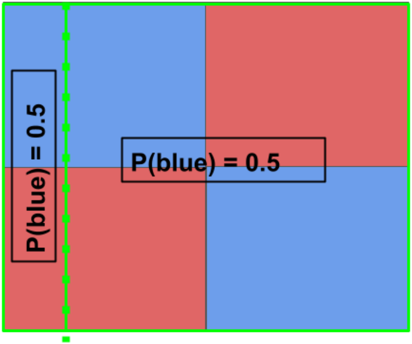


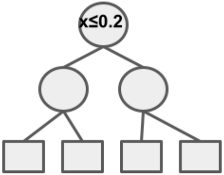


DPDT trees can be strictly better than greedy trees

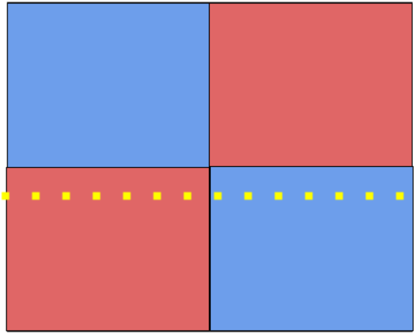


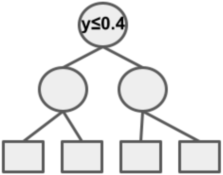
DPDT trees can be strictly better than greedy trees



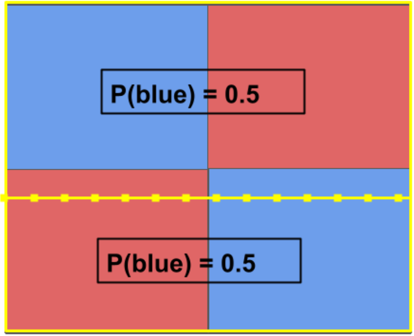


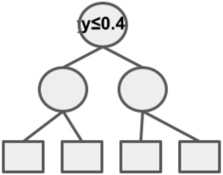
DPDT trees can be strictly better than greedy trees



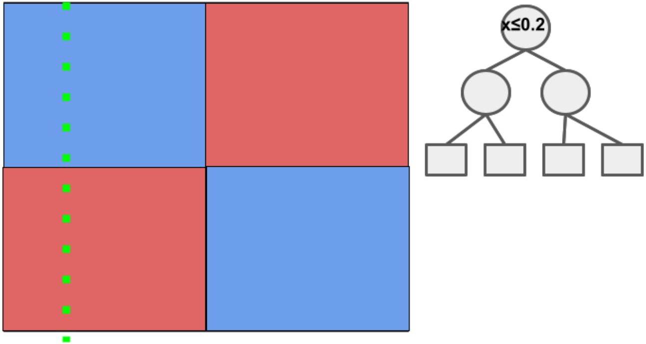


DPDT trees can be strictly better than greedy trees

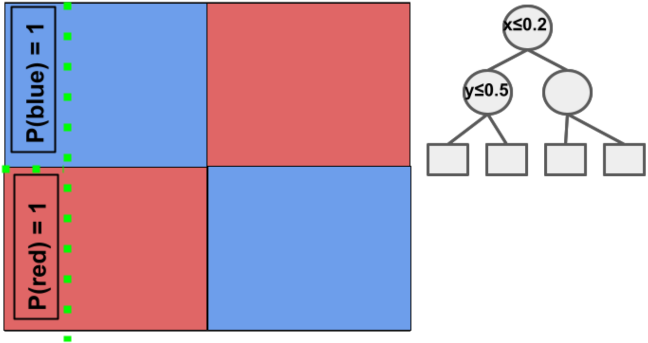




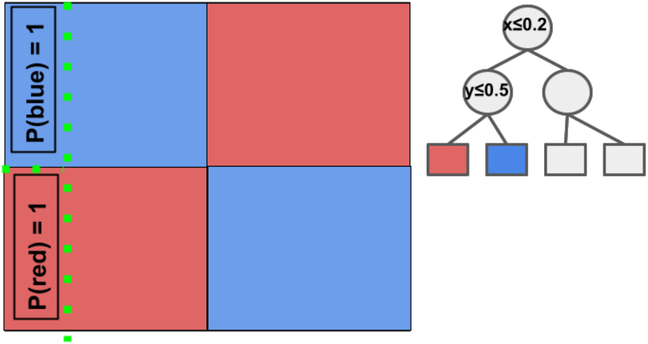
DPDT trees can be strictly better than greedy trees



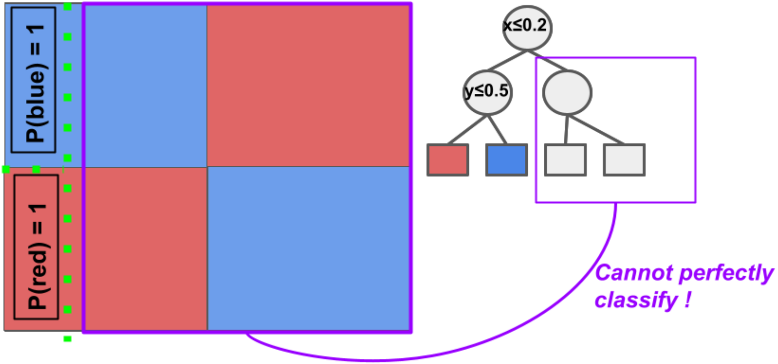
DPDT trees can be strictly better than greedy trees



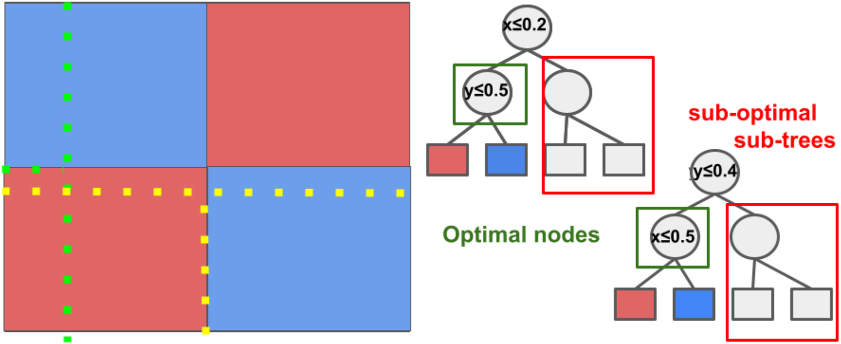
DPDT trees can be strictly better than greedy trees



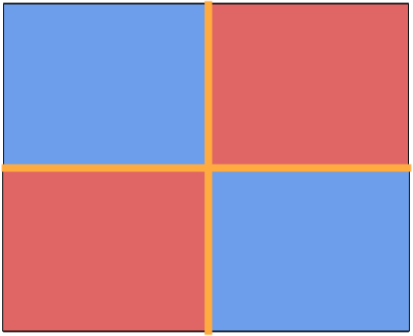
DPDT trees can be strictly better than greedy trees

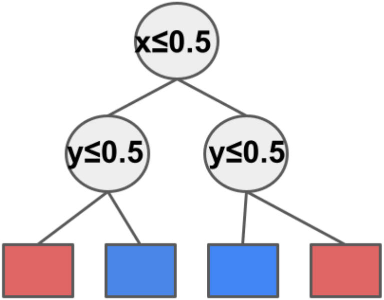


DPDT trees can be strictly better than greedy trees



DPDT trees can be strictly better than greedy trees





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?

A: It depends.

- 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?

A: It depends.

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?

A: It depends.

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?

A: It depends.

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?

A: It depends.

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?
A: It depends.

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability ≃ time for a human to compute the same.
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability ≃ time for a human to compute the same.
- Interpretability ≃ how much effort for a human to read through the entire policy.
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability ≃ time for a human to compute the same.
- Interpretability ≃ how much effort for a human to read through the entire policy.
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability ≃ time for a human to compute the same.
- Interpretability ≃ how much effort for a human to read through the entire policy.
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability ≃ **runtime in seconds?**
- Interpretability ≃ how much effort for a human to read through the entire policy.
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability $\simeq$ **runtime in seconds?**
- Interpretability $\simeq$ **size in bytes?**
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability ≃ **runtime in seconds?**
- Interpretability ≃ **size in bytes?**
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

How to measure policy interpretability?

Challenges [Gla+24; Lip18; DK17]

- No definition of interpretability.
- Measuring might require humans.
- Different hardwares (CPUs vs GPUs).
- Different implementations (matrix operations vs fully sequentially) [Luo+24]

The notion of *simulatability* [Lip18]

- Interpretability $\simeq$ **runtime in seconds?**
- Interpretability $\simeq$ **size in bytes?**
- Less parameters mean more interpretability [Fre14; Lav99].
- Time to formally verify a policy decreases with interpretability [Bar+20].

We propose policy unfolding

```
# 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
```

```
# 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]
```

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]
```

We propose policy unfolding

```
# 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
```

```
# 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]
```

Empirical validation

1. Is policy unfolding necessary?
2. What kind of results we can obtain using our proposed methodology?

Setup

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

Empirical validation

1. Is policy unfolding necessary?
2. What kind of results we can obtain using our proposed methodology?

Setup

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

Empirical validation

1. Is policy unfolding necessary?
2. What kind of results we can obtain using our proposed methodology?

Setup

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

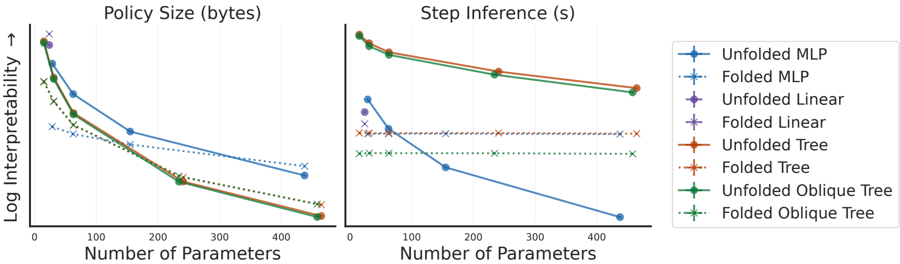
Empirical validation

1. Is policy unfolding necessary?
2. What kind of results we can obtain using our proposed methodology?

Setup

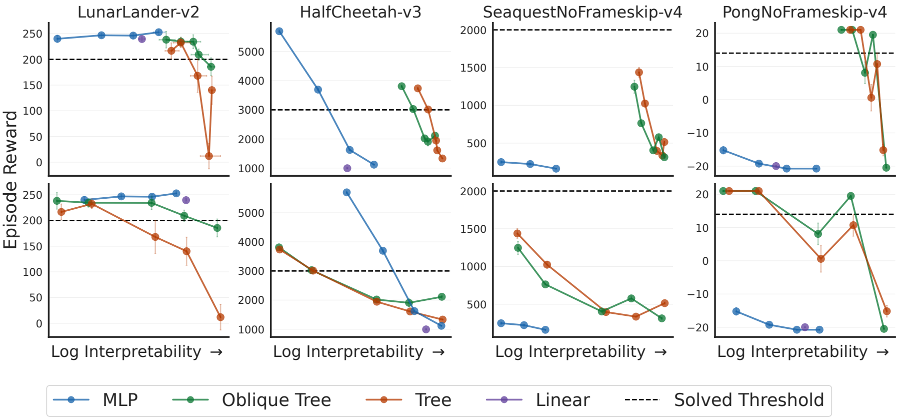
We imitate ~ 40000 expert policies from `stable-baselines3` using various policy classes/nb parameters on various environments.

Result: unfolding policies is necessary to respect consensus



Aggregated policies interpretability on classic control environments

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.

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- Tree-like policy classes can have good inductive bias (e.g. Atari).
- What about (very) big models?
- Can we use our policy programs as low level skills (hierarchical RL)?

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- Tree-like policy classes can have good inductive bias (e.g. Atari).
- What about (very) big models?
- Can we use our policy programs as low level skills (hierarchical RL)?

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- Tree-like policy classes can have good inductive bias (e.g. Atari).
- What about (very) big models?
- Can we use our policy programs as low level skills (hierarchical RL)?

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- Tree-like policy classes can have good inductive bias (e.g. Atari).
- What about (very) big models?
- Can we use our policy programs as low level skills (hierarchical RL)?

Perspectives

- Beliefs such as "trees are more interpretable than neural networks" should be used with caution.
- Tree-like policy classes can have good inductive bias (e.g. Atari).
- What about (very) big models?
- Can we use our policy programs as low level skills (hierarchical RL)?

Conclusion: interpretable SDM is a difficult research topic

- Technical challenges: **partial observability in SDM, NP-hardness**.
  → Focus on indirect approaches and/or on POMDP research first.
- Fundamental challenges: **no definition**.
  → Discuss with the community (InterpPol workshop).
- **Decision trees offer good inductive bias for SDM in games or tabular data**.

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary [Nag+24].

Conclusion: interpretable SDM is a difficult research topic

- Technical challenges: **partial observability in SDM, NP-hardness**.
  → Focus on indirect approaches and/or on POMDP research first.
- Fundamental challenges: **no definition**.
  → Discuss with the community (InterpPol workshop).
- **Decision trees offer good inductive bias for SDM in games or tabular data**.

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary [Nag+24].

Conclusion: interpretable SDM is a difficult research topic

- Technical challenges: **partial observability in SDM, NP-hardness**.
  → Focus on indirect approaches and/or on POMDP research first.
- Fundamental challenges: **no definition**.
  → Discuss with the community (InterpPol workshop).
- **Decision trees offer good inductive bias for SDM in games or tabular data.**

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary [Nag+24].

Conclusion: interpretable SDM is a difficult research topic

- Technical challenges: **partial observability in SDM, NP-hardness**.
  → Focus on indirect approaches and/or on POMDP research first.
- Fundamental challenges: **no definition**.
  → Discuss with the community (InterpPol workshop).
- **Decision trees offer good inductive bias for SDM in games or tabular data.**

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary [Nag+24].

Conclusion: interpretable SDM is a difficult research topic

- Technical challenges: **partial observability in SDM, NP-hardness**.
  → Focus on indirect approaches and/or on POMDP research first.
- Fundamental challenges: **no definition**.
  → Discuss with the community (InterpPol workshop).
- **Decision trees offer good inductive bias for SDM in games or tabular data.**

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary [Nag+24].

Conclusion: interpretable SDM is a difficult research topic

- Technical challenges: **partial observability in SDM, NP-hardness**.
  → Focus on indirect approaches and/or on POMDP research first.
- Fundamental challenges: **no definition**.
  → Discuss with the community (InterpPol workshop).
- **Decision trees offer good inductive bias for SDM in games or tabular data**.

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary [Nag+24].

Conclusion: interpretable SDM is a difficult research topic

- Technical challenges: **partial observability in SDM, NP-hardness**.
  → Focus on indirect approaches and/or on POMDP research first.
- Fundamental challenges: **no definition**.
  → Discuss with the community (InterpPol workshop).
- **Decision trees offer good inductive bias for SDM in games or tabular data**.

My hope

Motivate interpretability by finding a real-world problem where interpretability is *really* necessary [Nag+24].

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?
- **Teaching:** Can we use unfolded policies (and interpretability) for teaching?

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?
- **Teaching:** Can we use unfolded policies (and interpretability) for teaching?

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?
- **Teaching:** Can we use unfolded policies (and interpretability) for teaching?

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?
- **Teaching:** Can we use unfolded policies (and interpretability) for teaching?

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?
- **Teaching:** Can we use unfolded policies (and interpretability) for teaching?

[BA22] Andrea Baisero and Christopher Amato. "Unbiased Asymmetric Reinforcement Learning under Partial Observability". In: *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '22. Virtual Event, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, 2022, pp. 44–52. ISBN: 9781450392136.

[Bar+20] Pablo Barceló et al. "Model interpretability through the lens of computational complexity". In: *Advances in neural information processing systems* (2020).

[BD17] Dimitris Bertsimas and Jack Dunn. "Optimal classification trees". In: *Machine Learning* 106 (2017), pp. 1039–1082.

[BDA22] Andrea Baisero, Brett Daley, and Christopher Amato. "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, Jan. 2022, pp. 107–117. URL: https://proceedings.mlr.press/v180/baisero22a.html.

[Bla+23] Guy Blanc et al. "Harnessing the power of choices in decision tree learning". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 80220–80232.

[BPS18] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. "Verifiable Reinforcement Learning via Policy Extraction". In: (2018).

[Bre+84] L Breiman et al. *Classification and Regression Trees*. Wadsworth, 1984.

[CRB24] Ayman Chaouki, Jesse Read, and Albert Bifet. "Branches: A Fast Dynamic Programming and Branch & Bound algorithm for Optimal Decision Trees". In: (2024). arXiv: 2406.02175 [cs.LG]. URL: https://arxiv.org/abs/2406.02175.

[Dem+22] Emir Demirovic et al. "MurTree: Optimal Decision Trees via Dynamic Programming and Search". In: *Journal of Machine Learning Research* 23.26 (2022), pp. 1–47. URL: http://jmlr.org/papers/v23/20-520.html.

[DK17] Finale Doshi-Velez and Been Kim. "Towards A Rigorous Science of Interpretable Machine Learning". In: (2017). arXiv: 1702.08608 [stat.ML]. URL: https://arxiv.org/abs/1702.08608.

[Fre14] Alex A. Freitas. "Comprehensible classification models: a position paper". In: *SIGKDD Explor. Newsl.* 15.1 (Mar. 2014), pp. 1–10. ISSN: 1931-0145. DOI: 10.1145/2594473.2594475. URL: https://doi.org/10.1145/2594473.2594475.

[Gla+24] Claire Glanois et al. "A survey on interpretable reinforcement learning". In: *Machine Learning* (2024), pp. 1–44.

[GOV22] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. "Why do tree-based models still outperform deep learning on typical tabular data?" In: *Advances in neural information processing systems* 35 (2022), pp. 507–520.

[HR76] Laurent Hyafil and Ronald L. Rivest. "Constructing optimal binary decision trees is NP-complete". In: *Information Processing Letters* 5.1 (1976), pp. 15–17. ISSN: 0020-0190.

DOI: https://doi.org/10.1016/0020-0190(76)90095-8. URL: https://www.sciencedirect.com/science/article/pii/0020019076900958.

[Lav99] Nada Lavrač. "Selected techniques for data mining in medicine". In: *Artificial Intelligence in Medicine* 16.1 (1999). Data Mining Techniques and Applications in Medicine, pp. 3–23. ISSN: 0933-3657. DOI: https://doi.org/10.1016/S0933-3657(98)00062-1. URL: https://www.sciencedirect.com/science/article/pii/S0933365798000621.

[LBE25] Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. "Informed POMDP: Leveraging Additional Information in Model-Based RL". In: *Reinforcement Learning Journal* 2 (2025), pp. 763–784.

[LEM25] Gaspard Lambrechts, Damien Ernst, and Aditya Mahajan. "A Theoretical Justification for Asymmetric Actor-Critic algorithms". In: *Forty-second International Conference on*

Machine Learning. 2025. URL: https://openreview.net/forum?id=F1yANMCnAn.

[Lip18] Zachary C. Lipton. "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery.". In: *Queue* 16.3 (2018), pp. 31–57.

[Lit94] Michael L. Littman. "Memoryless policies: theoretical limitations and practical results". In: *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3: From Animals to Animats 3*. SAB94. Brighton, United Kingdom: MIT Press, 1994, pp. 238–245. ISBN: 0262531224.

[Luo+24] Lirui Luo et al. "End-to-End Neuro-Symbolic Reinforcement Learning with Textual Explanations". In: *International Conference on Machine Learning (ICML)* (2024).

[LWD23] Jacobus van der Linden, Mathijs de Weerdt, and Emir Demirović. "Necessary and Sufficient Conditions for Optimal Decision Trees using Dynamic Programming". In:

Advances in Neural Information Processing Systems 36 (2023). Ed. by A. Oh et al., pp. 9173–9212.

[Mar+25] Sascha Marton et al. "Mitigating Information Loss in Tree-Based Reinforcement Learning via Direct Optimization". In: (2025). URL: https://openreview.net/forum?id=qpXctF2aLZ.

[Mil+24] Stephanie Milani et al. "Explainable Reinforcement Learning: A Survey and Comparative Review". In: *ACM Comput. Surv.* 56.7 (Apr. 2024). ISSN: 0360-0300. DOI: 10.1145/3616864. URL: https://doi.org/10.1145/3616864.

[MMW22] Rahul Mazumder, Xiang Meng, and Haoyue Wang. "Quant-BnB: A Scalable Branch-and-Bound Method for Optimal Decision Trees with Continuous Features". In: *Proceedings of the 39th International Conference on Machine Learning*. Proceedings of Machine Learning Research 162 (17–23 Jul 2022). Ed. by Kamalika Chaudhuri et al., pp. 15255–15277. URL: https://proceedings.mlr.press/v162/mazumder22a.html.

[Mni+15] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[Nag+24] Myura Nagendran et al. "Eye tracking insights into physician behaviour with safe and unsafe explainable AI recommendations". In: *NPJ Digital Medicine* 7.1 (2024), p. 202.

[Pin+17] Lerrel Pinto et al. *Asymmetric Actor Critic for Image-Based Robot Learning*. 2017. arXiv: 1710.06542 [cs.RO]. URL: https://arxiv.org/abs/1710.06542.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[Qui86] J. R. Quinlan. "Induction of Decision Trees". In: *Mach. Learn.* 1.1 (1986), pp. 81–106.

[Qui93] J Ross Quinlan. "C4. 5: Programs for machine learning". In: *Morgan Kaufmann google schola* 2 (1993), pp. 203–228.

[RGB10] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: (2010).

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press, 1998.

[Sch+17] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[SJJ94] Satinder P. Singh, Tommi S. Jaakkola, and Michael I. Jordan. "Learning without state-estimation in partially observable Markovian decision processes". In: *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*. ICML'94. New Brunswick, NJ, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 284–292. ISBN: 1558603352.

[Top+21] Nicholay Topin et al. "Iterative bounding mdps: Learning interpretable policies via non-interpretable methods". In:

Proceedings of the AAAI Conference on Artificial Intelligence 35 (2021), pp. 9923–9931.

[Ver+18] Abhinav Verma et al. "Programmatically interpretable reinforcement learning". In: (2018), pp. 5045–5054.

[VZ19] Sicco Verwer and Yingqian Zhang. "Learning optimal classification trees using a binary linear program formulation". In: *Proceedings of the AAAI conference on artificial intelligence* 33 (2019), pp. 1625–1632.

[Wu+20] Mike Wu et al. "Regional Tree Regularization for Interpretability in Deep Neural Networks". In: 34 (Apr. 2020), pp. 6413–6421. DOI: 10.1609/aaai.v34i04.6112. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6112.