Interpretability, Decision Trees, and Sequential Decision Making

Hector Kohler

Université de Lille, CNRS, Inria, UMR CRIStAL 9189, France

November 17, 2025

Sequential decision making (SDM)

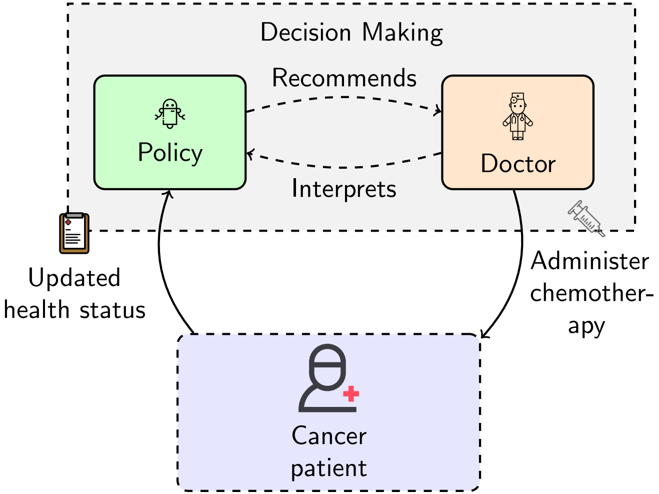


Figure: Sequential decision making in cancer treatment.

Machine learning (ML) of policies for SDM

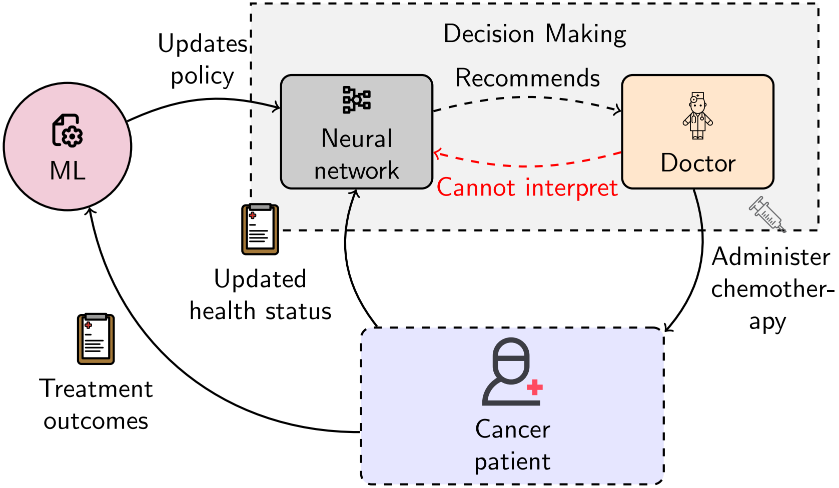


Figure: Machine learning of neural networks has many recent successes but neural networks are black-box.

Research direction

How to learn interpretable policies for sequential decision making?

Interpretable policies

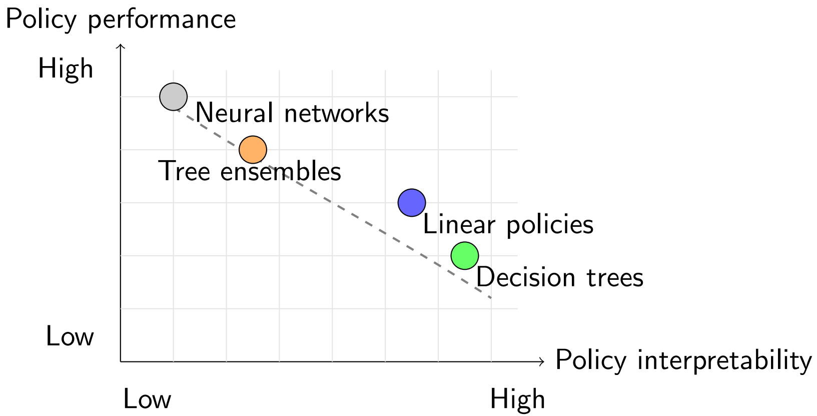


Figure: **Heuristic** interpretability-performance trade-offs of different policy classes. Interpretability is often presented in opposition to performances.

Decision trees

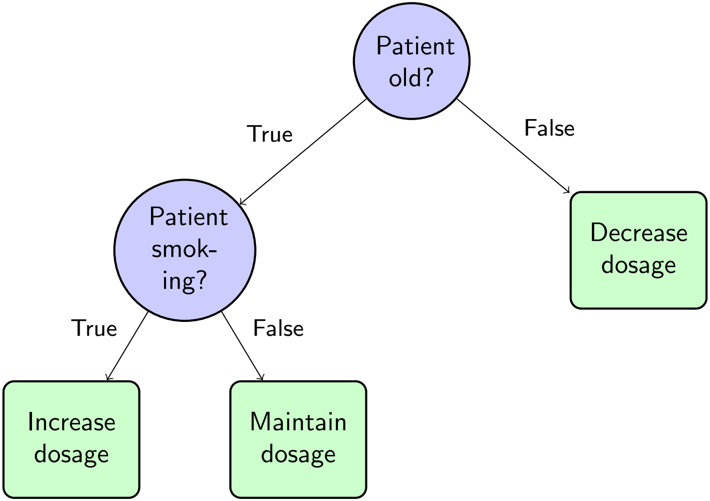


Figure: A generic decision tree of depth $D = 2$. Classification And Regression Trees (CART, 1984), Optimal Classification Trees (2015). What about sequential decision making?

Markov decision processes

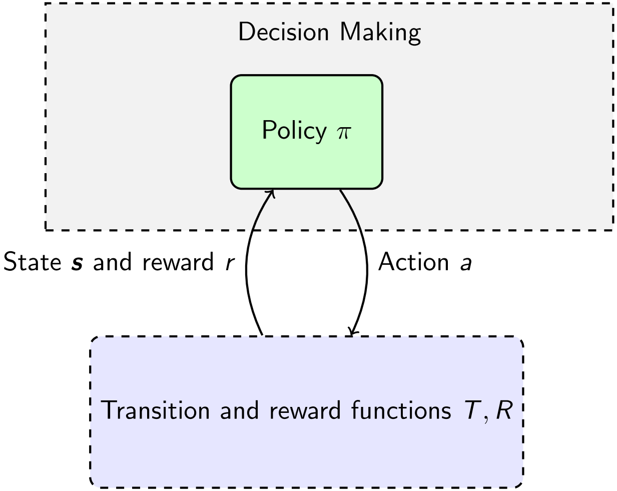

Decision Making
Policy π
State $\boldsymbol{s}$ and reward r
Action a
Transition and reward functions T, R

Figure: Markov decision process

Reinforcement learning (RL) objective

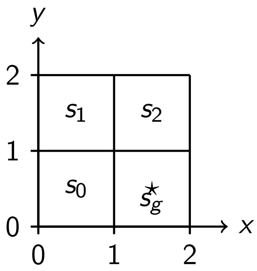
- Given an MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, the goal of reinforcement learning for sequential decision making is to find a model, also known as a policy, $\pi : S \to A$ that maximizes the expected discounted sum of rewards:

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

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

- Value iteration, Q-learning, Sarsa, Deep Q Networks, PPO, $\cdot$

Grid world MDP



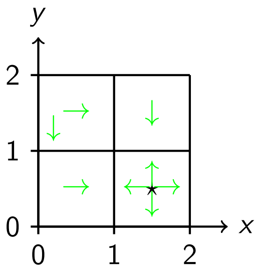
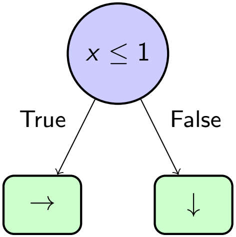


Figure: A grid world MDP and optimal actions w.r.t. the RL objective.

Example: a decision tree policy for the grid world MDP



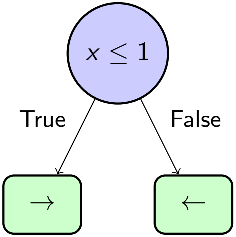


Figure: Left, an optimal depth-1 decision tree policy. On the right, a sub-optimal depth-1 decision tree policy.

Two ways to get interpretable policies for SDM

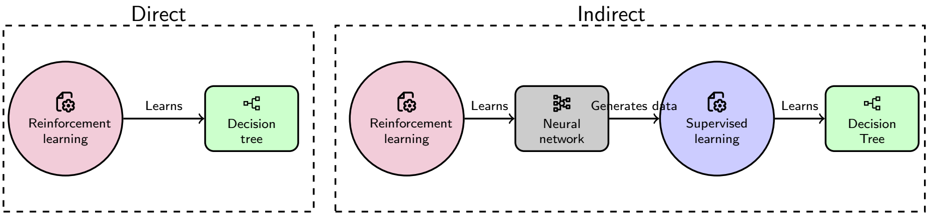


Figure: Comparison of direct and indirect approaches for learning interpretable policies in sequential decision making.

Indirect approach: imitation learning

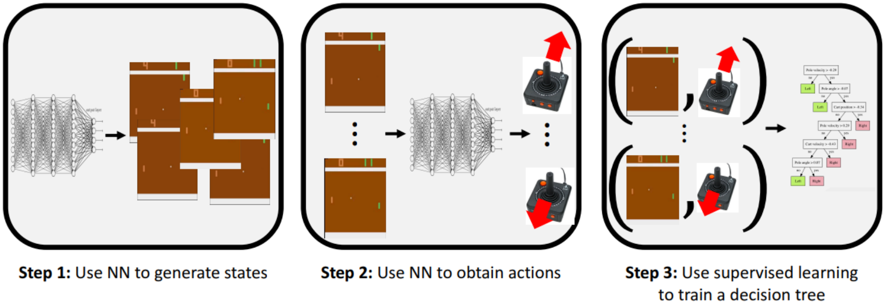
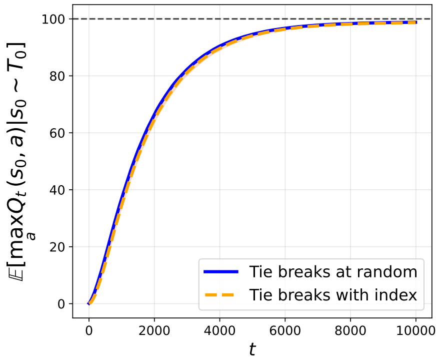

Step 1: Use NN to generate states
Step 2: Use NN to obtain actions
Step 3: Use supervised learning to train a decision tree

Figure: Imitation learning to get interpretable policies (DAgger, VIPER) [**viper**, **dagger**] works well in practice but no optimality guarantees.

Example: a decision tree policy for the grid world MDP



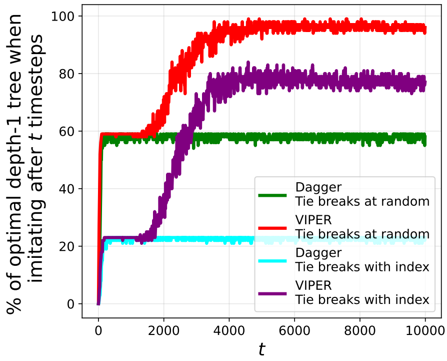


Figure: Left, sample complexity curve of Q-learning with default hyperparameters on the 2×2 grid world MDP over 100 random seeds. Right, performance of indirect interpretable methods when imitating the greedy policy with a tree at different Q-learning stages.

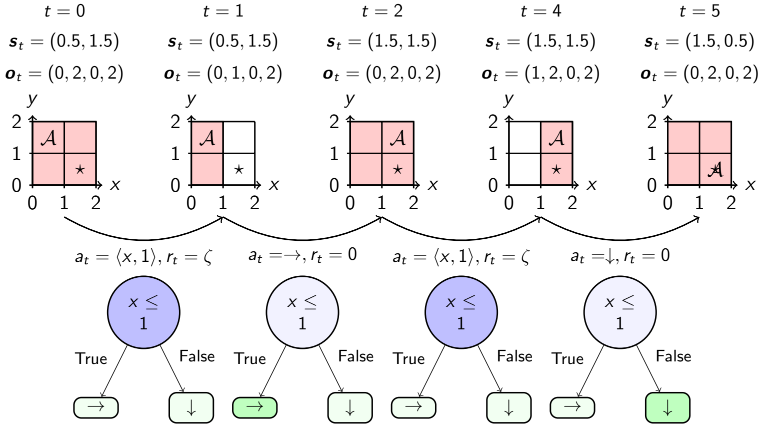
Direct approach to interpretable SDM

- Policy class: non-parametric trees[**topin2021iterative**].
- Optimize a trade-off of policy perofrmances $J(\pi)$ and interpretability.

Q: Can we use reinforcement learning to directly optimize trade-offs of performance and interpretability in SDM?

A: direct reinforcement leargning is hard because it involves partial observability.

Iterative bounding Markov decision processes (IBMDP)



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

Figure: Topin et. al. propose iterative bounding Markov decision processes (IBMDP). Trajectories in IBMDPs corresponding to decision tree policy building. Not all IBMDP policies are equivalent to decision tree policies

Pros and cons of IBMDPs

Pros

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

Cons

- Only **determinstic** and **partially observable** (a.k.a. memoryless or reactive) policies are equivalent to decision tree policies.
- Finding the best **deterministic** and **partially observable** policy is NP-hard [**littman**]!!

Re-formulation

Q: Can we use reinforcement learning to directly optimize trade-offs of performance and interpretability in SDM? $\equiv$
Q: How does RL perform for optimizing **deterministic** *and* **partially observable** *policies in IBMDPs?*

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

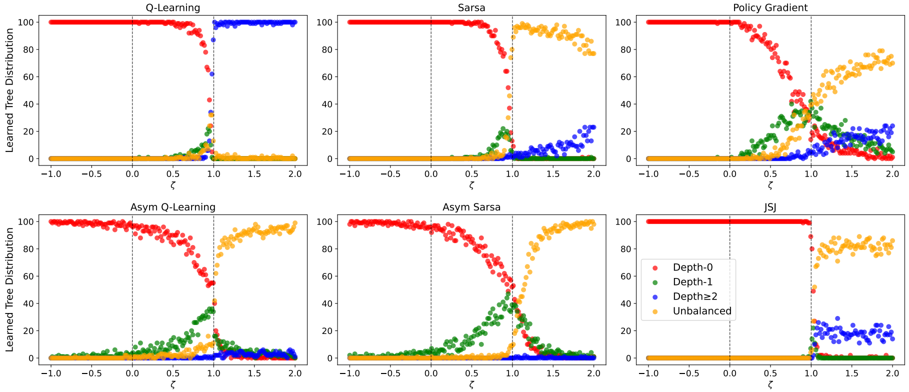


Figure: Distributions of final tree policies learned across the 100 seeds. For each ζ value, there are four colored points. Each point represent the share of depth-0 trees (red), depth-1 trees (green), unbalanced depth-2 trees (orange) and depth-2 trees (blue).

Experimental result: for similar problems, RL succeeds more often in retrieving optimal Markovian policies than partially observable policies

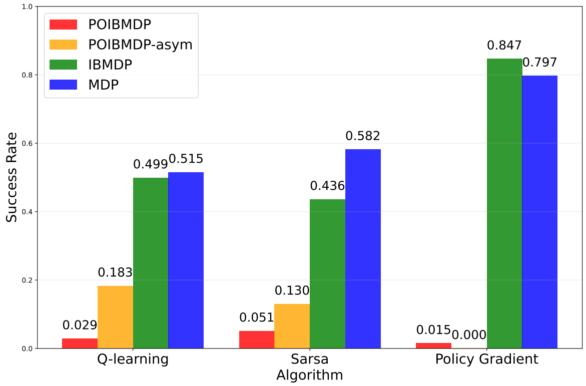
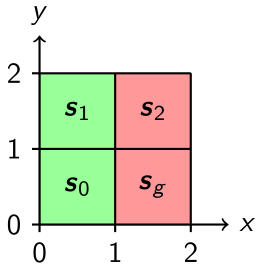


Figure: Success rates of different (asymmetric) RL algorithms over thousands of runs when applied to learning deterministic partially observable policies in an IBMDP for the grid world or learning deterministic Markovian policies in the same

Interesting sub-class of MDPs: classification MDPs



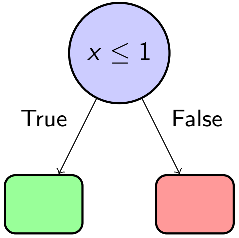


Figure: In this classification MDP, there are four data to which to assign either a green or red label. On the right, there is the unique optimal depth-1 tree for this particular classification MDP. This depth-1 tree also maximizes the accuracy on the corresponding classification task.

We show that in theory, deterministic partially observable policies for classification IBMDPs ($\equiv$ decision tree policies) are in fact Markovian.

Perspectives for direct RL of decision tree policies.

- Learning decision tree policies that trade-off performances and interpretability for SDM problems is difficult because for most problems there is partial observability.
- Still a lot of open questions:
  1. What are the pros and cons of fixing the policy tree structure a priori (paramteric trees, [**sympol**])?
  2. Can we specifically design algorithms that learn deterministic partially observable policies[**Lambrechts**]?

Q: Can we leverage SDM in classification MDPs to design new decision tree induction algorithms for the supervised learning (no sequentiality) setting? **A: Yes!**

Supervised learning

We assume that we have access to a set of N examples denoted $\mathcal{E} = \{(x_i, y_i)\}_{i=1}^N$. Each datum x_i is described by a set of p features. $y_i \in \mathcal{Y}$ is the label associated with x_i.

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i)) + \alpha C(f), \tag{1}$$

where $C : \mathcal{F} \to \mathbb{R}$ is a penalty for regularization

Decision trees in supervised learning

1. Decision trees are **interpretable**.
2. Tree-based models perform really well on **tabular** data, often **better than deep neural nets** (L. Grinsztajn et. al. 2022).

Optimal decision tree induction is NP-hard

- Greedy algorithms (C4.5, CART, ID3, ...) sub-optimal accuracy, but time complexity in $O(2^D)$.
- Optimal algorithms (MurTree, OCT, STreeD, Branches (Jesse Read's work ;)), ...) optimal accuracy, but time complexity in $O((2Np)^D)$.

In between?

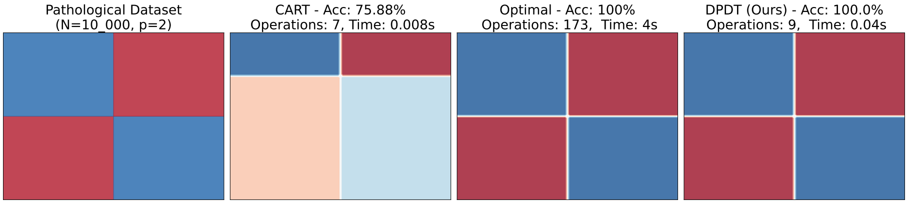


Figure: A checkers board data set highlights the limitations of existing works.

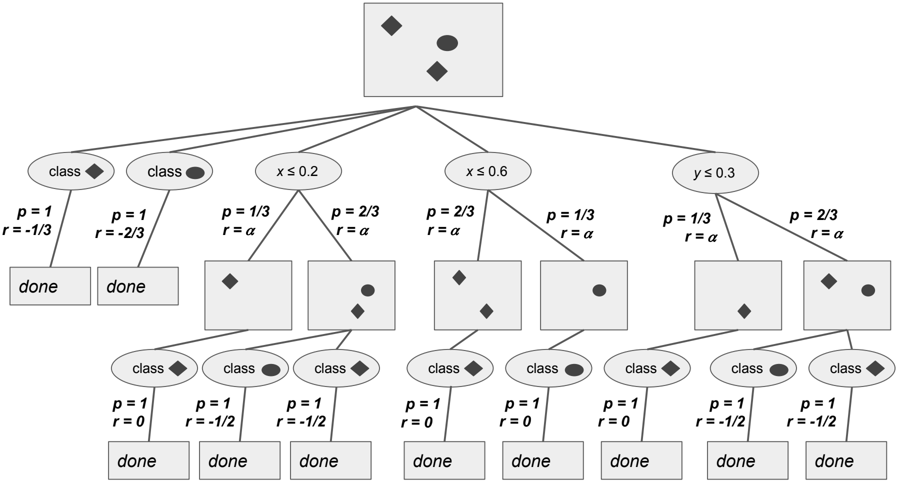
Decision tree induction as solving MDPs

Intuition

Given a set of examples $\mathcal{E}$, 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) $\mathcal{E}$, or to create a leaf node.

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

Decision tree induction as solving MDPs



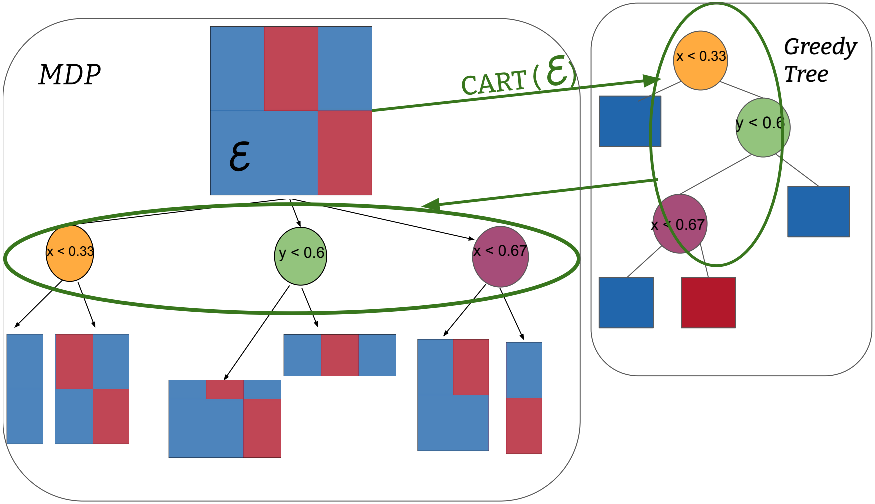
Controlling the time complexity of decision tree induction

- Greedy algorithms consider only one candidate action in each state which is the test that minimizes some impurity criterion $\rightarrow$ MDP state space size is $O(2^D)$.
- Optimal algorithms consider all possible actions in each state $\rightarrow$ MDP state space size is $O((2Np)^D)$.

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 $\rightarrow$ MDP state space size is $O(2^D)$.
- Optimal algorithms consider all possible actions in each state $\rightarrow$ MDP state space size is $O((2Np)^D)$.
- Let's choose candidate actions adaptively $\rightarrow$ for each MDP state consider B actions: state space size is $O((2B)^D)$.

Dynamic Programming Decision Trees (DPDT)[1]

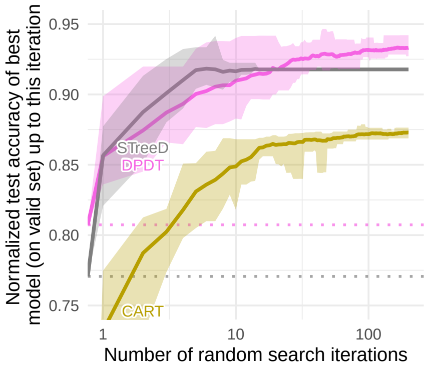


[1]Because states are entire datasets, we implement DPDT with a depth-first search to limit the space complexity.

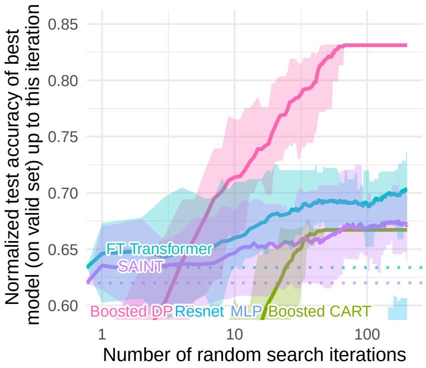
Comparing trees accuracy versus complexity

			Accuracy				Operations			
			Opt	Greedy	DPDT		Opt	Greedy	DPDT	
Dataset	N	p	Quant-BnB	CART	light	full	Quant-BnB	CART	light	full
room	8103	16	**0.992**	0.968	0.991	**0.992**	10^6	15	286	16100
bean	10888	16	**0.871**	0.777	0.812	0.853	$5{\cdot}10^6$	15	295	25900
eeg	11984	14	**0.708**	0.666	0.689	0.706	$2{\cdot}10^6$	13	289	26000
avila	10430	10	**0.585**	0.532	0.574	**0.585**	$3{\cdot}10^7$	9	268	24700
magic	15216	10	**0.831**	0.801	0.822	0.828	$6{\cdot}10^6$	15	298	28000
htru	14318	8	**0.981**	0.979	0.979	0.980	$6{\cdot}10^7$	15	295	25300
occup.	8143	5	**0.994**	0.989	0.991	**0.994**	$7{\cdot}10^5$	13	280	16300
skin	196045	3	**0.969**	0.966	0.966	0.966	$7{\cdot}10^4$	15	301	23300
fault	1552	27	**0.682**	0.553	0.672	0.674	$9{\cdot}10^8$	13	295	24200
segment	1848	18	**0.887**	0.574	0.812	0.879	$2{\cdot}10^6$	7	220	16300
page	4378	10	**0.971**	0.964	0.970	0.970	10^7	15	298	22400
bidding	5056	9	**0.993**	0.981	0.985	**0.993**	$3{\cdot}10^5$	13	256	9360
raisin	720	7	**0.894**	0.869	0.879	0.886	$4{\cdot}10^6$	15	295	20900
rice	3048	7	**0.938**	0.933	0.934	0.937	$2{\cdot}10^7$	15	298	25500
wilt	4339	5	**0.996**	0.993	0.994	0.995	$3{\cdot}10^5$	13	274	11300
bank	1097	4	**0.983**	0.933	0.971	0.980	$6{\cdot}10^4$	13	271	7990

DPDT trees generalization



(a) DPDT trees vs. other trees



(b) Boosted DPDT vs. other classifiers

Perspectives

- Great we did a new SOTA decision tree induction.

Q: Are decision trees really the most interpretable model? **A: It depends.**

How to measure policy interpretability?

Challenges [glanois-survey, lipton, rigourous]

- There is no clear definition of interpretability.
- Measuring interpretability might require humans.

How to measure policy interpretability?

Consensus

- The notion of *simulatability* [**lipton**]:
  1. Interpretability $\simeq$ how long it takes for human to make the same computations given an input.
  2. Interpretability $\simeq$ how much effort it would take a human to read through the entire policy once.
- Inside a given policy class, less parameters should mean more interpretability [**study-0**, **study-4**, **study-5**, **study-6**, **study-7**].
- The time required to formally verify a policy should decrease with interpretability [**viper**, **lens-complexity**].

A methodology to measure policy interpretability without humans

Simulatability [**lipton**]

1. How long it takes for human to make the same computations given an input $\simeq$ policy inference time.
2. How much effort it would take a human to read through the entire policy once $\simeq$ policy size in memory.

Not that simple in practice [**insight**]

- Different hardwares (tree policies are run on CPUs while neural policies are run on GPUs).
- Different implementations (neural policies compute outputs using matrix operations while tree operate fully sequentially) . . .

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. Does our methodology respect consensus on policy interpretability?
2. Is policy unfolding necessary to respect the consensus?
3. What kind of results we can obtain using our proposed methodology?

Empirical validation: obtaining ~ 40000 policies from different classes

Policy Class	**Parameters**	**Training algo.**
Linear policies	Determined by state-action dimensions	Linear/logistic Reg.
Decision trees	{4, 8, 16, 64, 128} nodes	CART
Oblique decision trees	{4, 8, 16, 64, 128} nodes	CART
Relu neural networks	{(2 ,2), (4, 4), (8, 8), (16, 16)} weights	SGD

Table: Summary of policy classes parameters and supervised learning algorithms to fit experts.

Empirical validation: obtaining $\sim$ 40000 policies from different classes

Classic	**MuJoCo**	**OCAtari**
CartPole (4, 2, **490**)	Swimmer (8, 2, **300**)	Breakout (452, 4, **30**)
LunarLander (8, 4, **200**)	Walker2d (17, 6, **2000**)	Pong (20, 6, **14**)
// Continuous (8, 2, **200**)	HalfCheetah (17, 6, **3000**)	SpaceInvaders (188, 6, **680**)
BipedalWalker (24, 4, **250**)	Hopper (11, 3, **2000**)	Seaquest (180, 18, **2000**)
MountainCar (2, 3, **90**)		
// Continuous (2, 1, -**110**)		
Acrobot (6, 3, -**100**)		
Pendulum (3, 1, -**400**)		

Table: Summary of considered environments (dimensions of states and number or dimensions of actions, **performance thresholds to solve**). OCAtari is an object-centric version of Atari.

Result 1: unfolding policies is necessary to respect consensus

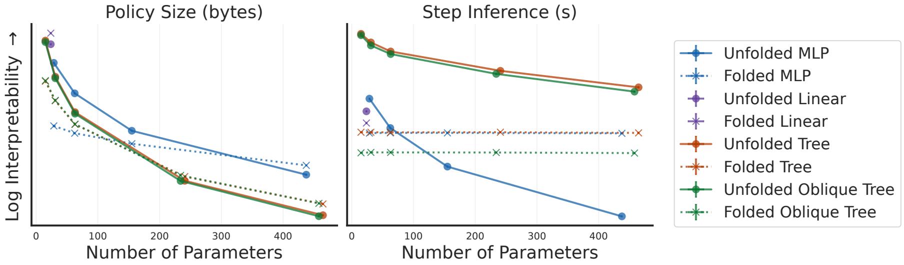


Figure: Policies interpretability on classic control environments. We plot 95% stratified bootstrapped confidence intervals around means in both axes. In each sub-plot, interpretability is measured with either bytes or inference speed.

Result 2: verification time does scale with step inference time

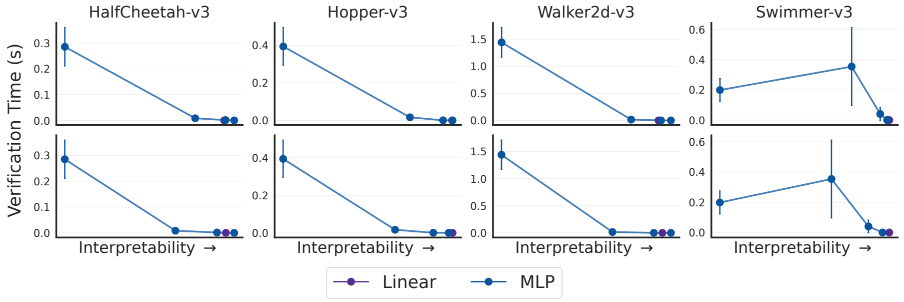


Figure: Verification time as a function of policy interpretability. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size.

Result 3: there is no dominating policy class for all environments

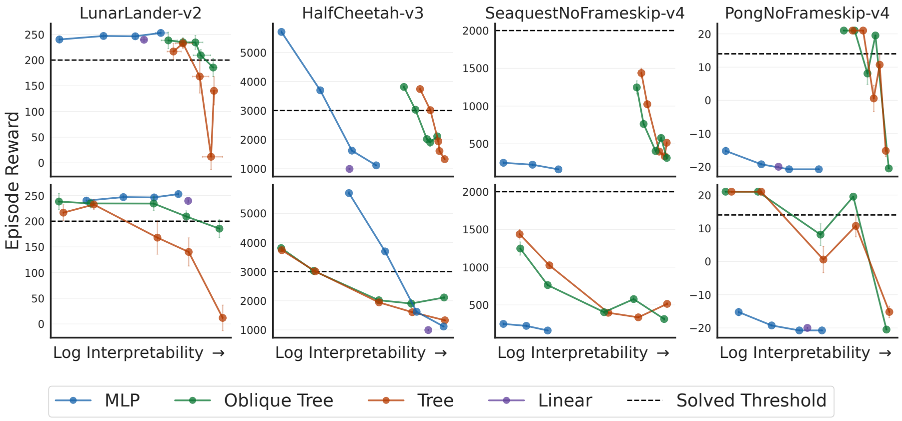


Figure: Interpretability-Performance trade-offs for representative environments. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size.

Take home messages

1. Because there is no dominating class for all problems in terms of interpretability-performance trade-offs, popular beliefs such as ”trees are more interpretable than neural networks” should be used with caution.
2. This further motivates the use of the proposed methodology when comparing policies from different classes.

Perspective

- Can a human study confirm our results?
- Can our methodology be used for evaluating the interpretability of (very) big models?
- Can we use our policy programs as low level skills (hierarchical RL)?

All the policy programs are available on github
`https://github.com/KohlerHECTOR/interpretable-rl-zoo`

Conclusion