Interpretable Sequential Decision Making

Hector Kohler

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

November 11, 2025

About me

- PhD student *Interpretability, Decision Trees, and Sequential Decision Making* at Inria in Lille (2022-Decembre 2025).
- Looking for a postdoc.
- Want to become prof./researcher.

Sequential decision making

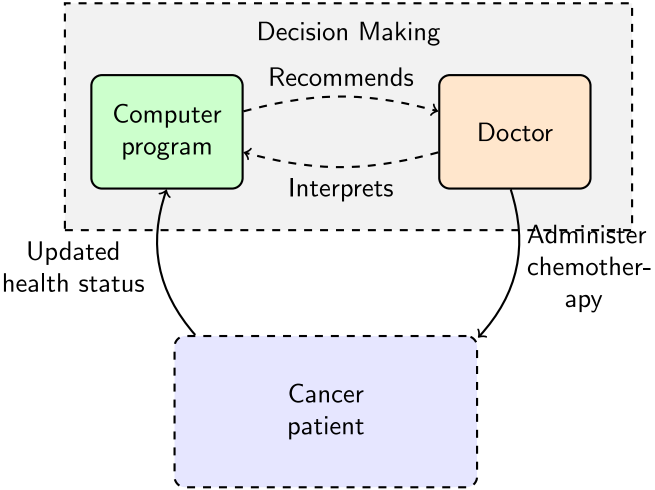


Figure: Sequential decision making in cancer treatment.

Machine learning (ML) to learn policies for sequential decision making

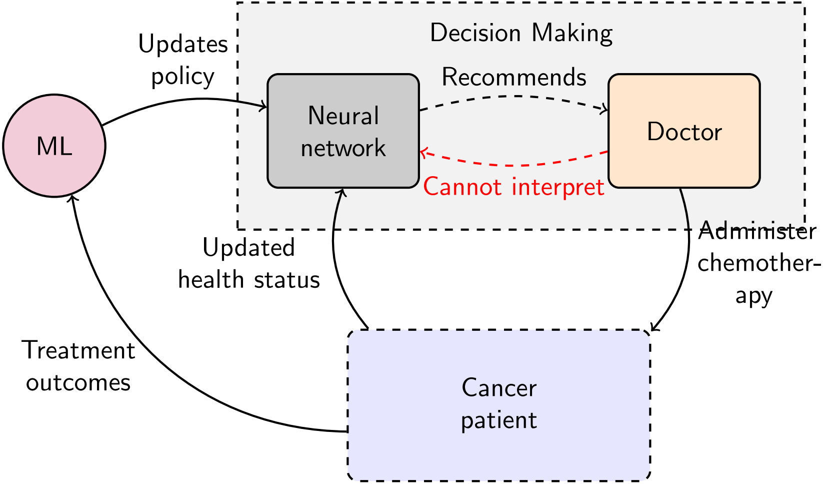


Figure: Machine learning of neural networks has many recent successes but neural networks are black-box.

Interpretable ML for sequential decision making

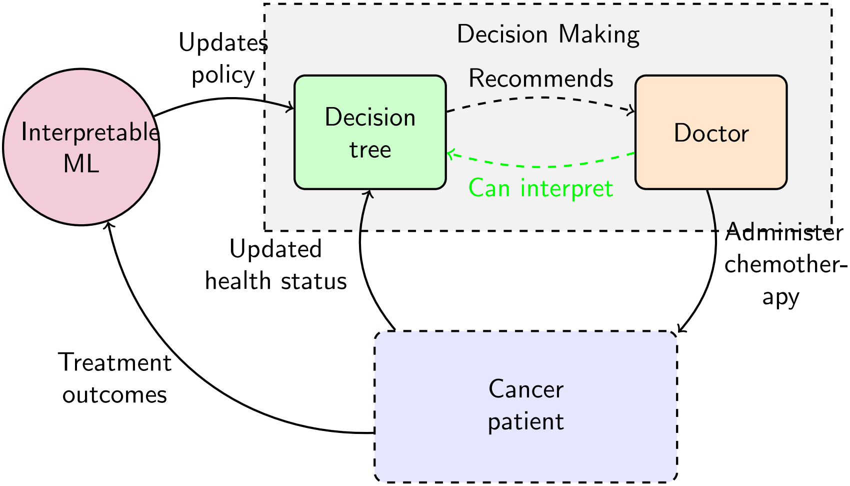


Figure: Some machine learning algorithms can learn interpretable policies, e.g. decision trees.

How to measure policy interpretability?

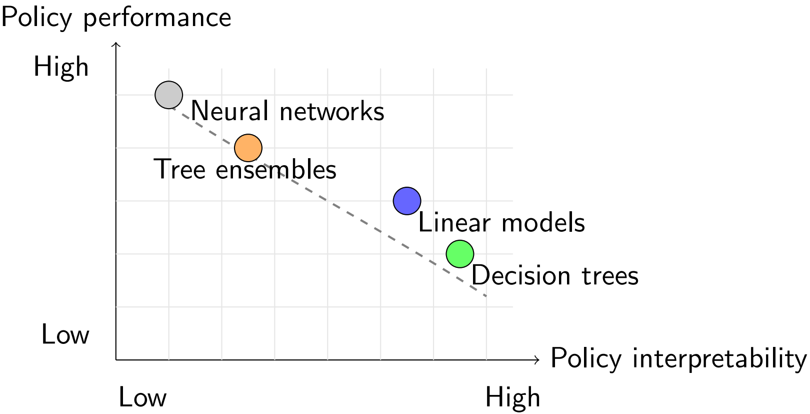


Figure: **Heuristic** interpretability-performance trade-offs of different policy classes (what-if neural network is very sparsed?). Interpretability is often presented in opposition to performances.

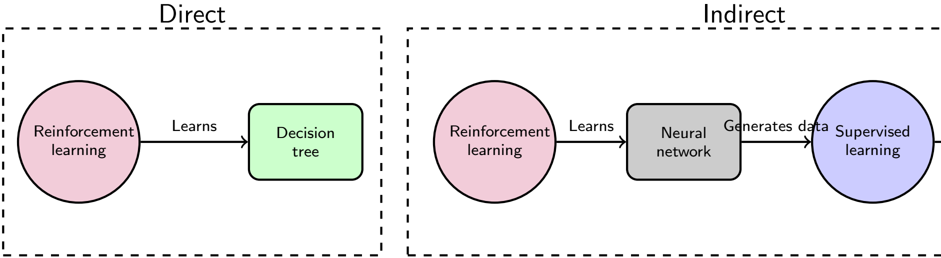


Figure: Comparison of direct and indirect approaches for learning interpretable models in sequential decision making.

Decision trees

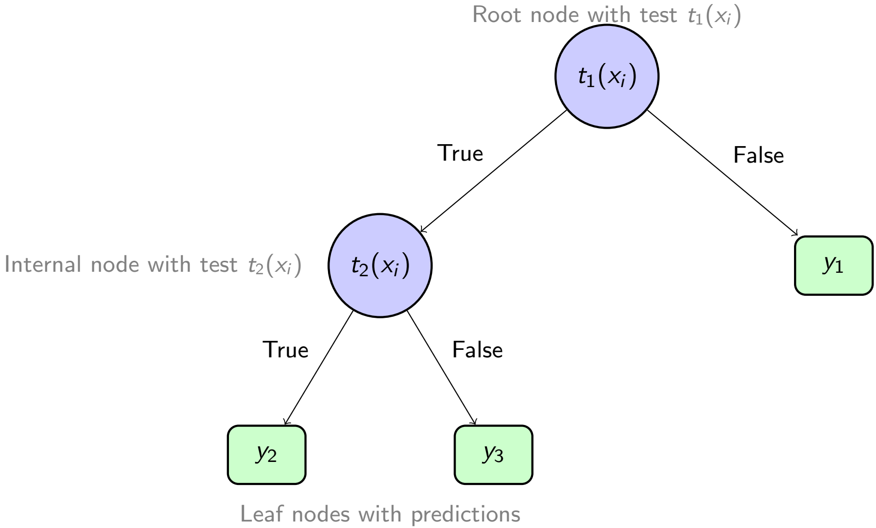


Figure: A generic decision tree of depth $D = 2$.

Decision trees for MDPs

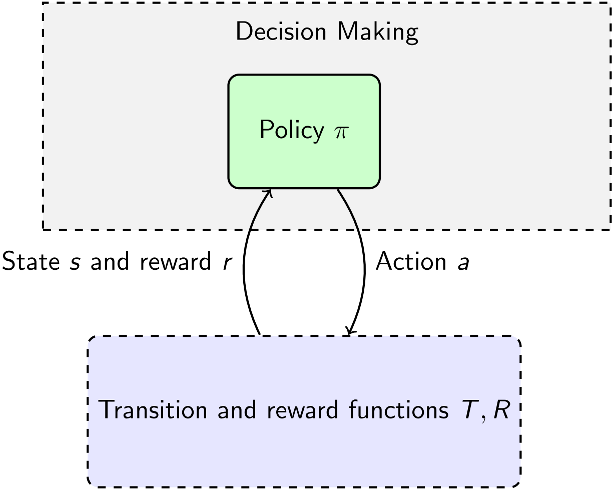


Figure: Markov decision process

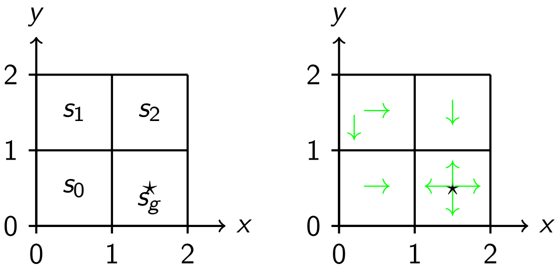


Figure: A grid world MDP (left) and optimal actions w.r.t. the objective (cf. definition **??**) (right).

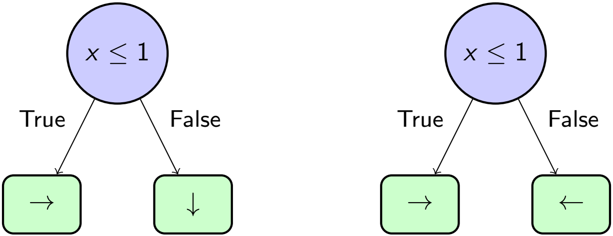


Figure: Left, an optimal depth-1 decision tree policy. On the right, a sub-optimal depth-1 decision tree policy.

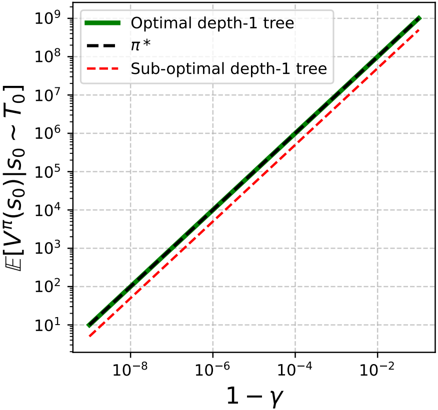


Figure: The RL objective values of the optimal policies from figure 8 and of the decision tree policies from figure 9.

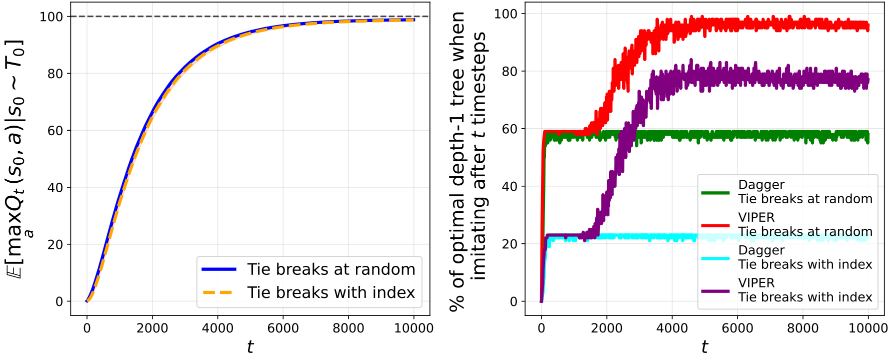


Figure: Left, sample complexity curve of Q-learning with default hyperparameters on the 2×2 grid world MDP over 100 random seeds. Right, performance of indirect interpretable methods when imitating the greedy policy with a tree at different Q-learning stages.

Interpretability, decision trees, and sequential decision making

- How to learn optimal interpretable policies for sequential decision making?
- **How to leverage sequential decision making to learn interpretable classifiers for supervised learning?**
- How to measure policy interpretability in sequential decision making?

1. Direct reinforcement learning of decision tree policies is hard because it involves POMDPs.
2. One can use dynamic programming in MDPs to induce highly performing decision tree classifiers and regressors.
3. In practice, controlling MDPs with interpretable policies does not necessarily decrease performances.

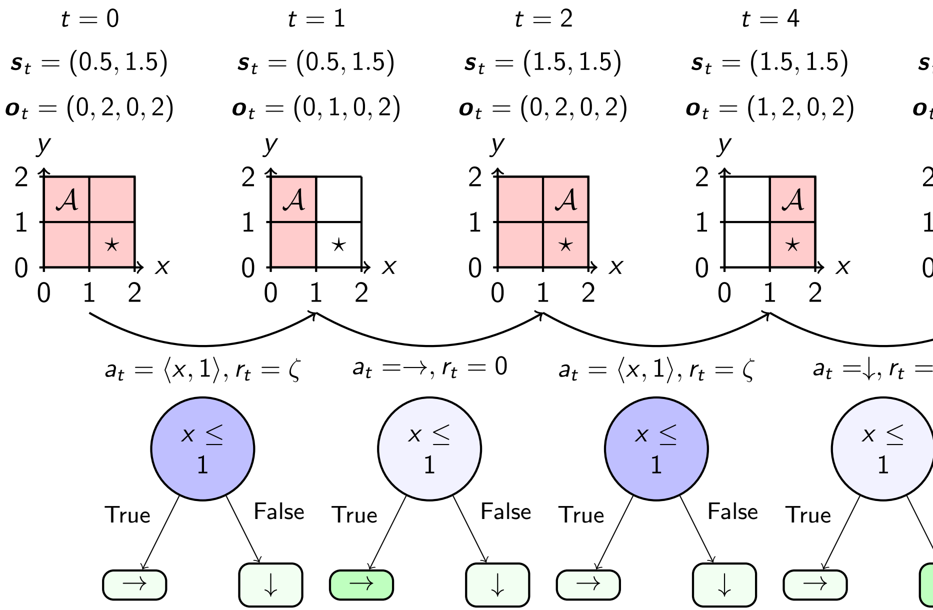


Figure: An IBMDP trajectory when the base MDP is 2×2 grid world. In the top

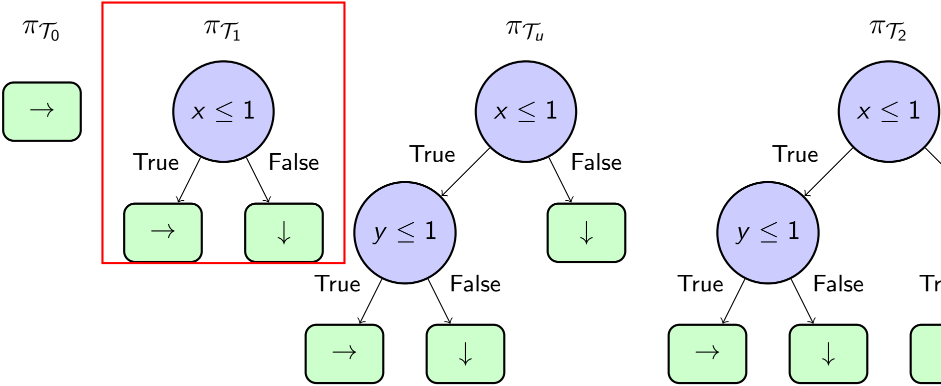


Figure: For each decision tree structure, e.g., depth-1 or unbalanced depth-2, we illustrate a decision tree which maximizes the RL objective (cf. definition **??**) in the grid world MDP.

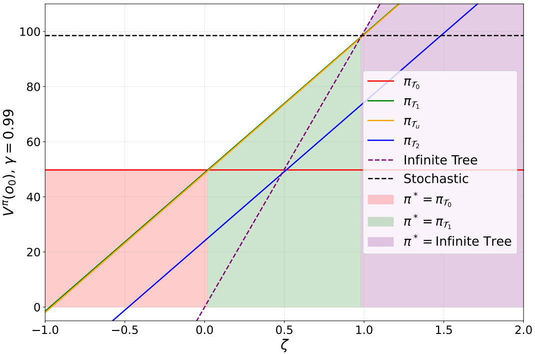


Figure: Interpretable RL objective values (cf. definition **??**) of different partially observable policies as functions of ζ. Shaded areas show the optimal *deterministic* partially observable policies in different ranges of ζ values.

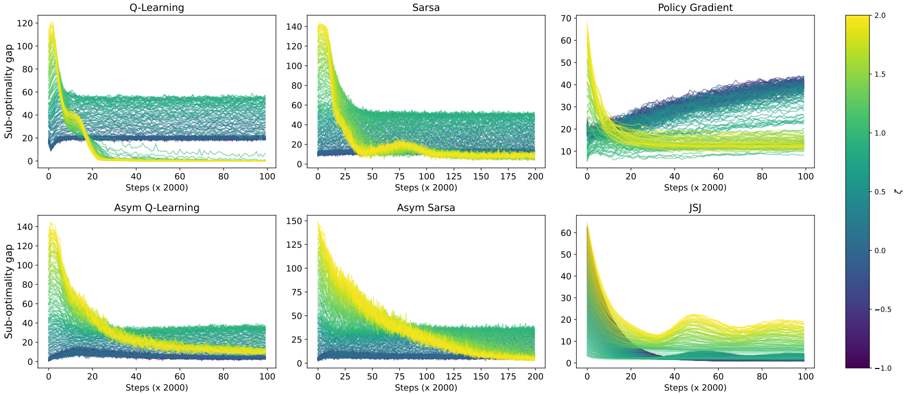
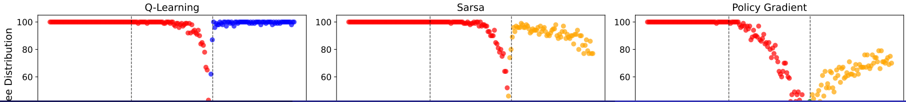


Figure: (Asymmetric) reinforcement learning in POIBMDPs. In each subplot, each single line is colored by the value of ζ in the corresponding POIBMDP in which learning occurs. Each single learning curve represent the sub-optimality gap averaged over 100 seeds.



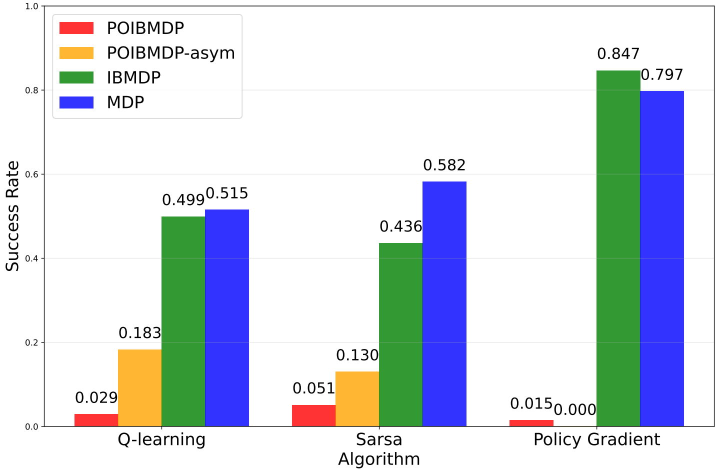
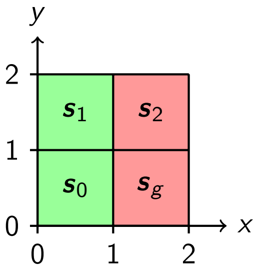


Figure: Success rates of different (asymmetric) RL algorithms over thousands of runs when applied to learning deterministic partially observable policies in a POIBMDP or learning deterministic policies in associated MDP and IBMDP.



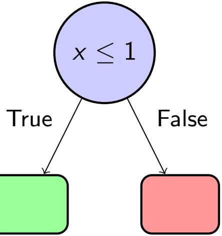


Figure: Classification MDP optimal actions. In this classification MDP, there are four data to which to assign either a green or red label. On the right, there is the unique optimal depth-1 tree for this particular classification MDP. This depth-1 tree also maximizes the accuracy on the corresponding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

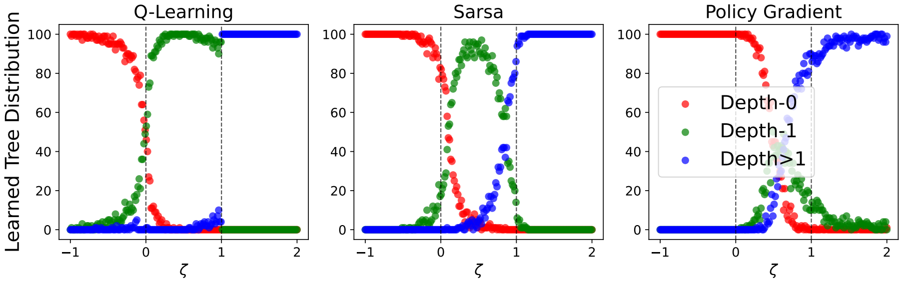


Figure: We reproduce the same plot as in figure 16 for classification POIBMDPs. Each colored dot is the number of final learned trees with a specific structure for a given ζ.

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

Why decision trees ?

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?

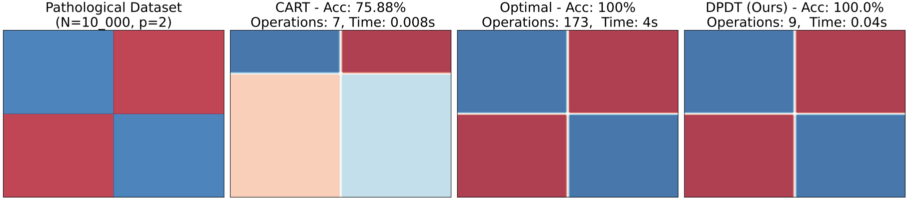


Figure: A checkers board data set highlights the limitations of existing works.

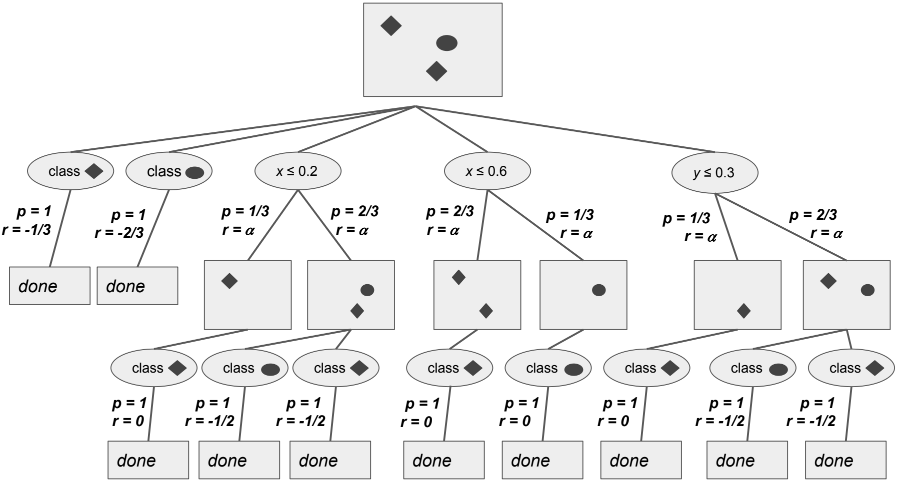
Decision tree induction as solving MDPs

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



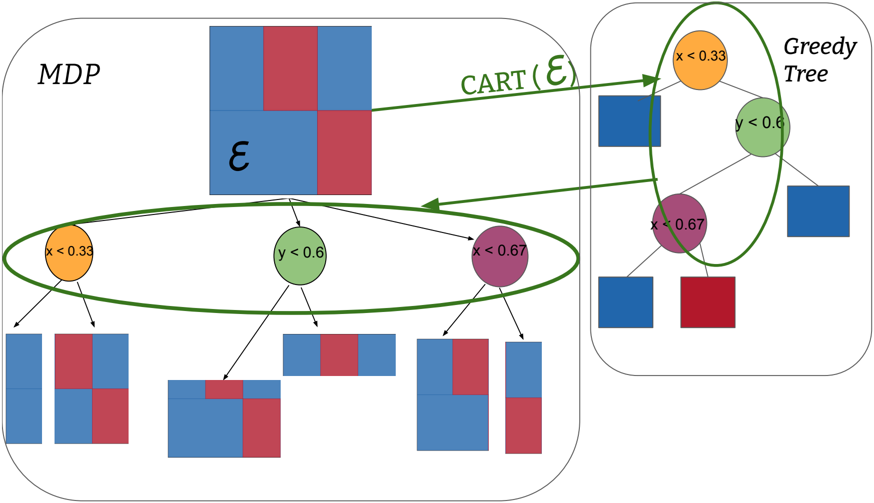
Controlling the time complexity of decision tree induction

- 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]

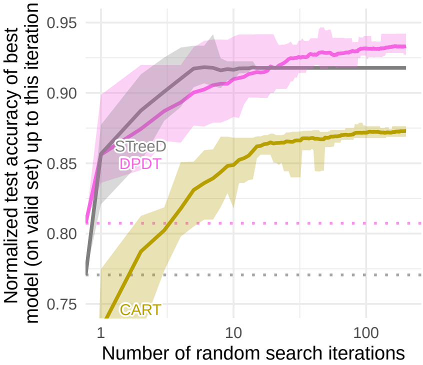


[1]Because states are entire datasets, we implement DPDT with a depth-first search to limit the space complexity.

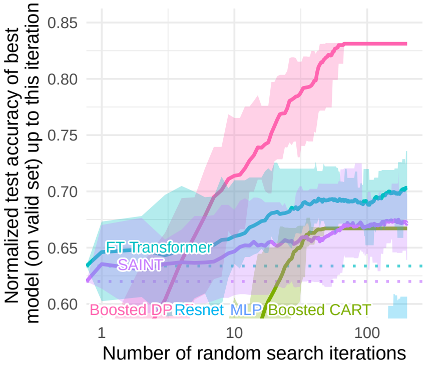
Comparing trees accuracy versus complexity

			Accuracy				Operations			
Dataset	N	p	Opt Quant-BnB	Greedy CART	DPDT light	DPDT full	Opt Quant-BnB	Greedy CART	DPDT light	DPDT 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

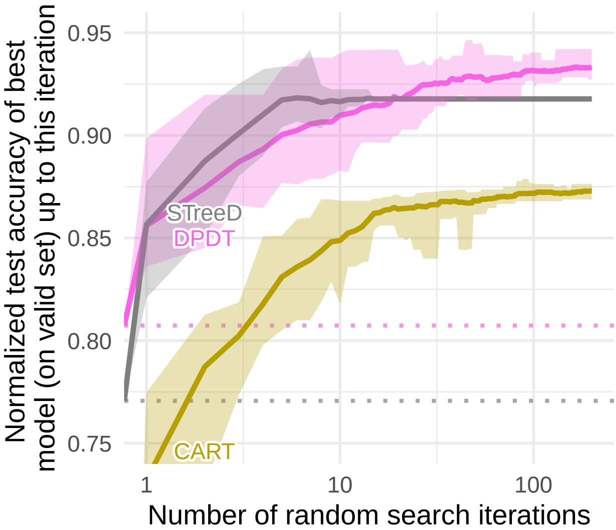


(a) DPDT trees vs. other trees

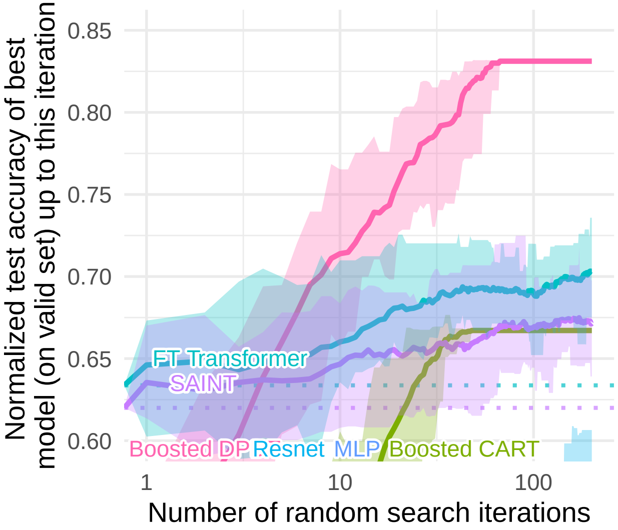


(b) Boosted DPDT vs. other classifiers

DPDT generalization



Boosting DPDT



How to measure policy interpretability?

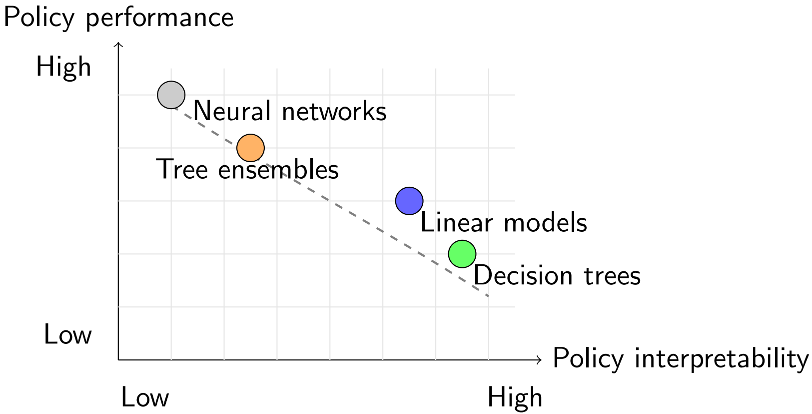


Figure: **Heuristic** interpretability-performance trade-offs of different policy classes (what-if neural network is very sparsed?). Interpretability is often presented in opposition to performances.

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

```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]
```

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 $\sim$ 40000 policies from different classes

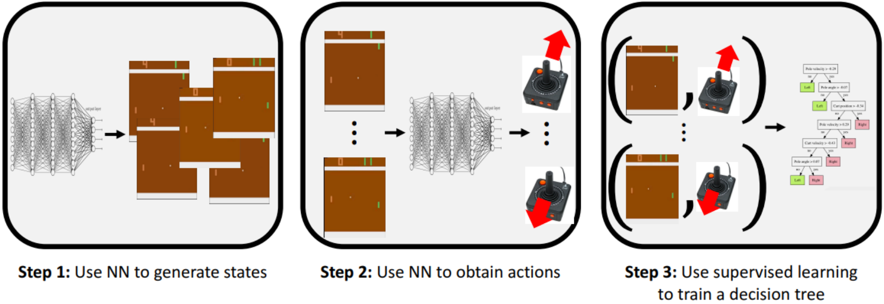


Figure: Imitation Learning to get policies from different classes (behaviour cloning, DAgger, VIPER) [**viper**, **dagger**, **behavior-cloning**]. The experts (a.k.a. oracles or teachers) neural network policies are obtained from the `stable-baselines3 zoo` [**zoo**].

Empirical validation: obtaining $\sim$ 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.

Metrics

1. For each policy class and each environment, we keep only the best policy in terms of performance for the task.
2. We measure the interpretability of each best-in-class policy on dedicated CPUs.
3. To measure interpretability, we track two metrics:
    1. Average inference time in seconds to predict actions given states.
    2. Space in memory in bytes.

Result 1: unfolding policies is necessary to respect consensus

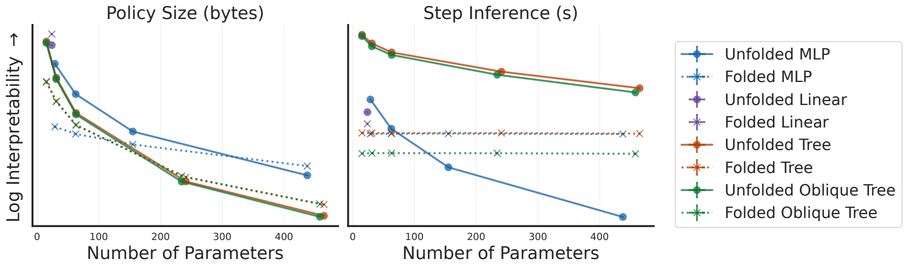


Figure: Policies interpretability on classic control environments. We plot 95% stratified bootstrapped confidence intervals around means in both axes. In each sub-plot, interpretability is measured with either bytes or inference speed.

Result 2: verification time does scale with step inference time

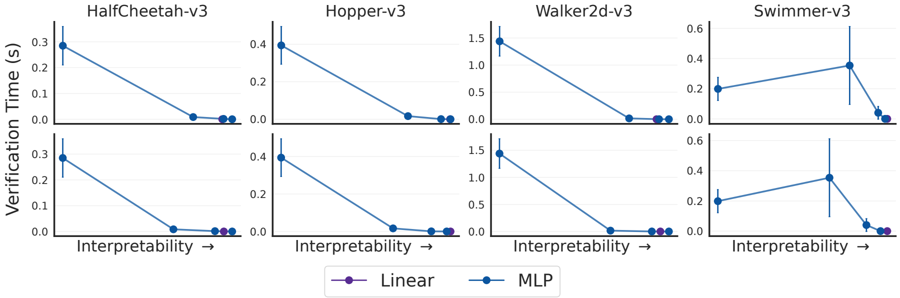


Figure: Verification time as a function of policy interpretability. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size.

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

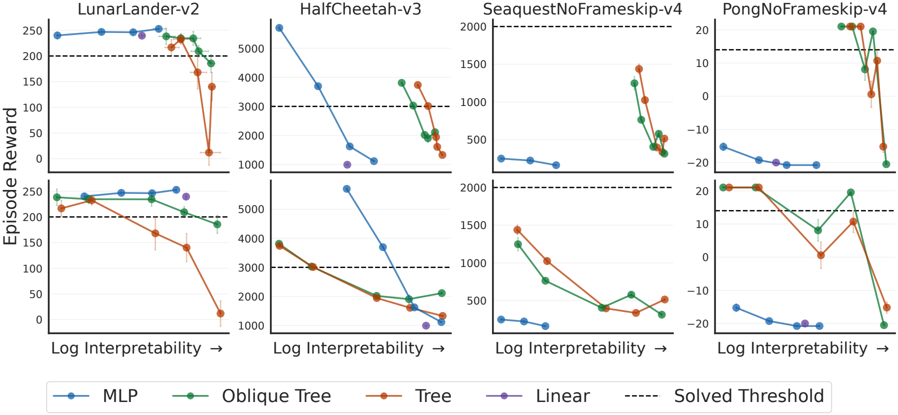


Figure: Interpretability-Performance trade-offs for representative environments. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size.

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.

Future work

- 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`