Interpretability and Performance of Surrogate Decision Trees Produced by Viper

Otto Kaaij[1]
Supervisor: Anna Lukina[1]
[1]EEMCS, Delft University of Technology, The Netherlands
o.k.n.kaaij@student.tudelft.nl, A.Lukina@tudelft.nl

Abstract

Machine learning models are being used extensively in many high impact scenarios. Many of these models are 'black boxes', which are almost impossible to interpret. Successful implementations have been limited by this lack of interpretability. One approach to increasing interpretability is to use imitation learning to extract a more interpretable surrogate model from a black box model. Our aim is to evaluate Viper, an imitation learning algorithm, in terms of performance and interpretability. To achieve this, we evaluate surrogate decision tree models produced by Viper on three different environments and attempt to interpret these models. We find that Viper generally produces high performance interpretable decision trees, and that performance and interpretability are highly dependent on context and oracle quality. We compare Viper performance to similar imitation learning approaches, and find that it performs as good as or better than these approaches, though our comparison is limited by the differences in oracle quality.

1 Introduction

In many domains and contexts, machine learning systems are increasingly used to make critical, high-impact choices that affect human lives. For instance, decisions about mortgages and the stock market, healthcare, parole, the structural integrity of bridges and self-driving cars are all domains in which machine learning has been applied [17]. However, in all of these areas, successful implementation has been limited by the lack of guarantees possible about robustness, correctness, performance, and model behavior in general [5, 23, 25, 18, 29].

The reason is that complex machine learning models are often 'black boxes': it can be difficult to understand why the model made a prediction. It might be unclear what features of the dataset play an important role in the model, or what combination of features will lead to a certain outcome.

For a system to be used in critical environments, it must be trusted, and this trust can only be achieved by ensuring the system is transparent and that its actions are justifiable. Additionally, AI has traditionally been used as 'additional input to otherwise soundly defined control systems.' However, over the last decades, there has been a tendency towards completely autonomous, completely AI based systems [18]. In these cases, it is clearly desirable to break through the 'black box', and thus be able to interpret the decisions the system makes. For these reasons, it is valuable to consider interpretability when creating models. This is reflected by the large amount of work in the field of explainable AI [18].

Many new approaches in machine learning are evaluated only on performance and accuracy, but not on interpretability. Because explainability and interpretability is, as argued above, crucial for successful and appropriate usage of these models in critical environments like healthcare and self-driving cars, all of these methods can benefit from a focus on explainability.

In our context, we are interested in training surrogate models from expert behavior. This process is called imitation learning, and it has gained much attention in recent years. Numerous methods have been presented [12]. Imitation learning attempts to train a policy – what actions to take based on a certain observation – by learning from expert demonstrations. These experts can be human. For example, we can use human demonstrations of driving a car to learn how to drive a car. On the contrary, these experts can also be existing machine learning models, and this is what we are interested in. In summary, we use imitation learning to extract a new, more explainable surrogate model from a 'black box' model, We evaluate Viper, an imitation learning algorithm presented by Bastani, Pu, and Solar-Lezama [5]. It uses an expert policy to extract decision trees, leveraging the oracles Q-function to sample more valuable data points and emphasize accuracy on critical states.

We use decision trees as surrogate models. There are two important justifications for this choice. Firstly and most importantly, decision trees – especially small ones – are inherently interpretable. The highly structured nature of these trees and the simplicity of the choices, which are just yes or no questions about the environment, make

a decision tree much easier to interpret than, for example, a neural network. Secondly, decision trees are simple and computationally inexpensive to train, while retaining the expressiveness to encode complex policies that generalize well. We use CART [6] to train decision trees.

Our goal is to evaluate VIPER in terms of performance and interpretability. In addition, VIPER will be compared with other algorithms that learn surrogate models from expert policies: AGGREVATE [21] and GENERATIVE ADVERSARIAL IMITATION LEARNING (GAIL) [11].

We conclude that VIPER produces high performance, interpretable decision trees for simple environments. We also find that the reward obtained by the surrogate models is dependent on more than just the average oracle reward, and we suggest possible improvements to VIPER.

Section 2 discusses related work. Section 3 discusses the algorithms used, and some needed terminology. Section 4 defines the problem we are solving, and discusses the method used to tackle this problem. In section 5 the experimental setup and the results from those experiments are presented. In sections 6 and 7 these results are discussed and the main conclusions are presented. We also include some suggestions for future work. In section 8 we reflect on the ethical aspects of this research, as well as the reproducibility of the methods used.

2 Related Work

We explore relevant work related to imitation learning and related to interpretability. There is a large volume of research into both of these areas, with results that are both interesting on their own and useful for our work.

Imitation learning

An extensive body of work exists on imitation learning. The simplest form of imitation learning, behavioral cloning, suffers from compounding errors when extracted policies make mistakes [22]. Varying approaches have been developed to circumvent this, using for example inverse reinforcement learning (IRL) [1] and genetic programming [26]. IRL approaches can be very expensive, requiring a full pass of reinforcement learning as an inner loop. GENERATIVE ADVERSARIAL IMITATION LEARNING [11] attempts to directly extract a policy, mimicking the indirect IRL process. Atkeson and Schaal [3] use an imitation learning approach that directly mimics the expert. However, this approach produces models that do not generalize well; blindly mimicking the expert might not be the best approach, as a model must also be able to make the right choice in unexpected scenarios.

DAGGER [22] attempts to circumvent the problem of compounding errors that behavioral cloning suffers from by leveraging expert supervision to aggregate the dataset. AGGREVATE [21] and VIPER are extensions of DAGGER, leveraging the cost-to-go function and the Q-function, respectively.

Wols and Lukina [28] and Meijer and Lukina [15] provide evaluations of respectively AGGREVATE and GAIL that are similar to our evaluation.

Interpretability

A large and growing body of literature has investigated the interpretability of AI models, both within and outside of the domain of imitation learning. Puiutta and Veith [18] provide a survey of interpretability methods in reinforcement learning. Some imitation learning algorithms are created in an explicit attempt to increase interpretability [26, 4, 14]. More generally, Molnar [17] offers a comprehensive overview of interpretability methods, ranging from using intrinsically interpretable models such as decision trees, to post hoc methods such as feature importance metrics and visualizing model internals. Explainability and interpretability are often used interchangeably [17], there is no consensus on their definitions [13] and their definitions differ, depending on domain and context [23]. While some quantitative metrics can be devised to evaluate the interpretability of a decision tree [8, 24], it is important to realize that a qualitative evaluation, such as in Doshi-Velez and Kim [9], might be more valuable. We use this approach to evaluate the surrogate models produced by VIPER.

3 Preliminaries

We first define necessary terms and functions. Then we provide a brief overview of the algorithms used for training oracle policies (Q-LEARNING and DQL) and the imitation learning algorithms (behavioral cloning and VIPER).

3.1 Definitions

Imitation learning finds a policy π for a certain environment, given an oracle policy π^*. We let (S, A, P, R) be a Markov Decision Process with time horizon T, where S is the set of states, A the actions, $P : S \times A \times S \to p \in \{0, 1\}$ are the transition probabilities (note that p is either 1 or 0, because all evaluated environments are deterministic), and $R : S \to \mathbb{R}$ is the reward function. Without loss of generality, we assume that there is a single initial state $s_0 \in S$. A policy is a function $\pi : S \to A$. We let

$$Q_t^\pi(s, a) = R(s) + \sum_{s' \in S} P(s, a, s') V_{t+1}^\pi(s')$$

be its Q-function, where $V_t^\pi(s) = Q_t^\pi(s, \pi(s))$ and $V_T^\pi(s) = 0$.

Let $d^\pi(s)$ be the distribution over states of π: it is 1 if the state s is visited by π at any time, starting at s_0. We use fidelity as a metric of similarity between two policies π_0 and π_1. This fidelity is given by:

$$\text{fid}(\pi_0, \pi_1) = \frac{100}{T} \sum_{t=0}^{T} \mathbb{I}[\pi_0(s_t) = \pi_1(s_t)]$$

where $\mathbb{I}$ is the indicator function. We use this to investigate how well π represents the oracle policy π^*.

Oracle policies are trained using either Q-LEARNING [27] or DEEP Q-LEARNING (DQL) [16]. From these oracle policies, π is extracted using VIPER. We use behavioral cloning as a baseline for imitation learning performance.

3.2 Training the oracle

For most environments we use DQL to train π^*. For the Mountaincar environment we also use Q-LEARNING to train π^* and investigate the dependence on oracle quality.

The Q-learning algorithm

Q-LEARNING [27] is used to train oracle policy π^*. Q-LEARNING is a traditional approach to reinforcement learning. It is used as a baseline learning performance because of its simplicity and reliability. By repeatedly interacting with the environment and observing the received reward, it learns the value of actions in certain states. Using this information, it learns a Q-table: for each (s, a) pair, it stores the current value estimate. This allows it to determine the best action in a state.

However, Q-LEARNING can not be used with continuous state spaces, as this would result in an infinitely large table. In order to circumvent this, we use DEEP-Q LEARNING. In the cases where we do use DQL with continuous state variables, we discretize the states.

Deep Q-learning

The core difference between DQL [16] and Q-LEARNING is that where the latter uses a table with value estimates, the former uses this table with a neural network. This allows the algorithm to work better for environments that require much detail, and where discretizing would throw out valuable precision.

DQL is used as the baseline for learning performance, to evaluate the performance difference between a traditional reinforcement learning approach and the imitation learning approach of VIPER.

3.3 Imitation learning algorithms

Two algorithms are used to extract π from π^*. Behavioral cloning is used as a baseline for imitation learning performance. VIPER is the main algorithm under evaluation.

Behavioral Cloning

Behavioral cloning is a simple algorithm (see Algorithm 1). TRAIN(D) uses CART[6] to train decision trees. Because of its simplicity, it is used as a baseline. However, behavioral cloning has a significant problem: it only gets data for states that π^* visits. If π makes a mistake, it will encounter states that it does not have knowledge about. Because of this, it may make another mistake. These mistakes can compound as π keeps making mistakes in states it has not seen before, leaving π unable to recover.

The Viper algorithm

VIPER[5] is an imitation learning algorithm that leverages the oracles Q-function to prioritize accuracy on critical states. The algorithm is shown in Algorithm 2. TRAIN(D) again uses CART[6] to train decision trees.

VIPER takes state-action pairs provided by expert demonstrations to train an initial decision tree. Then, iteratively, the decision tree explores the state space, and queries the oracle for supervision, allowing it to learn to recover. The dataset is then aggregated with these $(s, \pi^*(s))$ pairs. Then, VIPER leverages the oracles Q-function to resample pairs, giving higher probability to points, where making the worst choice leads to the most loss:

$$p(s) = V_t^{(\pi^*)}(s) - \min_{a \in A} Q_t^{(\pi^*)}(s, a)$$

Calculating this value is difficult when using neural network policies like in DQL. In this case, we let

$$p(s) = \max_{a \in A} p^{\pi}(s, a) - \min_{a \in A} p^{\pi}(s, a)$$

where $p^{\pi}(s, a)$ is the probability that π chooses action a in state s.

Intuitively, this means that D' will consist of more *critical* state-action pairs: pairs in which making the right choice is important. In a sense, VIPER prioritizes accuracy on these critical states to accuracy on less critical states, allowing it to train smaller decision trees more effectively.

Algorithm 1 Behavioral cloning

procedure BC($(S, A, P, R), \pi^*, N$)
  Sample N trajectories $D \leftarrow \{(s, \pi^*(s)) \sim d^{\pi^*}\}$
  Train decision tree $\pi \leftarrow$ Train(D)
  return π
end procedure

Algorithm 2 The VIPER algorithm

procedure VIPER($(S, A, P, R), \pi^*, Q^*, M, N$)
  Initialize empty data set $D \leftarrow \varnothing$
  Initialize initial policy $\pi \leftarrow \pi^*$
  for $i \leftarrow 1$ **to** N **do**
    Sample trajectories $D_i \leftarrow \{(s, \pi^*(s)) \sim d^{\pi_{i-1}}\}$
    Aggregate datasets $D \leftarrow D \cup D_i$
    Resample dataset $D' \leftarrow \{(s, a) \sim p((s, a))\}$
    Train decision tree $\pi_i \leftarrow$ Train(D')
  end for
  return best policy $\pi \in \{\pi_i, \ldots, \pi_N\}$
end procedure

4 Approach

Problem definition. *Given an oracle policy $\pi^* : S \to A$, our goal is to evaluate student policy $\pi : S \to A$, extracted by* VIPER*, in terms of interpretability and performance. We let π be a decision tree, since decision trees are both highly interpretable and highly expressive. We focus on simple environments: Cartpole, Mountaincar and Acrobot. We evaluate performance in terms of achieved reward and consistency, and evaluate interpretability using intrinsic model qualities and a qualitative analysis of the decision trees.*

Given an oracle π^* and a corresponding student policy π, we evaluate three properties. First, we look at the

performance of π, and check if it achieves similar average reward and is as consistent as π^*. Next, we check the fidelity of π with regard to π^*, in order to see to what extent π represents the same approach to the problem as π^*. In practice, we might want to use π to interpret and gain insight into the original oracle policy π^*. To do so, it is necessary to ensure that π encodes a similar policy.

Finally, we evaluate the interpretability of π. We use decision trees as models for π. Two things are immediately clear: decision trees are some of the most inherently interpretable models, and small trees are more interpretable than large trees. This gets us our first metric of interpretability: the height of the tree [17], or strongly related with this, the number of nodes or the maximum/average depth [13]. We train decision trees that are as small as possible without sacrificing performance. Then we attempt to interpret these trees, using a human-grounded approach, as in Doshi-Velez and Kim [9]. Given that our environments are relatively simple, and given the intrinsic interpretability of decision trees, we can attempt to interpret the decision trees as a whole.

In addition to these three properties, we use the analyses given in Wols and Lukina [28] and Meijer and Lukina [15] to compare VIPER to respectively AGGREVATE and GAIL. In addition to comparing performance and objective metrics like decision tree depth, we use the same qualitative, human grounded approach to interpretability.

5 Experimental Setup and Results

In summary, the experimental setup is as follows:

- We train a DQL oracle π^* on three environments: Cartpole, Acrobot and Mountaincar
- We use behavioral cloning to extract a decision tree policy π^{bc} for baseline imitation learning performance.
- We use VIPER to extract a decision tree policy π.
- We evaluate the interpretability of π, and its performance compared to π^* and π^{bc}.

Additionally, we use Mountaincar to investigate the dependence of VIPER on oracle quality. To do so, we compare its performance on the DQL oracle and on three different Q-LEARNING oracles.

For DQL, we use the deep-Q network implementation given in the Stable Baselines project [10]. Hyperparameters for DQL were taken from the RL Baselines Zoo project [20], which provides tuned hyperparameters for many OpenAI gym environments. We found hyperparameters for VIPER and behavioral cloning using an informal search[1].

[1]Performing more systematic hyperparameter optimization would likely result in slightly better performance. However, we think that the improvements gained would be marginal and have no significant impact on the results and conclusions in this text. For instance, the values used in the splits of the trees might slightly alter, but the structure of the trees would stay the same.

For Mountaincar, we produce decision trees with a depth of 3. The resulting tree has 8 leaf nodes and 7 internal nodes. For Cartpole and Acrobot, a tree of depth 2 suffices, resulting in 4 leaf nodes and 3 internal nodes. In cases where it is applicable, we manually apply prune the tree, removing any node that lead to the same action in all cases. For all three environments, creating larger trees does not lead to significantly better performance, and creating smaller trees leads to significantly worse performance.

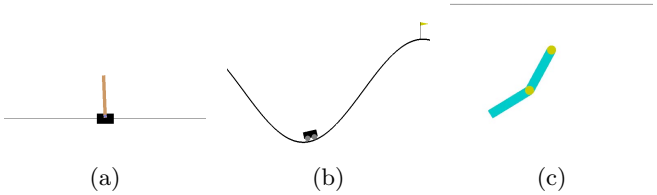
5.1 The environments: Cartpole, Acrobot and Mountaincar

We evaluate VIPER on three environments. See figure 1 for examples of the environments.

Mountaincar has a 2 dimensional state space $(x, v) \in \mathbb{R}^2$, where x is the horizontal position of the cart and v its velocity. It has a discrete action space $A = \{\textit{left}, \textit{right}\}$. The goal is to get the cart to the flag. To achieve this, the car must build momentum by going back and forth. The environment gives a reward of -1 for every timestep, and has a time horizon $T = 200$ The environment is considered solved when, averaged over 100 rollouts, a reward greater than -110 is achieved.

Cartpole has a 4 dimensional state space $(x, v, \theta, \omega) \in \mathbb{R}^4$ where x is the cart position, v the cart velocity, θ the pole angle, and ω is the angular velocity of the pole. It has a discrete action space $A = \{\textit{left}, \textit{nothing}, \text{RIGHT}\}$. The goal is to balance the pole for 500 timesteps. The environment gives a reward of $+1$ for every timestep the pole remains balanced. The environment is solved when a consistent reward of 500 is achieved.

Lastly, Acrobot features a double pendulum. It has two links and two joints, where the middle joint is actuated. It has a 6 dimensional state space for the *sin* and *cos* of the angle of both joints, and the velocities of those joints: $(sin(\theta_1), cos(\theta_1), sin(\theta_2), cos(\theta_2), \dot{\theta_1}, \dot{\theta_2})$. It has a discrete action space $A = \{-1, 0, +1\}$, which are torques applied to the lower of the two joints. The goal is to swing the lower link to a given height. A reward of -1 is given for every timestep. The environment does not have a set target reward, so we use DQN performance as a baseline.



(a) (b) (c)

Figure 1: Three simple environments. (a) shows the Cartpole environment: the pole must remain balanced on the cart. (b) shows the Mountaincar environment: the car must get to the flag. (c) shows the Acrobot environment: the tip of the lower link must get above the line.

Benchmark	π^*	π	π^{bc}	fid(π, π^*)
Cartpole	$\mathbf{500.0 \pm 0.0}$	$\mathbf{500.0 \pm 0.0}$	466.5 ± 33.5	62.8%
Acrobot	-85.9 ± 17.1	$\mathbf{-84.3 \pm 20.81}$	-114.2 ± 93.8	82.1%
Mountaincar	-112.1 ± 1.8	$\mathbf{-111.2 \pm 2.7}$	-114.17 ± 3.3	97.1%

Table 1: Average rewards and standard deviations over 500 rollouts of the DQL oracle π^*, the student policy π extracted with VIPER and π^{bc}, extracted with behavioral cloning, and the fidelity of π with regard to π^*. For each environment, the policy that achieves the highest reward is in bold.

5.2 Performance and interpretability in simple environments: Mountaincar, Acrobot and Cartpole

Table 1 shows the average reward per rollout and its standard deviation (over 100 rollouts) for π^* and the corresponding policy π extracted by VIPER and π^{bc} extracted by behavioral cloning. While performance is not the main focus here, the table shows that VIPER consistently outperforms behavioral cloning, and is able to learn decision trees that achieve performance close to π^*.

Additionally, the table shows the fidelity of the policies with regard to the oracles. We see that for Mountaincar π represents π^* very well. It has a fidelity of 97.1%. The Acrobot policy also has a fairly high fidelity. It is lower (82.1%), but this is expected: as we will see in the analysis, this policy uses some random chance to solve the environment. In these cases, interpreting π can be used as a proxy to interpret π^*. For Cartpole, the fidelity is significantly lower: only 62%. We think this is because in Cartpole, small inefficiencies do not have a direct effect on reward, as long as we can recover and keep the pole upright. Performing the action sequence $(Left, Left, Right, Right)$ might result in the same state as performing $(Left, Right, Left, Right)$, but significantly reduce fidelity. Still, as we will see, the interpretation of π has value, and is still relevant, even as a tool for interpreting π^*.

Mountaincar

Figure 2 shows the decision tree for the Mountaincar environment. It has 11 nodes and uses both features of the state space: x and v. Note that the lowest point of the valley is at $x = -0.5$. The tree is simple to interpret: if the cart is moving to the left it keeps moving left, unless it is both moving left very slowly and on the left slope. If the cart is moving right, the cart keeps moving right, unless it is moving right slowly and is on the right slope. This corresponds neatly to an intuitive understanding of how to solve Mountaincar.

It is clear that the tree encapsulates the symmetrical nature of the environment. We note that when reversing directions on the left slope, the tree immediately applies acceleration to the right. In contrast, on the right slope the tree reverses direction without applying further acceleration and letting gravity pull the car back down. According to our intuitive understanding of Mountaincar this must be inefficient. And it is: manually changing that node to perform action '0: left' increases the performance of the tree to -108 ± 2.9. This is a significant improvement. This modified tree 'solves' the environment where the original does not: the tree achieves an average reward larger than -110.

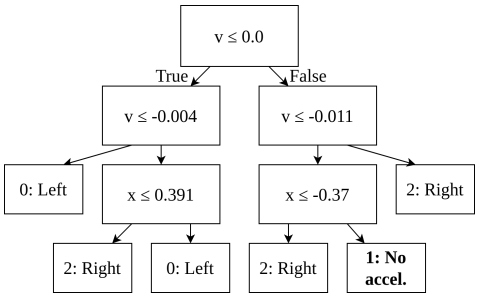


Figure 2: Decision tree that solves Mountaincar. The node in bold is an 'inefficient' node. Manually changing this to '0: Left' improves performance.

Cartpole

Figure 3 shows the decision tree produced by VIPER. It has 7 nodes and uses two features of the state space: θ and ω. If the pole is angled to the left, we move right if the pole is rotating right at a high velocity, else we move left. If the pole is angled to the right, we move right if the pole is rotating left at a high velocity, else we move right. The structure of the decision tree allows us to easily see the symmetric nature of the environment. Let us focus on the left half of the tree, corresponding to the situation where the pole is angled to the left. Doing this allows us to gain the following insight: we need to move left to catch the pole, and to get it to move back to the right. However, once the pole is moving to the right, even before it is angled to the right, we need to move the cart back to the right to catch the pole once its center of mass tips to the right.

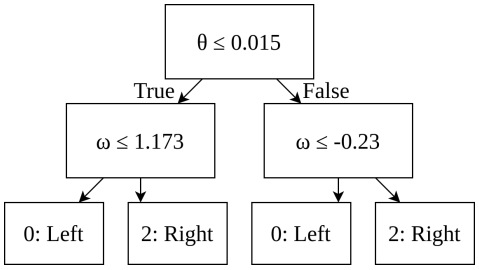


Figure 3: Decision tree that solves Cartpole.

Acrobot

Acrobot is, in essence, a double pendulum. A double pendulum is a notoriously chaotic system, and this makes the tree much harder to think and reason about than the trees for Cartpole or Mountaincar. It is helpful to think of the Acrobot as a gymnast hanging from a bar. The lower link is the legs, the actuated middle joint is the hips, and the upper link is the arms and torso.

Figure 4 shows the decision tree produced by VIPER, including the sample count for each leaf node. It has 7 nodes and, like Cartpole uses, only two features of the state space: $\dot{\theta}_1$ and $\dot{\theta}_2$.

The tree solves the environment in a surprisingly straightforward way. Note that the middle two leaf notes have a significantly higher sample count compared to the leftmost and rightmost nodes. Essentially, these middle nodes represent the 'kicking' motion needed to build momentum. If the lower link is swinging to the right, we also kick to the right, until gravity pulls us back down. Then we kick the other way. Once we have built up enough momentum, we let this momentum carry us over the line. The tree is unable to learn a clear strategy for getting the lower link up. Essentially, it just builds up significant momentum, and then trusts in luck and random chance to swing the lower link high enough[2]. This is also reflected in the high standard deviation.

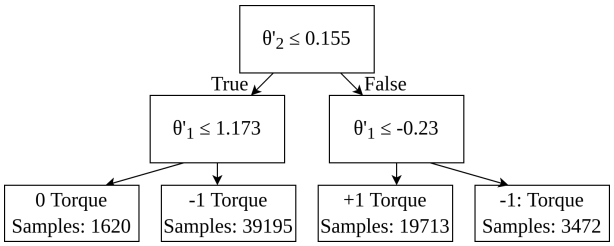


Figure 4: Decision tree that solves Acrobot.

5.3 Dependence on oracle quality

The quality of π naturally depends on the quality of π^*. Table 2 shows results for policies, trained on a variety of different oracles. As well as the DQL oracle we have seen in table 1 it also includes three different Q-LEARNING oracles, that achieve a variety of different rewards. They are labeled respectively DQ1, Q1, Q2, and Q3 for convenience.

There are three important observations to make. Firstly, a higher performing oracle does not necessarily allow for extracting a better policy. Secondly, oracles Q1 and Q2 achieve very similar performance, but the extracted policies differ significantly in performance quality. Lastly, the policy extracted from the lowest performing oracle, Q1, achieves by far the highest average reward, but also with a higher standard deviation. It outperforms the policy extracted from the DQL oracle and – recalling that the environment is considered solved if we achieve a reward larger than -110 for 100 consecutive episodes – is the only policy to solve the environment. This shows that other aspects of the oracle play an important role in the quality of the extracted policies.

[2]This can be verified: letting the tree take a random action in the leftmost and rightmost leafs has no significant impact on tree performance: it achieves an average reward of -86.3 ± 18.2, an improvement of only 0.11σ

Oracle	π^*	π
DQ	-112.1 ± 1.8	-111.2 ± 2.7
Q1	$\mathbf{-142.4 \pm 23.6}$	$\mathbf{-104.85 \pm 9.1}$
Q2	-137.7 ± 23.3	-119.6 ± 3.6
Q3	-129.4 ± 25.3	-116.6 ± 0.7

Table 2: Average rewards and standard deviations over 500 rollouts on Mountaincar of 4 different oracles: one DQL oracle and three different Q-LEARNING oracles π^*, and the corresponding policy π extracted by VIPER. The row in bold is the row with the biggest difference in oracle and student performance.

5.4 Comparison to AggreVaTe and Gail

We use the analyses provided by Wols and Lukina [28] and Meijer and Lukina [15] to compare the performance of respectively AGGREVATE and GAIL to that of VIPER. Let us first note a couple of important points to put this comparison into context. Firstly, Meijer and Lukina [15] use a modified version of GAIL that can extract decision trees. Secondly, as we have seen, imitation learning performance is highly dependent on oracle quality. Our oracles consistently perform just as well as Cartpole, and better on Mountaincar and Acrobot, though we have also seen that this does not necessarily lead to better surrogate model performance.

Mountaincar

On Mountaincar, performance of all three algorithms is equal for trees up to 7 nodes, all achieving a reward of approximately -119, and all learning the exact same tree with a single split on $v <= 0.0$. Obviously, these trees are also equally interpretable. Both AGGREVATE and GAIL are unable to learn trees that perform better then this, whereas VIPER improves on this performance significantly when training (slightly) larger trees (see table 1).

Cartpole

Both our oracle and the oracle used by Meijer and Lukina [15] achieve the maximum reward, but only VIPER is able to match that performance at any depth. GAIL is able to learn a tree with 11 nodes that achieves a reward of 498 ± 14.116. In contrast, we have seen that VIPER is able to learn a tree that achieves a perfect reward with only 7 nodes. However, it is important to note that our oracle was much more stable: it kept the cart in the middle of the environment, whereas the oracle used by Meijer and Lukina [15] traversed the entire horizontal space. AGGREVATE is also able to match its oracle, though the oracle used performs worse. While the tree produced by AGGREVATE is slightly smaller, using only 5 nodes, it is unable to fully solve the environment and is missing the inherent symmetry that the tree produced by VIPER has.

Acrobot

On Acrobot, GAIL and VIPER perform very similarly, achieving a reward that is almost equal. The tree AG-

GREVATE produces is extracted from an oracle that performs much worse – achieving a reward of only -237.3 – and the performance comparison is therefore irrelevant. The tree produced by GAIL is smaller, though, requiring a tree of only 3 nodes, where VIPER requires 7 nodes to achieve this performance.

6 Discussion

Interpretability remains highly dependent on context and problem complexity. The fact that the evaluated environments can be solved with small decision trees contributes massively to the fact that these trees remain interpretable. The trees learned with behavioral cloning are similarly interpretable, although for Acrobot and Cartpole they do use more features. Just as important to interpretability is the problem context. The chaotic nature of Acrobot's double pendulum and the chance based approach the policy takes make it harder to reason about the consequences of actions and therefore also to interpret π.

We have used the interpretability of these models to understand the control policies, and have managed to improve the Mountaincar policy using this interpretation. This shows how interpretable the models are. It is interesting that VIPER is unable to learn this improvement by itself, though this is almost certainly a consequence of oracle quality.

We rely on the inherent interpretability of decision trees to learn interpretable models. While VIPER only extracts decision trees, the more general Q-DAGGER[5] can be used to extract any surrogate model, like a neural network or a linear regression model, using the same 'accuracy on critical states' approach.

Additionally, we have shown that the performance of VIPER is dependent on oracle quality, and that oracle quality cannot be expressed solely in terms of the performance the oracle achieves. In VIPER, we let the decision tree explore the state space, and then query the oracle for supervision. This means that the oracle needs to know how to solve the environment from the starting position(s), but also to be able to recover from suboptimal states the decision tree might put it in. Simply put, the oracle needs to know the optimal action in as many states as possible, not only in the states it visits during a normal run. This dependency should be investigated further. What oracle characteristics play an important role in VIPER performance? Can we improve VIPER based on knowledge of these characteristics?

Finally, we have compared our results to similar experiments using imitation learning algorithms AGGREVATE and GAIL. While it may seem from our results that VIPER tends to perform better and produce smaller decision trees, it is important to note that the aforementioned dependence on oracle quality plays an important role here, making it harder to make a fair comparison. Still, VIPER's emphasis on critical state accuracy is a sensible approach to learning interpretable models.

7 Conclusion

We have demonstrated that VIPER is capable of producing decision trees that perform well and are highly interpretable for simple environments. These surrogate models perform better than or just as good as the oracle models they are trained from and are much more interpretable. The fact that VIPER produces small decision trees, because of its emphasis on accuracy in critical states, contributes to the interpretability of these trees. For all environments, we have shown that we can understand how the resulting policy solves the environment, and that this understanding can also help us gain a deeper understanding of the environment itself. Because VIPER also achieves a high fidelity, we can use these interpretations as a proxy to interpret the oracle policies themselves. We have also seen that performance of VIPER – and likely of all imitation learning approaches – is highly dependent on the quality oracle the oracle used, and that this quality cannot be expressed only in terms of performance. We also conclude that VIPER performs well across the environments, compared to other imitation learning algorithms, such as AGGREVATE and GAIL. VIPER is the only algorithm of these three that matches or outperforms its oracle on all three environments. The trees it needs to achieve these results are, with one exception, smaller than or equal in size to the trees produced by AGGREVATE and GAIL. Still, the differences in the oracles used significantly reduce the value of this comparison.

Future work

An important research direction to pursue is to compare imitation learning approaches, such as VIPER, AGGREVATE and GAIL, using a unified set of oracles. Additionally, it would be interesting to investigate oracles produced by reinforcement learning approach other than DQL, like PPO, DDPG or TRPO, to investigate what oracle qualities are important for effective imitation learning.

Another direction is the evaluation of the performance and interpretability of models produced by VIPER on more complex environments. In more complex environments, VIPER's emphasis on critical state accuracy might contribute more significantly to the interpretability of the models. Another research direction to pursue is to see if this approach facilitates interpretability, not only in decision trees, but also in other types of models.

Lastly, a final research direction is to see how we can improve VIPER, using the insights gained in this research, to train more interpretable decision trees. VIPER uses CART to train decision trees, but we can probably do better than this. Decision trees can often be simplified after training, without loss of accuracy but with an obvious gain in interpretability, using pruning methods like cost-complexity pruning [19]. Wols and Lukina [28] has already shown that pruning is an effective method when training decision trees with AGGREVATE. VIPER selects the tree using cross validation, taking into account only the performance of the tree. It would be interesting to explore if we can automatically select

more interpretable trees, or programmatically evaluate the performance-interpretability trade-off. However, to evaluate this, more complex benchmarks are most likely needed, as well as a quantitative metric of interpretability.

8 Responsible Research

Let us first start by noting that the methods used in this research do not use human subjects. This means that questions of human research ethics – the questions that most people will think about first when talking about 'ethical' science – are not relevant here. Nevertheless, there are still many pitfalls in terms of research integrity and reproducibility that are worthy of detailed analysis. To facilitate the analysis of ethical aspects of this research and reproducibility of the methods in a structured way, we reflect on each of the guiding principles of the Netherlands Code of Conduct for Research Integrity [2]: honesty, scrupulousness, transparency, independence, and responsibility. In addition, we discuss the reproducibility of the research.

Honesty

We have done our best to report the research process, as well as the results, accurately. No data in this text are falsified, and there is no fabricated material. During the research, we have had to find a balance between not reporting data that are not relevant – for instance, because the experiment wasn't tuned well yet – and reporting data that is not as expected: for example, the (for us) surprising non-linear relation between the performances of π^* and π. We believe that, after careful consideration, we have managed to find a reasonable balance, and have done justice to *all* the relevant results. Additionally, we have tried to be explicit about the uncertainties. Because the environments used in the research are simple, it is not possible to draw general conclusions, especially not about more complex scenarios. We have tried to be clear and explicit about this when drawing conclusions.

Scrupulousness

Because we were far from experts on the subject (and still are, though we have learned much), ensuring that our research is carefully and properly designed has been a challenge. In addition to our own ideas and critical thinking, we have taken both the opinion of all co-authors and the approaches taken by Bastani, Pu, and Solar-Lezama [5] and Ross, Gordon, and Bagnell [22] into careful consideration when designing this study. Using this approach, we are reasonably certain that the methods we have used are justified. Additionally, we have tried to ensure that our writing and reporting of the data is clear, and covers all the relevant details, though we are convinced we have much to learn in this area.

Transparency

We have always been very critical of our writing. We believe that our writing can be substantially improved, and with every rewrite, we improve a little. One of the goals in those rewrites is to make the text as clear as possible, to ensure that it is clear to others how the research was performed, why certain choices were made and how results were achieved. As always, we would have liked to spend more time on this. Still, we think that we have managed to make the line of reasoning reasonably clear.

Lastly, it is important to mention that there is no need to keep any data private, and therefore all data has been made public. The code used to run the experiment is also public, and this allows for an even more transparent result.

Independence

It should be clear to everyone that there are no influences of a commercial or political nature. There is, however, one clear non-scientific and non-scholarly consideration that may be, if left unchecked, of an influence to this research: the fact that this is a course, for which we will get a grade. This fact could hypothetically lead to the temptation to fabricate or alter data, to create a better story. Apart from the much more important and obvious reason that we are trying to perform honest research, we are also conscious of the fact that 'better' results will not lead to a higher grade. We strongly believe that we have not let this influence our choices of design, conduct, and reporting of this research in a way that is detrimental to the science in this paper.

It is however clear that the educational nature of this project has influenced the design and conduct of this research to some extent: for example, the scope of the research, the length of this text, the time allotted to performing this research and many other aspects are influenced by the fact that this research has been performed for a course.

Responsibility

The field of interpretable AI has significant social and scientific relevance and value. We believe that, with the increase in the use and influence of AI, interpretability is a very important consideration. We are glad that we have managed to learn about this, and have been able to make a contribution to it, however small.

Reproducibility

This research uses freely available, commonly used environments [7] and oracles [10, 20]. The experimental setup is clearly explained, and the algorithms it uses that are not from an open library have been discussed in this text and cited. The code used to tie it all together is also public. Because of this, the results should be relatively easy to reproduce.

References

[1] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 1.

[2] Royal Netherlands Academy of Arts and Sciences. "Netherlands Code of Conduct for Research Integrity". In: (2018). URL: https://www.nwo.nl/sites/nwo/files/documents/Netherlands%20Code%20of%20Conduct%20for%20Research%20Integrity_2018_UK.pdf.

[3] Christopher G Atkeson and Stefan Schaal. "Robot learning from demonstration". In: *ICML*. Vol. 97. Citeseer. 1997, pp. 12–20.

[4] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. "Interpretability via model extraction". In: *arXiv preprint arXiv:1706.09773* (2017).

[5] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. "Verifiable reinforcement learning via policy extraction". In: *arXiv preprint arXiv:1805.08328* (2018).

[6] Leo Breiman et al. *Classification and regression trees*. Routledge, 2017.

[7] Greg Brockman et al. "OpenAI Gym". In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

[8] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. "Machine learning interpretability: A survey on methods and metrics". In: *Electronics* 8.8 (2019), p. 832.

[9] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

[10] Ashley Hill et al. *Stable Baselines*. https://github.com/hill-a/stable-baselines. 2018.

[11] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf.

[12] Ahmed Hussein et al. "Imitation learning: A survey of learning methods". In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–35.

[13] Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva. "On explaining decision trees". In: *arXiv preprint arXiv:2010.11034* (2020).

[14] Yunzhu Li, Jiaming Song, and Stefano Ermon. "Infogail: Interpretable imitation learning from visual demonstrations". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 3815–3825.

[15] Caspar Meijer and Anna Lukina. "Towards a rigorous science of interpretable machine learning". In: *TU Delft CSE3000* (2021).

[16] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[17] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[18] Erika Puiutta and Eric MSP Veith. "Explainable reinforcement learning: A survey". In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer. 2020, pp. 77–95.

[19] J.R. Quinlan. "Simplifying decision trees". In: *International Journal of Man-Machine Studies* 27.3 (1987), pp. 221–234. ISSN: 0020-7373. DOI: https://doi.org/10.1016/S0020-7373(87)80053-6. URL: https://www.sciencedirect.com/science/article/pii/S0020737387800536.

[20] Antonin Raffin. *RL Baselines Zoo: a Collection of Pre-Trained Reinforcement Learning Agents*. https://github.com/araffin/rl-baselines-zoo. 2018.

[21] Stephane Ross and J Andrew Bagnell. "Reinforcement and imitation learning via interactive no-regret learning". In: *arXiv preprint arXiv:1406.5979* (2014).

[22] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[23] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[24] Philipp Schmidt and Felix Biessmann. *Quantifying Interpretability and Trust in Machine Learning Systems*. 2019. arXiv: 1901.08558 [cs.LG].

[25] Kacper Sokol and Peter Flach. "Explainability Fact Sheets: A Framework for Systematic Assessment of Explainable Approaches". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* '20. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 56–67. ISBN: 9781450369367. DOI: 10.1145/3351095.3372870. URL: https://doi.org/10.1145/3351095.3372870.

[26] Gilles Vandewiele et al. *GENESIM: genetic extraction of a single, interpretable model*. 2016. arXiv: 1611.05722 [stat.ML].

[27] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[28] Jonathan Wols and Anna Lukina. "Towards a rigorous science of interpretable machine learning". In: *TU Delft CSE3000* (2021).

[29] He Zhu et al. "An inductive synthesis framework for verifiable reinforcement learning". In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2019, pp. 686–701.