Modelling Agent Policies with Interpretable Imitation Learning

Tom Bewley, Jonathan Lawry and Arthur Richards

March 14, 2020

Modelling Agent Policies with Interpretable Imitation Learning[⋆]

Tom Bewley[0000−0002−5460−0744], Jonathan Lawry[0000−0002−3488−5682], and Arthur Richards[0000−0001−9500−5514]

University of Bristol, Bristol, UK
{tom.bewley,j.lawry,arthur.richards}@bristol.ac.uk

Abstract. As we deploy autonomous agents in safety-critical domains, it becomes important to develop an understanding of their internal mechanisms and representations. We outline an approach to imitation learning for reverse-engineering black box agent policies in MDP environments, yielding simplified, interpretable models in the form of decision trees. As part of this process, we explicitly model and learn agents' latent state representations by selecting from a large space of candidate features constructed from the Markov state. We present initial promising results from an implementation in a multi-agent traffic environment.

Keywords: explainable artificial intelligence · interpretability · imitation learning · representation learning · decision tree · traffic modelling.

1 Introduction

Data-driven learning is state-of-the-art in many domains of artificial intelligence (AI), but raw statistical performance is secondary to the trust, understanding and safety of humans. For autonomous agents to be deployed at scale in the real world, people with a range of backgrounds and remits must be equipped with robust mental models of their learning and reasoning. However, modern learning algorithms involve complex feedback loops and lack semantic grounding, rendering them black boxes from a human perspective. The field of explainable artificial intelligence (XAI) [5] has emerged in response to this challenge.

Most work in XAI focusses on developing insight into classification and regression systems trained on static datasets. In this work we consider dynamic problems comprising agents interacting with their environments. We present our approach to interpretable imitation learning (I2L), which aims to model the policy of a black box agent from analysis of its input-output statistics. We call the policy model *interpretable* because it takes the form of a binary decision tree, which is easily decomposed and visualised, and can be used for both factual and counterfactual explanation [3]. We move beyond most current work in the imitation learning literature by explicitly learning a latent state representation used by the agent as the basis for its decision making. After formalising our approach, we report the initial results of an implementation in a traffic simulator.

[⋆] Supported by an EPSRC/Thales industrial CASE award in autonomous systems.

2 I2L Framework

Preliminaries Our I2L approach is applied to agents that operate in Markov Decision Process (MDP) environments. At time t, an agent perceives the current Markov state $s_t \in \mathcal{S}$. We then assume that it maps this state into an intermediate representation $x_t = \phi(s_t)$ then selects an action from the discrete space $\mathcal{A}$ via a deterministic policy function $a_t = \pi(x_t)$. The next Markov state s_{t+1} is a function of both s_t and a_t. Modelling a state-dependent reward function is not necessary for the present work.

Generic Problem Formulation We adopt the perspective of a passive spectator of the MDP with access to its Markov state and the action taken by the agent at each time step. This allows us to observe a history of N states and actions $\mathcal{H} = [(s_1, a_1), ..., (s_N, a_N)]$ to use as the basis of learning. Our objective is to imitate the agent's behaviour, that is, to reverse-engineer the mapping $s_t \rightarrow a_t$. This effectively requires approximations of both ϕ and π, denoted by ϕ' and π' respectively. The need to infer both the policy and the representation on which this policy is based makes this problem a hybrid of imitation learning and state representation learning [4].

It is essential to constrain the search spaces for ϕ' and π' (Φ and Π respectively) so that they only contain functions that are human-interpretable, meaning that their operation can be understood and predicted via visualisations, natural-language explanations or brief statements of formal logic. This property must be achieved while minimally sacrificing imitation quality or tractability of the I2L problem. Given the history of state-action pairs $\mathcal{H}$, this problem can be formulated as an optimisation over ϕ' and π':

$$\underset{\phi' \in \Phi, \pi' \in \Pi}{\text{argmin}} \left[\sum_{t=1}^{N} \ell\left(\pi'(\phi'(s_t)), a_t\right) \right] \quad \text{where} \quad \Phi, \Pi = \text{``\textit{interpretable}''} \tag{1}$$

and $\ell : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$ is a pairwise loss function over the discrete action space. The schematic in figure 1[1] outlines the task at hand.

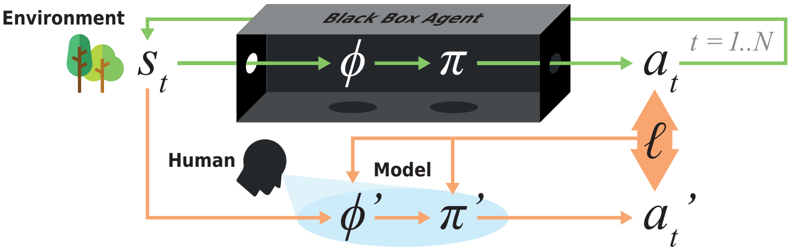


Fig. 1. Generic I2L problem setup. The objective is to minimise the loss between a_t and a'_t $\forall t \in 1..N$, while ensuring ϕ' and π' are comprehensible under human scrutiny.

[1] Icons from users *Freepik* and *Pixel Perfect* at www.flaticon.com.

Space of Representation Functions As highlighted above, it is important that Φ permits human-interpretable state representations. We achieve this by limiting its codomain to vectors of real-valued features, generated from s_t through recursive application of elementary operations from a finite set $\mathbb{F}$. To limit the complexity of Φ, we specify a limit r on the recursion depth. Prior domain knowledge is used to design $\mathbb{F}$, which in turn has a shaping effect on ϕ' without requiring the stronger assumption of which precise features to use. Each feature also has a clear interpretation by virtue of its derivation via canonical operations. In our traffic simulator implementation, $\mathbb{F}$ contains operations to extract a vehicle's speed/position, find the nearest vehicle/junction ahead of/behind a position, and subtract one speed/position from another.

Space of Policy Functions Π must be similarly limited to functions that are comprehensible to humans while retaining the representational capacity for high-quality imitation. Following prior work [2,6], we achieve this by adopting a decision tree structure. The pairwise loss function ℓ over the action space $\mathcal{A}$ is used to define an impurity measure for greedy tree induction. For a decision node n, let $P_n(a)$ be the proportion of data instances at that node with action value a. We use the following measure of node impurity:

$$I(n) = \sum_{a\in\mathcal{A}} \sum_{a'\in\mathcal{A}} P_n(a) \cdot P_n(a') \cdot \ell(a, a') \tag{2}$$

The popular Gini impurity is a special case of this measure, recovered by defining ℓ such that $\ell(a, a') = 0$ if $a = a'$, and $\ell(a, a') = 1$ otherwise.

Learning Procedure In general, the joint inference of two unknown functions (in this case ϕ' and π') can be very challenging, but our constraints on Φ and Π allow us to approximately solve equation 1 through a sequential procedure:

1. Apply domain knowledge to specify the feature-generating operations $\mathbb{F}$ and recursion depth r. Denote the representation of all valid features ϕ_{all}.
2. Iterating through each state s_t in the observed state-action history $\mathcal{H}$, apply ϕ_{all} to generate a vector of numerical feature values. Store this alongside the corresponding action a_t in a training dataset.
3. Define a pairwise action loss function ℓ and deploy a slightly-modified version of the CART tree induction algorithm [1] to greedily minimise the associated impurity measure (equation 2) on the training set. Let the induction process continue until every leaf node is pure, yielding a large, overfitted tree $\mathcal{T}_0$.
4. Prune the tree back using minimal cost complexity pruning (MCCP) [1], whose output is a tree sequence $[\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, ...]$, each a subtree of the last, representing a progressive reduction of $\mathcal{T}_0$ down to its root.
5. Pick a tree from this sequence to use as the policy model π', and define ϕ' as the subset of features from ϕ_{all} used at least once in that tree.

Having a sequence of options for the tree model and associated representation allows us to manage a tradeoff between accuracy on one end, and interpretability (through simplicity) on the other. In the following implementation, we explore the tradeoff by selecting several trees from across the pruning sequence.

3 Implementation with a Traffic Simulator

We implement I2L in a traffic simulator, in which multiple vehicles follow a common policy to navigate a track while avoiding collisions. Five track topologies are shown in figure 3 (left). Coloured rectangles are vehicles and red circles are junctions. Since the policy is homogeneous across the population we can analyse the behaviour of all vehicles equally to learn ϕ' and π'.

Target Policies Instead of learned policies, we use two hand-coded controllers as the targets of I2L. From the perspective of learning these policies remain opaque. For both policies, $\mathcal{A}$ contains five discrete acceleration levels, which determine the vehicle's speed for the next timestep within the limits $[0, v_{\max}]$.

- Fully-imitable policy (π_{F}): a rule set based on six features including the vehicle's speed, distance to the next junction, and separations and speeds relative to other agents approaching that junction. This policy is itself written as a decision tree with 39 leaves, hence can be modelled exactly in principle.
- Partially-imitable policy (π_{P}): retains some logic from π_{F}, but considers more nearby vehicles and incorporates a proportional feedback controller. These changes cannot be modelled exactly by a finite decision tree.

Representation We use a set of eight operations $\mathbb{F}$ which allows the six-feature representation used by π_{F} (denoted ϕ_{F}) to be generated. These are:

- pos : $i \rightarrow p$ or $j \rightarrow p$. Get the position of vehicle i or junction j.
- speed : $i \rightarrow v$. Get the speed of vehicle i.
- fj : $i \rightarrow j$. Find the next junction in front of vehicle i.
- fa : $i \rightarrow i'$ or $j \rightarrow i'$. Find the next vehicle in front of vehicle i or junction j.
- ba : $i \rightarrow i'$ or $j \rightarrow i'$. Find the next vehicle behind vehicle i or junction j.
- twin : $j \rightarrow j'$. Flip between the two track positions comprising a junction.
- sep : $(p_1, p_2) \rightarrow s$. Compute the separation between two positions.
- sub : $(v_1, v_2) \rightarrow \Delta$ or $(s_1, s_2) \rightarrow \Delta$. Subtract two speeds or separations.

With $r = 6$, ϕ_{all} has 308 features, including all six in ϕ_{F}. The vast majority are irrelevant for imitating the target policies, so we face a feature selection problem.

Training We run simulations with 11 vehicles on all five topologies in figure 3 (left). The recorded history $\mathcal{H}$ is converted into a training dataset by applying ϕ_{all} for each vehicle. After rebalancing action classes, our final dataset has a length of 125000. For tree induction, we use the simple loss function $\ell(a, a') = |a - a'|$. For each tree in the sequence output by MCCP, we measure the predictive accuracy on a validation dataset, and consider the number of leaf nodes and features used as heuristics for interpretability. Figure 2 plots these values across the sequence for both targets. We select five pruning levels (in addition to $\mathcal{T}_0$) for evaluation.

Baselines We also train one tree using only the six features in ϕ_{F}, and another with an alternative representation $\phi_{\text{naïve}}$, indicative of what might be deemed useful without domain knowledge (radius, angle, normal velocity and heading of nearby agents in egocentric coordinates). Two further baselines use training data from only one track topology (either the smallest topology A or the largest D). This tests the generalisability of single-topology learning. For all baselines, we use the pruned tree with the highest validation accuracy.

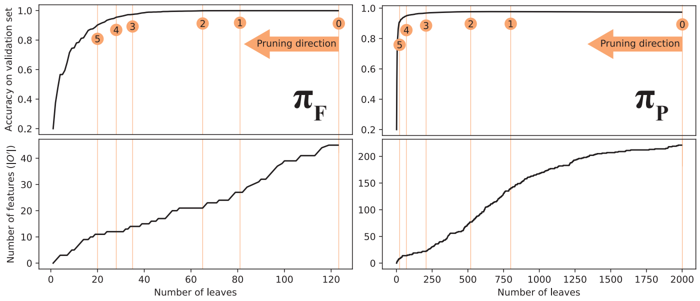
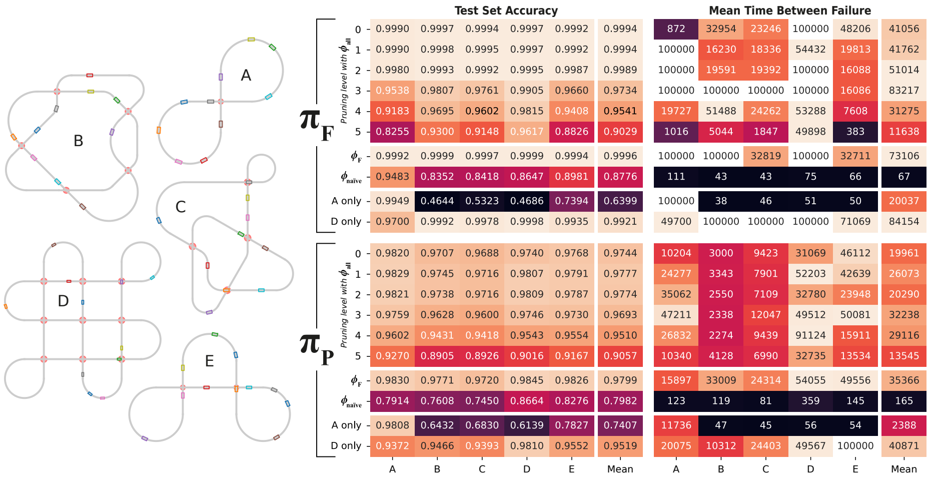


Fig. 2. Validation accuracy and used feature count across the pruning sequences for each target policy. **Note the very different horizontal axis scales**; the unpruned tree for π_P is much larger. Numbered vertical lines indicate trees chosen for evaluation.

4 Results and Discussion

The heat maps in figure 3 (right) contain results for two metrics of imitation quality for both π_F and π_P. Each row corresponds to a pruned tree or baseline, and each column indicates the track topology used for testing.



		Test Set Accuracy						Mean Time Between Failure					
		A	B	C	D	E	Mean	A	B	C	D	E	Mean
π_F	0	0.9990	0.9997	0.9994	0.9997	0.9992	0.9994	872	32954	23246	100000	48206	41056
	1	0.9990	0.9998	0.9995	0.9997	0.9992	0.9994	100000	16230	18336	54432	19813	41762
	2	0.9980	0.9993	0.9992	0.9995	0.9987	0.9989	100000	19591	19392	100000	16088	51014
	3	0.9538	0.9807	0.9761	0.9905	0.9660	0.9734	100000	100000	100000	100000	16086	83217
	4	0.9183	0.9695	0.9602	0.9815	0.9408	0.9541	19727	51488	24262	53288	7608	31275
	5	0.8255	0.9300	0.9148	0.9617	0.8826	0.9029	1016	5044	1847	49898	383	11638
	ϕ_F	0.9992	0.9999	0.9997	0.9999	0.9994	0.9996	100000	100000	32819	100000	32711	73106
	$\phi_{naïve}$	0.9483	0.8352	0.8418	0.8647	0.8981	0.8776	111	43	43	75	66	67
	A only	0.9949	0.4644	0.5323	0.4686	0.7394	0.6399	100000	38	46	51	50	20037
	D only	0.9700	0.9992	0.9978	0.9998	0.9935	0.9921	49700	100000	100000	100000	71069	84154
π_P	0	0.9820	0.9707	0.9688	0.9740	0.9768	0.9744	10204	3000	9423	31069	46112	19961
	1	0.9829	0.9745	0.9716	0.9807	0.9791	0.9777	24277	3343	7901	52203	42639	26073
	2	0.9821	0.9738	0.9716	0.9809	0.9787	0.9774	35062	2550	7109	32780	23948	20290
	3	0.9759	0.9628	0.9600	0.9746	0.9730	0.9693	47211	2338	12047	49512	50081	32238
	4	0.9602	0.9431	0.9418	0.9543	0.9554	0.9510	26832	2274	9439	91124	15911	29116
	5	0.9270	0.8905	0.8926	0.9016	0.9167	0.9057	10340	4128	6990	32735	13534	13545
	ϕ_F	0.9830	0.9771	0.9720	0.9845	0.9826	0.9799	15897	33009	24314	54055	49556	35366
	$\phi_{naïve}$	0.7914	0.7608	0.7450	0.8664	0.8276	0.7982	123	119	81	359	145	165
	A only	0.9808	0.6432	0.6830	0.6139	0.7827	0.7407	11736	47	45	56	54	2388
	D only	0.9372	0.9466	0.9393	0.9810	0.9552	0.9519	20075	10312	24403	49567	100000	40871

Fig. 3. (Left): The five test topologies used. (Right): Results for I2L quality metrics.

Accuracy This metric is predictive accuracy on a held-out test set. For both π_F and π_P, mean accuracy exceeds 90% for even the smallest prune levels and 95% for the second-smallest. The fact that accuracy for π_P is bounded at around 97.5% reflects the fact that this policy is not a decision tree so cannot be perfectly imitated by one. As expected, providing ϕ_F upfront (thereby removing the representation learning requirement) yields somewhat better accuracy, but it is promising to see that down to prune level 2, performance differs by under 0.25% for both targets. Lacking information about junctions, the tree using $\phi_{\text{naïve}}$ is unable to obtain the same levels of accuracy, demonstrating the importance of choosing the correct representation for imitation. The single-topology training results show that trees generalise well from the large topology D, but poorly from the small topology A. This suggests the latter contains insufficient variety of vehicle arrangements to capture all important aspects of the target policies.

Failure Here we deploy the models as control policies in the environment and measure the mean time between failures (either collision or ‘stalls’, when traffic flow grinds to a halt) over 100 episodes of up to 1000 timesteps. A value of 100000 indicates zero failures occurred. While results broadly correlate with accuracy, this metric shows a more marked aggregate distinction between π_{F} and π_{P}, and between different test topologies. Nonetheless, there is minimal degradation with pruning down to level 4, with a constant average of just 1-2 failures for π_{F}, and 3-4 for π_{P}. In fact, it appears that intermediate pruning levels fail less often than the largest trees. While the reason for this is not immediately clear, it may be that having fewer leaves yields less frequent changes of acceleration and smoother motion. Providing ϕ_{F} confers no significant benefit over ϕ_{all}, while $\phi_{\mathrm{naïve}}$ and single-topology training on A are utterly unable to perform/generalise well.

5 Conclusion

We have introduced our approach to interpretable imitation learning for black box agent control policies that use intermediate low-dimensional state representations. Our models take the form of decision trees, which select from large vectors of candidate features generated from the Markov state using a set of basic operations. The accuracy-interpretability tradeoff is managed by post-pruning.

Our initial implementation has shown that trees trained by I2L exhibit high predictive accuracy with respect to two hand-coded control policies, and are able to avoid failure conditions for extended periods, even when heavily pruned. It has also highlighted that using a plausible-but-incorrect state representation places a severe limitation on imitation quality, and that learning from data that do not capture the full variation of the environment leads to poor generalisation.

In ongoing follow-up work, we are exploring how our decision tree models can be used to interpret and explain their target policies, and are also implementing I2L with a truly black box policy trained by reinforcement learning.

References

1. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees. Wadsworth & Brooks. Cole Statistics/Probability Series (1984)
2. Coppens, Y., Efthymiadis, K., Lenaerts, T., Nowé, A., Miller, T., Weber, R., Magazzeni, D.: Distilling deep reinforcement learning policies in soft decision trees. In: Proceedings of the IJCAI 2019 Workshop on Explainable AI. pp. 1–6 (2019)
3. Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., Turini, F.: Factual and counterfactual explanations for black box decision making. IEEE Intelligent Systems (2019)
4. Lesort, T., Díaz-Rodríguez, N., Goudou, J.F., Filliat, D.: State representation learning for control: An overview. Neural Networks **108**, 379–392 (2018)
5. Samek, W., Wiegand, T., Müller, K.R.: Explainable AI: Understanding, visualizing and interpreting deep learning models. arXiv:1708.08296 (2017)
6. Turnbull, O., Lawry, J., Lowenberg, M., Richards, A.: A cloned linguistic decision tree controller for real-time path planning in hostile environments. Fuzzy Sets and Systems **293**, 1–29 (2016)