Synthesising Reinforcement Learning Policies through Set-Valued Inductive Rule Learning*

Youri Coppens[1,2][0000−0003−1124−0731], Denis Steckelmacher[1][0000−0003−1521−8494], Catholijn M. Jonker[3,4][0000−0003−4780−7461], and Ann Nowé[1][0000−0001−6346−4564]

[1] Vrije Universiteit Brussel, Brussels, Belgium
[2] Université Libre de Bruxelles, Brussels, Belgium
[3] Delft University of Technology, Delft, The Netherlands
[4] Leiden Institute of Advanced Computer Science (LIACS), Leiden, The Netherlands
yocoppen@ai.vub.ac.be

Abstract. Today's Deep Reinforcement Learning algorithms produce black-box policies, that are often difficult to interpret and trust for a person. We introduce a policy distilling algorithm, based on the CN2 rule mining algorithm, that distills the deep policy into a rule-based decision system. At the core of our approach is the fact that an RL process does not just learn a policy, a mapping from states to individual actions, but also produces extra meta-information, such as lists of visited states, or action probabilities. This meta-information can, for example, indicate whether more than one action is near-optimal for a certain state. We exploit knowledge about these equally-good actions to distill the policy into fewer rules, which contributes to interpretability, while ensuring that the performance of the distilled policy still matches the original policy. This ensures that we don't provide an explanation for a degenerate or over-simplified policy. We demonstrate the applicability of our algorithm to the Mario AI benchmark, a complex task that requires modern deep reinforcement learning algorithms. The explanations we produce capture the learned policy in only a few rules, and can be further refined and tailored by the user with a two-step process that we introduce in this paper.

Keywords: Reinforcement Learning · Inductive Rule Learning · Explainable AI · Policy Distillation.

1 Introduction

Reinforcement Learning (RL) is a machine learning technique that learns from experience, rather than data that has been collected offline. By interacting with its environment, an intelligent agent learns to adapt its actions to show the

* This work was supported by the Research Foundation Flanders (FWO) [grant numbers G062819N and 1129319N], the AI Research Program from the Flemish Government (Belgium) and the Francqui Foundation.

desired behaviour. This environment is in general stochastic and its dynamics unknown to the agent.

RL is particularly useful in settings without a model of the environment, or when the environment is hard to model. RL is also interesting when an accurate simulation model can be built, but solving that model to obtain a decision or control strategy is hard. Telecommunication applications such as routing, load balancing and call admission control are examples where RL has proven its worth as a machine learning technique. RL has unfortunately one crucial weakness which may prevent it from being applied in future critical applications: the control policies learned by the agent, especially using modern algorithms based on neural networks, are black boxes that are almost impossible to explain. This shortcoming could prove a major obstacle to mainstream the adoption of RL. For example, the new EU privacy regulation, more specifically Article 22 of the General Data Protection Regulation (GDPR), requires that all AI with an impact on human lives needs to be accountable. Interpretability of machine learning based models is therefore crucial, as it forms the basis for other concerns such as accountability, that subsumes the justification of the decisions taken. Assume that RL is used to synthesise a prevention strategy for a dangerous infection disease. The agent should learn what regulations and procedures people should follow, while optimally allocating a budget, such as vaccines, medication or closed-school days. While this scenario is plausible, see [12], will a black-box strategy be accepted? The above observation brings us to the following key question: Wouldn't it make more sense to use RL to explore possible strategies and simulate the potential effects on our society, before we would decide to implement such a strategy? However, this requires that the strategies found by RL algorithms are presented in a human-understandable format.

Explainable AI, reviewed in [15], is however not new. Even in the early days of Expert Systems, many approaches have been proposed to allow to automatically generate explanations and to reflect on the reasoning being used, see e.g. [14]. Recently, explainability has been put more prominently on the agenda in the Machine Learning (ML) community as well [17]. The massive success of the black-box ML-techniques has culminated in many applications that display powerful results, but provide poor insight into how they find their conclusion. Over the last three years, workshops at major conferences in Machine Learning (e.g. ICML, NeurIPS), Artificial Intelligence (e.g. IJCAI), Human-Computer Interaction (e.g. ACM CHI, ACM IUI) and Planning (e.g. ICAPS) have been held for contributions that increase human interpretability of intelligent systems. At these workshops, initial research related to data mining has been proposed. However, ideas to make RL more explainable are still limited.

2 Related Work

Until now, insights in the learnt RL policy were typically limited to showing a typical trace, i.e. an execution of the policy or through analysing and visualising agent reward structures [1]. Relevant efforts to mention are [19], where the authors visualise action versus parameter exploration and [2] that analyses and visualises the usefulness of heuristic information in a state space.

Distillation techniques translate the behaviour from large complex models to simpler surrogate models [8]. While these methods reduce computational overhead and model complexity, the general focus is mostly set on maintaining performance rather than increasing the interpretability through the surrogate model, even in the context of RL policies [18]. In [4], a deep RL policy is distilled into a surrogate soft decision tree model [5], in an effort to gain insights in the RL policy. The surrogate model determines a hierarchy of filters that perform sequential feature extraction from environment observations. These filters highlight which features are considered important in the decision making. Our work is complementary in the sense that the rules we extract look at the content, i.e. the actual value of the observation features, rather than the importance of a particular feature.

Relational RL (RRL) could be seen as an interpretable RL approach *by design* [24] as it allows to express the policy as well as the quality of the policy using relational expressions. Relational RL requires domain-specific knowledge from the designer, as a set of concepts have to be defined *a priori*. If these concepts are not carefully defined, the expressiveness of the agent is reduced, which may prevent it from learning any good policy. In [26] relational inductive biases have been incorporated into a deep learning process, in an attempt to scale RRL without pre-defined concepts, thus allowing for relevant concepts to be learned. The approach is shown to reveal the high-level plan of the RL strategy, thus identifying subgoals and options that can be generalised. Our work does not consider that subgoals are available, and instead focuses on making the actual policy, the mapping from states to actions, transparent to the user.

The paper by Madumal et al. [13] introduces an approach to explain a policy which can either be obtained through model-based RL or planning. It builds upon causal graphs as a means to calculate counterfactuals. As the causal graph is assumed to be given, the symbolic concepts are also defined *a priori*, and have to be carefully chosen, leading to the same vulnerability as Relational RL.

In our approach, we don't want to restrict the policy *a priori*. We allow the RL process to learn complex policies if needed. Recent advances in (deep) function approximation for RL have yielded powerful techniques that learn rich state representations and have advanced state-of-the-art in terms of learning performance in complex environments, see e.g. [16,20]. However, these techniques obscure a proper understanding of the policy that is being learned.

In the approach we propose, we start from existing state-of-the-art RL techniques and make them more interpretable. Our work performs an *a posteriori* simplification and communication of the strategy. This is realised through a rule-mining approach that exploits the meta-information that is present in the RL learning algorithm. While decision trees are a more commonly adopted interpretable surrogate model that can be visualised in a natural manner, we opted for decision rules, because these have a similar expressiveness while also being more compact and robust. In addition, the use of meta-information, such as which actions are equally good in a state, is more naturally implemented in a rule-mining system, as well as the refinement process we propose. The RL process not only provides us with the best action for a given state, but the probability that an RL actor gives to each of the actions is providing us information on the quality of the sub-optimal actions. Exploiting this meta-information allows to go beyond a mere translation of the policy into a set of rules. As, for instance, knowing that two actions have nearly the same probability in a given state indicates that they are both *equally good*, which allows our rule mining algorithm to chose one action or the other, depending on what leads to the simplest rule set. This exploitation of the meta-information of an RL process also differentiates our work from approaches that translate the RL policy in a set of fuzzy rules, such as [9] and [7].

3 Background

Our work covers several domains of research such as Reinforcement Learning and Inductive Rule Learning.

3.1 Policies learnt through Reinforcement Learning

Reinforcement Learning (RL) tackles sequential decision making problems in an interactive setting. [22] An artificial agent learns by operating in an unknown environment, from scratch or boosted by some initial domain knowledge, which actions are optimal over time. Based on the observation of the state of the environment, the agent executes an action of choice which makes the agent transition to a next state. In addition to the new state, the agent will also receive feedback from the environment in the form of a numerical reward. Based on that feedback, the agent will adapt its behaviour into a desired strategy.

Reinforcement learning problems are formally represented as a Markov Decision Process (MDP), a 5-tuple (S, A, T, R, γ), with S and A the sets of states and actions available in the environment. Unknown to the agent are T and R. $T : S \times A \times S \to [0, 1]$ represents the transition probability function, assigning probabilities to a possible transition from state $s \in S$ to another state $s' \in S$, by executing action $a \in A$. $R : S \times A \to \mathbb{R}$ is the environment's reward function, providing the agent with feedback and finally a discount factor $\gamma \in [0, 1]$ regulates the agent's preference for immediate rewards over long-term rewards. The agent's goal is to maximise the expected return $\mathbb{E}[G_t]$, where $G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i}$, the sum of future discounted rewards. Lower values of γ set the preference on immediate reward, whilst higher values result in a more balanced weighing of current and future reward in G_t. The agent learns a policy $\pi : S \times A \to [0, 1]$, mapping state-action pairs to probabilities, describing the optimal behaviour strategy.

In value-based RL methods, this policy is implicitly present through the use of a *value function* in combination with an action-selection strategy. The value function $Q(s, a)$ is often represented as the expected return when taking an action a in state s: $Q(s,a) = \mathbb{E}[G_t | S_t = s, A_t = a]$, hence describing which actions are expected to result in more future rewards.

When the Q-values have converged, the highest ranked action is the optimal one, whilst others are granted to be sub-optimal. However, when the Q-values are equal or almost equal, this indicates that there is no clear preference over these actions.

Policy-based algorithms are another family of RL methods. They maintain an explicit policy, or *actor*, often described as a parametric expression π_θ. The policy is optimised iteratively, using methods such as computing and following the gradient of its quality [23], or training it to progressively imitate the greedy policy of a collection of critics [21]. Policy-based methods typically learn a stochastic policy where the difference in probability can be considered to be a proxy for the difference in quality of two alternative actions in a state. The BDPI algorithm [21] that we use in this paper, for instance, learns a policy that associates nearly-equal probability to actions that are of comparable quality, while actions that are clearly worse than the other ones have a probability near 0. In other words, when the probability of two actions in a given state is close to each other, this expresses there is no clear preference of one action over the other action. We exploit this property in our framework in order to find a small and simple set of rules capturing the main elements of the policy.

3.2 Inductive Rule Learning

Inductive rule learning allows us to extract knowledge in a symbolic representation from a set of instances in a supervised manner. The goal of supervised rule induction is the discovery of rules that reflect relevant dependencies between classes and attribute values describing the given data [6]. In our setting, these data instances are state-action pairs drawn from a learned RL policy. More precisely, the instances are pairs of states with *set-valued* actions: for a single instance, a state might be paired with multiple actions.

The set-valued aspect is important to express that two or more actions are high-quality alternatives in a certain state. By supporting the possibility to express this indifference towards which action of that set is selected in the corresponding state, we can create rules with a larger support and as a consequence fewer rules. Regular inductive rule learning is not designed to deal with this set-valued aspect, and therefore we extend this learning paradigm in Section 4. Before we introduce our set-valued rule mining, we first summarise the main elements of inductive rule learning and the CN2 algorithm.

Inductive rule learning allows to summarise a set of instances, which might be discrete valued or continuous valued, along with a discrete target class, into a set of rules. The rules are of the form 'IF *Conditions* THEN c', where c is the class label. Because of their symbolic nature, the resulting model is easy to interpret by humans.

A common approach in inductive rule learning algorithms is to construct an unordered set or an ordered list of rules based on a *separate-and-conquer* principle. In each step, a rule is learnt which on the one hand tries to cover as many instances as possible, and on the other hand tries to be as accurate as possible, thus balancing coverage and accuracy. When such a rule is found, the training examples that are covered by this rule are removed and the process is repeated on the remaining data.

We base our work on CN2 [3], a frequently used rule learning algorithm based on the separate-and-conquer principle, and extend it to support multiple equally-good labels per input. In CN2, rules are iteratively identified through the application of beam search, a heuristic tree search algorithm. The best rule that is identified in this search procedure will be selected as new rule. The consequent (classifying label) of the rule is the majority class of the examples covered by the rule.

The commonly used evaluation heuristic in CN2 is *weighted relative accuracy* (WRA) [11,25], which aims to find a good balance between a large coverage and a high accuracy. Equation 1 describes the WRA of a proposed rule r:

$$\text{WRA}(r) = \frac{\hat{P}(r) + \hat{N}(r)}{P(r) + N(r)} \times \left(\frac{\hat{P}(r)}{\hat{P}(r) + \hat{N}(r)} - \frac{P(r)}{P(r) + N(r)} \right) \tag{1}$$

where $P(r)$ and $N(r)$ represent the number of positive and negative examples in the current training set and $\hat{P}(r)$ and $\hat{N}(r)$ represent the number of positive and negative examples covered by the considered rule r. From hereon, we will drop the parameter r as no confusion is to be expected. Examples are said to be positive when they have the same classification label as the consequent of the rule, otherwise they are considered to be negative. $\hat{P}$ and $\hat{N}$ can be thus considered the true and false positive examples covered by the rule respectively. The first factor in Equation 1, $\frac{\hat{P}+\hat{N}}{P+N}$, represents the *coverage* of the proposed rule, the rate of examples that satisfy the antecedent, and weighs the second factor of the equation. That factor, $\left(\frac{\hat{P}}{\hat{P}+\hat{N}} - \frac{P}{P+N}\right)$, represents the relative classification accuracy of a rule. It is the difference between the accuracy of the samples covered by the rule and the overall accuracy of the training examples.

The algorithm stops generating rules if there are no other significant rules to be found or an insufficient number of samples remains. These thresholds are expected to be application dependent and are therefore to be determined by the user. It should however be noted that setting these thresholds can turn out to be difficult, and some unexpected side effects can occur.

We now introduce the main contribution of this paper, a rule mining algorithm based on CN2 that extends two aspects of it relevant for reinforcement learning applications: allowing states to be mapped to *several* equally-good actions, and ensuring that the distilled policy performs comparably to the original policy.

4 Extending CN2 to Set-Valued Labels

Our main contribution consists of extending two important aspects of CN2, to make it suitable to policy distillation in an explainable reinforcement learning setting:

1. We exploit meta-information generated by the reinforcement learning process to associate *sets* of equally-good actions to each state. By offering choice to the rule miner in which action to map to every state, we potentially (and in practice do) reduce the number of rules needed to cover the original policy.
2. Contrary to classical Supervised Learning settings, the main application domain of CN2, what matters in our setting is not primarily the classification accuracy, but the loss in performance of the distilled policy. In other words, one little misclassification in a particular state can have a big impact on the performance of the policy, but in other states, this misclassification might almost be unnoticeable from a policy performance perspective. We propose a two-step approach that allows to identify those *important* misclassifications, leading to short rule sets that, when executed in the environment, still perform as well as the original policy. This ensures that the explanations we provide with the mined rules correspond to a good policy, and not some over-simplified degenerate policy.

The first point, associating sets of equally-good actions to each state, requires an extension of the CN2 algorithm itself. The second point, and the application of the whole framework to reinforcement learning policy distillation, requires an iterative framework that we present in Section 5.

In order to extend the CN2 algorithm to a set-valued setting, we have to rethink the search process of the rule mining algorithm. As explained above, CN2 iteratively learns a rule through a beam search procedure. The objective of this step is to search for a rule that on the one hand covers a large number of instances, and on the other hand has a high accuracy. This is steered through the evaluation heuristic, i.e. the weighted relative accuracy (WRA), which makes a distinction between positive and negative examples based on the label of the training sample. In order to deal with the Set-Valued instances, which express an indifference with respect to the actual label chosen by the rule, we adapted the WRA heuristic.

In the regular single-label setting, the sum of positive and negative examples covered by the rule and in the overall training set, $\hat{P} + \hat{N}$ and $P + N$, always corresponds to the number of respective training examples, $\hat{E}$ and E. However, in a multi-label setting this is not necessarily the case, as an instance can be assigned to any of the labels and is thus potentially both a positive and negative example at the same time according to the standard WRA heuristic. The heuristic aims at maximising the correctly classified instances for a rule by assigning the majority class of the covered samples as the class label. In order to deal with the indifference to which instances with multiple class labels are assigned, we propose the following adaptation to the WRA heuristic, described in Equation 2, where we replace the sum of positive and negative examples covered by the rule and in the overall set by the actual number of distinct samples in the set, respectively denoted by E and $\hat{E}$:

$$\text{WRA}_{set} = \frac{\hat{E}}{E} \times \left(\frac{\hat{P}}{\hat{E}} - \frac{P}{E} \right) \tag{2}$$

This will have the effect that the heuristic will only count the multi-labelled instances as a positive example when the majority class is one of the options, and as a negative example otherwise.

In Figure 1 we illustrate our set-valued rule mining on synthetic multi-label data consisting of 2 labels, with both the regular and our adapted heuristic. Our heuristic, WRA$_{set}$, allows to express the data into two nice groups, where the multi-labelled green points get assigned to one of the labels of the corresponding set, to trade-off both the coverage of the rule and its accuracy.

Our extension of CN2, able to learn from pairs of states and multiple equally-good labels, is only one component of our explainable reinforcement learning framework. We now fully detail the iterative algorithm that we built on our CN2 extension, that allows a policy to be learned, distilled, evaluated and fine-tuned by the user, to ensure that state-action pairs that are crucial for the performance of the policy are classified correctly.

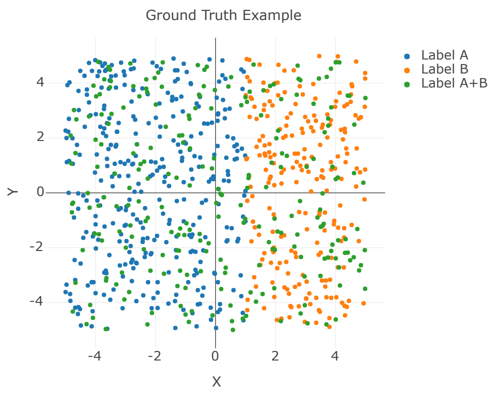
5 Explaining Policies through Distillation

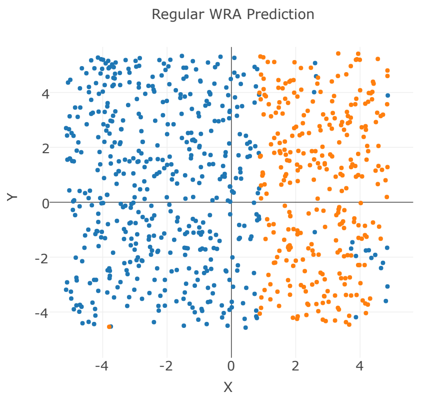
In this work, we are using inductive rule mining to translate a deep reinforcement learning policy into a simplified ordered list of rules that reveal the agent's behaviour in the environment. The simplification is performed through a rule-mining approach that exploits the meta-information that is present in the RL learning algorithm, hereby balancing between accuracy and readability.

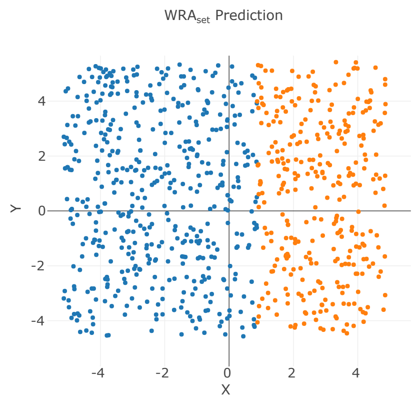
We extended the implementation of CN2 in the open source data mining framework Orange[5], in order to support multi-label data and incorporated our proposed WRA$_{set}$ heuristic to perform set-valued rule mining. We evaluated our methods on an altered level of the Mario AI benchmark [10]. The goal of the Mario agent is to collect rewarding red coins and avoiding penalising green and blue coins. The action space of the Mario agent is restricted so that it can only press one of the 5 available controller buttons during each step. The agent can also decide not to press any controller button, bringing the total number of actions to 6.

The observations provided to the RL agent are built from a description of the enemies, coins and blocks around the current position of Mario in the environment. We consider a 10 by 10 receptive field, that is, we look at 10 by 10 unit squares in the environment, centred around Mario's current position. Each square can either be empty, or contain a block, enemy or coin. Because every class of enemy, or every colour of a coin, is uniquely identified, each square can take 24 different values (0 being empty, then positive values identifying the kind of object in the square). When producing observations for the agent, we consider this 10 by 10 set of squares that can each take a value among 24, and produced a flattened, one-hot encoded, large observation vector of $10 \times 10 \times 24 = 2400$ entries, with each entry either 0 or 1. Figure 2 illustrates the transformation from a game frame to a receptive field. Because one-hot encoded states can be

[5] https://github.com/biolab/orange3





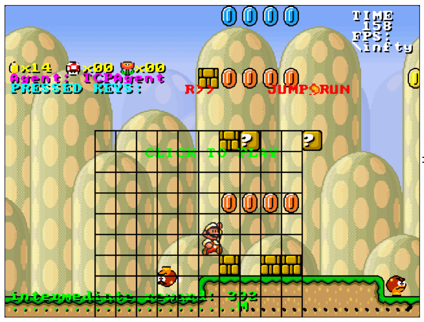


```
1. IF X<=1.0 AND Y>=-4.96 THEN Class=A
2. IF Y>=-0.81 AND Y<=3.34 AND X>=1.12 THEN Class=B
   ...
6. IF X<=3.86 AND Y<=-2.18 AND Y>=-4.97 THEN Class=B
7. IF TRUE THEN Class=A
```

```
1. IF X<=1.0 THEN Class=A
2. IF X>=1.01 THEN Class=B
3. IF TRUE THEN Class=A
```

Fig. 1. The top graph is a synthetic multi-label data set. Instances can have one or two labels. Samples only associated to label A are shown in blue, instances only associated to label B in orange. Instances associated to both labels are shown in green. The green instances are scattered throughout the entire graph. Label predictions made by our Set-Valued CN2 algorithm using the standard WRA and our adapted WRA$_{set}$ heuristic are visualised in the graphs underneath. In the leftmost graph (standard WRA), some of the instances for which either label is fine got labelled A (blue). This results in a large list of 7 rules. Our adapted WRA$_{set}$ heuristic addresses this problem. The resulting list of only 3 rules cleanly divides the data into two groups. In both list, the last rule represents a default rule.

decoded back to their integer variant (a 10 by 10 grid of integers in our case), and the integers we consider uniquely identify objects in the Mario game that have human-friendly game, any observation passed to the agent can also be described in a human-friendly way, such as "*there is a red coin above Mario*".



$$\Rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 9 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 & 9 & 9 \\ 0 & 0 & 0 & 23 & 0 & 11 & 11 & 11 & 11 & 11 \\ 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 2. Transformation from a MarioAI game frame to a 10 by 10 receptive field grid centred around Mario, which is returned by the emulator. Each cell of the grid corresponds to a pixel region about the size of a brick block in the game. The game objects in each of the grid's cells are translated into numerical labels as state representation for the learning agent. Whenever the receptive field ranges out of the game frame, it is padded with zeros (e.g. the bottom row).

Our algorithm to produce rules of the Deep Reinforcement Learning policy works in two phases. The first phase produces a list of rules that approximate how the Deep RL policy maps states to actions. These rules capture the general and high-level behaviour of the policy, and produce compact and human-friendly descriptions. However, some of the approximations made by our rule mining algorithm may lead to rules that, if executed in the environment, lead to a significant loss in performance. In principle, this can be tuned by the parameter settings of the rule-mining algorithm. We could increase the importance of the accuracy, to get more accurate rules. However, as mentioned before, a single misclassification can have a huge impact, or it can be hardly noticed. Therefore, our approach provides a second phase, consisting of a refinement of the rules. Incorrectly classified state-action pairs, that have a significant impact on the performance of the resulting policy, are identified and used to refine the corresponding rules.

5.1 Phase 1: Supervised post-processing

We construct a pipeline to process the behaviour of the RL agent, consisting of the following steps:

1. Train a Deep RL agent, we assume that the RL agent produces a stochastic policy (which is the case for BDPI, that we use in our experimental evaluation);

2. Record agent behaviour by logging multiple trajectories of episodes performed by the policy;
3. Translate the trajectories into a collection of symbolic states and set-valued actions, so that each state can be mapped to several actions if the Deep RL policy gives similar probability to these actions;
4. Mine rules with Set-Valued CN2.

First, a deep RL agent is trained to control Mario to play a single level. In this paper we realise this through the use of Bootstrapped Dual Policy Iteration (BDPI) [21]. We chose this algorithm because it produces an explicit actor, from which we can obtain action probabilities, and the off-policy actor-critic nature of the algorithm leads to an actor that computes the probability that an action is optimal, instead of a simple number that empirically led to high returns, as Policy Gradient approaches do.

Once a policy has been learnt, a data set is recorded by executing the converged RL policy in the environment for 50 episodes. The data set consists of tuples $(s, \pi(s))$ of state observations and action probabilities for each step of the performed episodes.[6]

Given the data set of states and their corresponding action probabilities, a translation is applied in order to prepare for mining the rules. The numerical values of the state observations are transformed back to their symbolic counterpart, as each value in the observation represents a specific game object in Mario's receptive field. Since the state observation represents a receptive field around Mario, we can assign x,y coordinates to each cell (feature of the state) with the agent in the centre.

For each state observation s in the data set, the predicted action distribution $\pi(s)$ is transformed into a set, $\mathcal{L}(s)$. This set denotes which actions are considered suitable to be executed in that particular state. Equation 3 describes this transformation formally. For each state s in the recordings, we take the probability of the best action, i.e. the action with the highest probability, $\max_{a'}(\pi(s, a'))$, and set a proportional threshold, $\tau * \max_{a'} \pi(s, a')$, where $\tau \in (0, 1]$ determines how tolerant we are with respect to sub-optimal actions. Lower values of τ will lead to more actions exceeding the threshold, while high values will put the threshold closer to $\max_{a'}(\pi(s, a'))$. For our experiments we picked a value for $\tau = 90\%$.

$$\forall s \in S : \mathcal{L}(s) = \{a \in A(s) \ : \pi(s, a) \geq \tau * \max_{a'}(\pi(s, a')) \mid \forall a' \in A(s)\} \tag{3}$$

Once the symbolic data set is created, we can proceed mining rules with our Set-Valued CN2 algorithm. We explored different settings; providing the rule mining the full state space of the RL as feature space or reducing the feature space to the upper right quadrant of Mario's receptive field, that is the cells in front of Mario. We also performed rule mining including and excluding the inequality (! =) predicate. The reduced feature set, with the excluded inequality

[6] Since BDPI is an actor-critic algorithm, we use the actor predictions as output, i.e. probabilities over actions. With value-based algorithms, e.g. DQN [16], one could record the Q-values instead.

predicate provides more interpretable rules, and are listed below. The default parameters of the beam search, expressing the maximal number of conditions per rule, the minimal number of samples to be covered by a rule and the beam width, were put respectively to 5, 20 and 10. Learned rules by Set-Valued CN2 using our WRA$_{set}$ heuristic, that explain the policy our BDPI agent has learned in the Mario environment, are listed below:

```
 1. IF (1, 5)==NULL AND (0, 5)==NULL AND (0, 1)==NULL THEN Class=JUMP
 2. IF (1, 5)==BRICK AND (1, 0)==COIN_RED THEN Class=RIGHT
 3. IF (1, 5)==BRICK AND (2, 4)==COIN_RED THEN Class=RIGHT
 4. IF (4, 2)==NULL AND (0, 4)==NULL THEN Class=JUMP
 5. IF (4, 5)==NULL AND (3, 3)==NULL THEN Class=JUMP
 6. IF (3, 5)==BRICK THEN Class=JUMP
 7. IF (2, 2)==BRICK THEN Class=JUMP
 8. IF (0, 2)==NULL AND (0, 1)==NULL THEN Class=RIGHT
 9. IF (0, 5)==NULL THEN Class=JUMP
10. IF TRUE THEN Class=JUMP
```

The extraction resulted in a list of 10 rules, capturing the agent's behaviour at a high level. In overall, the conditions of the rules focus on empty spaces (NULL) and the presence of bricks or red coins. The second rule, for instance, explains that the policy likes to move to the right when there is a (positive rewarding) red coin in front of Mario. However, a problem with the learned rules is that they over-emphasise jumping, the action that is most often the best one, but that is not enough to finish a Mario level (Mario also has to move right to win). While the rules provide a high-level explanation of the policy followed by the agent, they do not allow the agent to obtain high returns when executed. Our second phase, the refinement, addresses this issue by allowing the rule mining algorithm to produce more refined rules, that lead to better execution performance. A distillation of the greedy policy ($\tau = 1$) was also performed but resulted in a longer rule list with similar poor performance.

5.2 Phase 2: Refinement

The rules produced by the first phase, described above, are optimised for classification accuracy. However, accuracy is not the main objective in policy distillation, as high accuracy (except in some rare but important states) may still lead to poor performance in the environment. We therefore introduce a second phase, that refines the rules of the first phase based on roll-outs in the reinforcement learning environment. The second phase ensures that good policies are produced by our algorithm:

1. Use the Deep RL policy to perform an episode in the environment, observe the predicted actions step-wise;
2. In each time-step, predict the action that would be chosen by the obtained rules from phase 1 and note which rule was applied for the prediction;

3. When the predicted action of the Deep RL policy and the rules are different, store the observed state together with the action of the Deep RL policy in the training set of the respective rule;
4. Once the episode is finished, we extend each training set with samples from phase 1 that were correctly classified in order to get a balanced data set;
5. Launch a new rule mining process for each balanced training set using Set-Valued CN2 with the respective rule of the first phase as a starting point for the beam search process.

We execute the Deep RL policy in the environment for a whole episode and compare its predictions with the actions that would have been selected by the rules learned during phase 1. In every time-step, we collect the state visited by the agent and the action predicted by the Deep RL policy. We also predict the action that the rules from the first phase would choose and log the actual rule that is fired. When both policies disagree, we will store the observed state and predicted action from the Deep RL policy in a separate training set for the rule that predicted incorrectly, in order to refine it after the episode finishes.

Given the set of mismatches per rule, each *parent rule* will be refined through our set-valued CN2 with the same parameters as during the first phase. Here we allow both the equality and inequality operator to be used to generate conditions in the rule mining process. Note that we extend each training set with samples that were correctly covered by that rule, to get a balanced data set, rather than learning new rules on a set solely consisting of misclassified samples. We use the parent rule as a starting point in each iteration of the beam search, hence adding more conditions to the parent rule and refining it. If we were to take for example the fourth rule from the resulting rule list in the first phase, one of the resulting refined rules would be:

`IF (4, 2)==NULL AND (0, 4)==NULL AND (2, 5)!=BRICK AND (0, 3)==NULL AND (1, 2)!=COIN_RED AND (1, 2)!=COIN_BLUE AND (1, 2)!=BORDER_CANNOT_PASS_THROUGH THEN Class=RIGHT`

By refining a rule when appropriate, more instances will be classified correctly. Yet, the high-level view is not altered. The same procedure is applied for the subsequent rules, whenever this contributes to increasing the performance.

This refined rule set can now be evaluated by checking its performance. When satisfactory results are obtained, we stop, otherwise a deeper refinement can be added in e.g. a third phase.

Performing this refinement procedure in our example of Mario with the rules obtained from the first phase (listed in Subsection 5.1), caused a refinement of multiple rules. All the refined rules can be inserted in the original list in front of their parent rule, thus expanding the original list.

While a refinement generates a lot of extra rules, it should be noted that this is a *hierarchical refinement*: the high-level view remains untouched. We executed the refined rule list in the environment for 50 episodes and compared its performance with the performance of the original BDPI policy. This is shown in Figure 3. The rules were capable to reach a performance similar to BDPI, hence we did not perform further refinement.

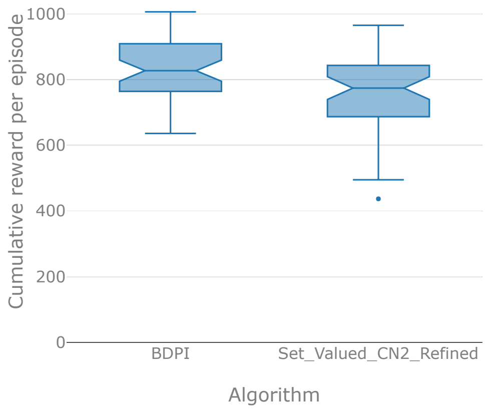


Fig. 3. Comparison of episodic cumulative reward (y-axis) achieved by the refined extracted rules of our Set-Valued CN2 on the Mario environment with the original Deep RL agent. 50 episodes were played by both agents (x-axis). The refined rules we obtain after phase 2 have an equivalent performance to the original policy since the box plots highly overlap. A bad-performing agent obtains 100 on this environment.

6 Conclusion

In this paper we have proposed an approach to translate a policy trained by a Deep RL algorithm into a set of rules. The set is human-readable, captures the behaviour of the policy, and can be iteratively refined to obtain higher returns if executed in the environment. A key element of our approach is that we associate every state to a *set* of equally-good actions, obtained by looking at the action probabilities predicted by the RL policy. This part of our contribution allows our modified rule mining algorithm to simplify its rule set. By allowing to select one of the almost equally-good actions for a state, the number of generated rules is significantly reduced compared to applying CN2 on the greedy policy. We find the iterative refinements appealing with respect to explainability. The rules of phase one are not altered, but only hierarchically expanded. Thus, the rules of the first phase provide a high-level view and every refinement adds details. Our refinement process is driven by the performance of the rules, measured by their execution in the environment. This prevents us from being dependent on magic numbers (i.e. the setting of the different parameters) of the rule mining algorithm. Note that we also do not use the number of mismatches (or accuracy) as an evaluation measure, but the performance of the policy expressed in the rules, which is a more meaningful measure. Furthermore, our method only assumes that the Deep RL algorithm produces a probability distribution over the actions, which can also be retrieved from value-based algorithms by using e.g. a Softmax function over Q-values. Alternatively, Equation 3 can also be directly applied to Q-values.

References

1. Agogino, A.K., Tumer, K.: Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. Autonomous Agents and Multi-Agent Systems **17**(2), 320–338 (Oct 2008). https://doi.org/10.1007/s10458-008-9046-9
2. Brys, T., Nowé, A., Kudenko, D., Taylor, M.E.: Combining multiple correlated reward and shaping signals by measuring confidence. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. p. 1687–1693. AAAI Press, Palo Alto, California (2014)
3. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3**(4), 261–283 (Mar 1989). https://doi.org/10.1007/BF00116835
4. Coppens, Y., Efthymiadis, K., Lenaerts, T., Nowé, A.: Distilling Deep Reinforcement Learning Policies in Soft Decision Trees. In: Miller, T., Weber, R., Magazzeni, D. (eds.) Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence. pp. 1–6. Macau (Aug 2019)
5. Frosst, N., Hinton, G.: Distilling a Neural Network Into a Soft Decision Tree. In: Besold, T.R., Kutz, O. (eds.) Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017. AI*IA Series, vol. 2071. CEUR Workshop Proceedings, Aachen, Germany (Nov 2017)
6. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of Rule Learning. Cognitive Technologies, Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-540-75197-7
7. Gevaert, A., Peck, J., Saeys, Y.: Distillation of Deep Reinforcement Learning Models using Fuzzy Inference Systems. In: Beuls, K., Bogaerts, B., Bontempi, G., Geurts, P., Harley, N., Lebichot, B., Lenaerts, T., Louppe, G., Van Eecke, P. (eds.) Proceedings of the 31st Benelux Conference on Artificial Intelligence (BNAIC 2019) and the 28th Belgian Dutch Conference on Machine Learning (Benelearn 2019). vol. 2491. CEUR Workshop Proceedings, Aachen, Germany (2019)
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the Knowledge in a Neural Network. arXiv e-prints arXiv:1503.02531 (Mar 2015)
9. Huang, J., Angelov, P.P., Yin, C.: Interpretable policies for reinforcement learning by empirical fuzzy sets. Engineering Applications of Artificial Intelligence **91**, 103559 (May 2020). https://doi.org/10.1016/j.engappai.2020.103559
10. Karakovskiy, S., Togelius, J.: The Mario AI Benchmark and Competitions. IEEE Transactions on Computational Intelligence and AI in Games **4**(1), 55–67 (Mar 2012). https://doi.org/10.1109/TCIAIG.2012.2188528
11. Lavrač, N., Flach, P., Zupan, B.: Rule Evaluation Measures: A Unifying View. In: Džeroski, S., Flach, P. (eds.) Inductive Logic Programming. Lecture Notes in Computer Science, vol. 1634, pp. 174–185. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48751-4_17
12. Libin, P., Moonens, A., Verstraeten, T., Perez Sanjines, F.R., Hens, N., Lemey, P., Nowé, A.: Deep reinforcement learning for large-scale epidemic control. In: Proceedings of the Adaptive and Learning Agents Workshop 2020 (ALA2020) at AAMAS (2020)
13. Madumal, P., Miller, T., Sonenberg, L., Vetere, F.: Explainable reinforcement learning through a causal lens. In: Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence. vol. 3, pp. 2493–2500. AAAI Press, Palo Alto, California (2020). https://doi.org/10.1609/aaai.v34i03.5631
14. Maes, P.: Computational Reflection. In: Proceedings of the 11th German Workshop on Artificial Intelligence. pp. 251–265. GWAI '87, Springer-Verlag, Berlin, Heidelberg (Sep 1987)

15. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1–38 (Feb 2019). https://doi.org/10.1016/j.artint.2018.07.007
16. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
17. Molnar, C.: Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. Leanpub (2019), https://christophm.github.io/interpretable-ml-book/
18. Rusu, A.A., Colmenarejo, S.G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., Hadsell, R.: Policy Distillation. In: 4th International Conference on Learning Representations (Nov 2016), arXiv:1511.06295
19. Rückstieß, T., Sehnke, F., Schaul, T., Wierstra, D., Sun, Y., Schmidhuber, J.: Exploring Parameter Space in Reinforcement Learning. Paladyn, Journal of Behavioral Robotics **1**(1), 14–24 (Mar 2010). https://doi.org/10.2478/s13230-010-0002-4
20. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of Go without human knowledge. Nature **550**(7676), 354–359 (Oct 2017). https://doi.org/10.1038/nature24270
21. Steckelmacher, D., Plisnier, H., Roijers, D.M., Nowé, A.: Sample-efficient model-free reinforcement learning with off-policy critics. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) Machine Learning and Knowledge Discovery in Databases. Lecture Notes in Computer Science, vol. 11908, pp. 19–34. Springer International Publishing, Cham, Switzerland (2020)
22. Sutton, R.S., Barto, A.G.: Reinforcement Learning : An Introduction. MIT Press, Cambridge, Massachusetts, 2nd edn. (2018)
23. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation. In: Neural Information Processing Systems (NIPS). pp. 1057–1063 (2000)
24. Tadepalli, P., Givan, R., Driessens, K.: Relational Reinforcement Learning: An Overview. In: Tadepalli, P., Givan, R., Driessens, K. (eds.) Proceedings of the ICML-2004 Workshop on Relational Reinforcement Learning. pp. 1–9. Banff, Canada (2004)
25. Todorovski, L., Flach, P., Lavrač, N.: Predictive Performance of Weighted Relative Accuracy. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science, vol. 1910, pp. 255–264. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/3-540-45372-5_25
26. Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., Battaglia, P.: Deep reinforcement learning with relational inductive biases. In: 7th International Conference on Learning Representations (2019), https://openreview.net/forum?id=HkxaFoC9KQ