Safety Aware Reinforcement Learning by Identifying Comprehensible Constraints in Expert Demonstrations

Leopold Müller,[1] Lars Böcking, [1] Michael Färber [1]
[1] Karlsruhe Institute of Technology
leopold.mueller@student.kit.edu, lars.boecking@student.kit.edu, michael.faerber@kit.edu

Abstract

When used in real-world environments, agents must meet high safety requirements as errors have direct consequences. Besides the safety aspect, the explainability of the systems is of particular importance. Therefore, not only should errors be avoided during the learning process, but also the decision process should be made transparent. Existing approaches are limited to solving a single problem. For real-world use, however, several criteria must be fulfilled at the same time. In this paper we derive comprehensible rules from expert demonstrations which can be used to monitor the agent.

The developed approach uses state of the art classification and regression trees for deriving safety rules combined with concepts in the field of association rule mining. The result is a compact and comprehensible rule set that explains the expert's behavior and ensures safety. We evaluate our framework in common OpenAI environments. Results show that the elaborated approach is able to identify safety-relevant rules and imitate expert behavior especially in edge cases. Evaluations on higher dimensional observation spaces and continuous action spaces highlight the transferability of the approach to new tasks while maintaining compactness and comprehensibility of the rule set.[1]

Introduction

In real applications, such as autonomous vehicles, the explainability of the systems' decisions is highly relevant. When the algorithm is in charge, we want to know what it is doing and we want to know why it is doing it. The explainability of the systems is not only relevant for the legal framework but also for social acceptance. Reinforcement learning approaches based on deep learning achieve excellent results in terms of their target function such as reward, but do not offer explainability and traceability. Association rule mining is concerned with identifying patterns in data and formalizing them for reproducibility and explainability. Existing approaches are mainly concerned with modeling the complete context in data which leads to comprehensiblity problems. The approach presented here combines these research areas in a new way, with a special focus on application.

In the area of reinforcement learning, there are various approaches to increase the safety of the agent. However, these are limited either to modifying the optimization criterion or to restricting the exploration process. First approaches to draw on existing knowledge in the form of expert demonstrations focus mainly on an accelerated convergence via so-called warm starting. So far, there is a lack of approaches to specifically extract existing knowledge from previously unused data and integrate this knowledge into the learning process.

This work attempts to make use of existing expert demonstrations in two ways. Firstly, it tries to make the decision-making process of the systems more transparent by identifying human-understandable rules. Secondly, the rules found are used to monitor the system. The system is prohibited from violating the given safety rules.

The combination of state of the art classification and regression trees with association rule mining represents a novelty in the field of safety reinforcement learning and opens up new possibilities to transfer the algorithms to new application areas. In critical domains such as medicine or transportation, the presented framework fulfills key requirements in terms of comprehensibility and transparency. The key contributions of this work are:

- applying association rule mining to RL demonstrations
- combining classification and regression trees with association rule mining
- a framework for deriving comprehensible set of rules
- integrating a safety layer into the learning process

Related Work

Modification of the optimization criterion. In the simplest case, the optimization criterion is the risk-neutral criterion. Trust Region Policy Optimization (*TRPO*) is a further development of *policy gradient methods*. They are used to optimize strategies in the form of neural networks (Schulman et al. 2015). The change of parameters must not exceed a step size called *Trust Region*, which restricts the set of possible strategies. A further development of the TRPO is, the *Proximal Policy Optimization Algorithm* presented in (John Schulman et al. 2017) (*PPO*). While the process of strategy discovery by the two algorithms is significantly more effective and goal-oriented than a stochastic learning algorithm,

[1]https://github.com/leopoldmueller/safety-aware-reinforcement-learning

they rely on the assumption of ergodicity (Moldovan and Abbeel 2012). However, this does not exist in reality. Fast and effective strategy discovery does not protect against errors or irreversible states. The approach of this work does not try to replace learning algorithms like TRPO or PPO, rather it tries to complement them. Thus, we use the PPO algorithm in combination with the approach of this paper.

Modification of the exploration process. To avoid undesirable states, these must be marked as undesirable without visiting them. This is impossible without external knowledge, since otherwise they would have to be visited during the random exploration process (Javier García, Fern, and o Fernández 2015). In general, external knowledge can be used in two different ways: (1) derive a strategy from a set of trajectories or (2) guide the exploration process by recommendations from a teacher (Javier García, Fern, and o Fernández 2015). First subjects the agent to a kind of initialisation procedure. This is done on the basis of prior knowledge about the task. The RL agent learns a strategy based on expert demonstrations. In doing so, it imitates the behaviour that the expert demonstrated. In the literature, this approach is referred to as *Imitation Learning* (Hussein et al. 2017). Another approach to imitation learning is *Inverse RL* (Finn, Levine, and Abbeel 2016). Instead of the strategy, a reward signal is learned using the state-action pairs. Similarly, an approach is possible where the agent is guided by an expert when the chosen action falls below a safety value. In (Menda, Driggs-Campbell, and Kochenderfer 2019) an approach is presented in which a trained agent takes over the role of the expert. However, in many application areas, no agent exists that can take on this role.

Explainability of reinforcement learning systems. Explainability is often considered a trade off to performance (Puiutta and Veith 2020; Longo et al. 2020). In (Bastani, Pu, and Solar-Lezama 2018) an agent is deployed in an environment and its behaviour is recorded. Based on the demonstrations, a decision tree is then trained and used to verify the agent's strategy. The most common group of algorithms in the field of **association rule mining** are Apriori Algorithm. They are designed to identify rules that have a minimum level of confidence and support. Algorithms like ID3 were continuously improved resulting in C4.5 which removed the restriction that features must be categorical, or memory optimized versions like the C5.0. Most recent advancements were achieved by CART (Classification and Regression Trees) which constructs binary trees. There are existing approaches to combine the research field of association rule mining with other fields of application such as integrating classification and association rule mining (Liu, Hsu, and Ma 1998). Further research deals with online learning where association rules are derived from a continuous stream of information (Hidber 1999). Combined with reinforcement learning, there are first applications of association rule mining to solve, for example, multi-agent environments (Kaya and Alhajj 2005).

Approach

This paper attempts to make reinforcement learning safer. For this purpose, safety rules are derived from existing data sets. On the one hand, these serve to increase the transparency of the agent's decision-making, and on the other hand, they can be used to monitor the agent.

The starting point for this approach are expert demonstrations. It is expected that for the task to be solved there is already an instance of any kind that can interact with the environment. This instance is called an expert. In the case of autonomous driving, a human driver could control the vehicle. Figure 1 summarises the complete systematics of the approach.

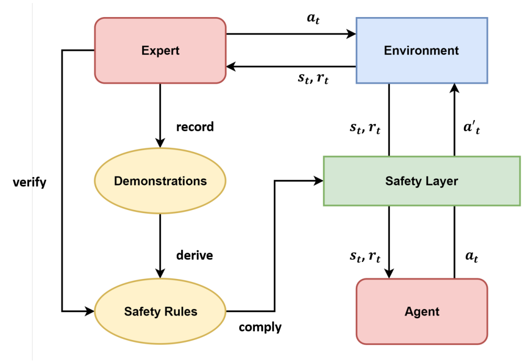

Figure 1: Systematics of the approach of this paper.

The derivation of the safety rules and the integration of these, in the form of a safety layer, form the basis of the approach and are explained below.

Derive Safety Rules

The starting point for deriving safety rules are the expert demonstrations (see Figure 1). They consist of a finite set of trajectories containing a sequence of states and actions. This behaviour is assumed to be safe.

In the first step, the CART algorithm is applied to the expert demonstrations. This algorithm creates a decision tree based on the states and the actions that follow. The structure of the decision tree provides the basis for the decision rules, which are interpreted as safety rules. The decision tree classifies states according to their characteristics. By querying the edges, the states are grouped according to the same actions. It is first assumed that a finite set of actions, such as "braking" and "accelerating", is available for selection. The actions are thus discrete. In the next step, the paths of the tree are converted into decision rules and summarised as a rule set.

The result of the framework is a rule set consisting of all decision rules or paths of the decision tree. The length and number of rules is determined by the depth of the tree. Therefore, the rule set can become arbitrarily large depending on the complexity and amount of labelled data. To make the decision process comprehensible, the rule set should be kept as compact as possible. This means that the size of the tree must be limited with the help of a termination criterion.

One possibility is to set a **maximum tree depth**. In this case, the splitting of a node stops as soon as its depth corresponds to the maximum tree depth. If, for example, the algorithm stops at a leaf node that specifies ”braking” for one half of the states and ”accelerating” for the other, the rule found is not a suitable candidate for a safety rule. A safety rule should specify an action that is as unambiguous as possible. Other factors must therefore be taken into account.

An alternative is to use the **ginicoefficient**. In this case, the equality of all selected actions is quantified. Trying to apply this criterion in larger action spaces, however, means that rules in which only a small subset of the possible actions are used can also be included in the set of rules due to their low ginicoefficient. This distribution based criterion does not ensure unambiguous actions.

To achieve this, metrics from the **association rules** area are considered. Decision rules are understood as association rules of the form $X \implies e_k$. The queries of the characteristics of states s_t (referred to as x_t in the decision tree) along the edges of a path represent X. The action a_t (represented in the decision tree by y), to which most states in the leaf node of the path are assigned, forms the consequence e_k of X. Now the rule set can be filtered for relevant rules. For this purpose, minimum value for **support** and **confidence** are set. Algorithm 1 describes the framework for discrete action spaces.

Algorithm 1: Constraints Identifier

Input: set of trajectories Ω
Parameter: $supp_{min}$, $conf_{min}$
Output: set of rules C

```
 1: X = list of states in $\Omega$
 2: Y = list of actions in $\Omega$
 3: $C$ = create empty list of rules $C$
 4: filtered paths = create empty list of filtered paths
 5: tree = DecisionTreeClassifier($X$, $Y$)
 6: for path in tree do
 7:     for node in path do
 8:         if $supp(node) > supp_{min}$ and $conf(node) > conf_{min}$ then
 9:             cut off subsequent nodes
10:             append shortened path to filtered paths
11:         else
12:             if node is leaf then
13:                 go to next path in tree
14:             else
15:                 go to next node in path
16:             end if
17:         end if
18:     end for
19: end for
20: filtered paths = make filtered paths unique
21: for path in filtered paths do
22:     c = ConvertPathIntoRule(path)
23:     append $c$ to $C$
24: end for
25: return set of rules $C$
```

Rules that do not meet the minimum values are removed from the rule set. The support describes the statistical relevance of a rule. A rule with a support of 0.1 applies to 10% of all states. Confidence of a rule indicates its uniqueness. A rule with a confidence of 0.8 means that in a state where the rule applies, the action specified by the rule is performed in 80% of the cases. The use of support and confidence as termination criteria leads to a rule set in which all rules contain both a statistical relevance and a safe action instruction determined by the confidence. After filtering the paths of the decision tree using the hyperparameters supp_{min} and conf_{min}, redundant rules are removed in line 20 and converted into safety rules in line 22. For this purpose, all queries of the decision nodes are linked via logical *ands* starting from the root of the tree to the currently considered leaf. The most frequently selected action in the leaf is assigned to the rule.

Adaptations for continuous action spaces. Differentiate two cases: (1) there are two discrete actions ”braking” and ”accelerating” and (2) there are two continuous actions ”braking” and ”accelerating”, both can take a value from $[0, 1]$. In case (1), algorithm 1 can be applied without adaptation. A decision tree is created for the classification of states. In case (2), we extend our framework to categorize continuous values into discrete ranges. Then algorithm 1 is applied to each action. In order to specify the granularity of the subdivision of the action space, we add the hyperparameter **divider**, which specifies the number of intervals to divide into.

Integration of Safety Rules

The final component uses the set of rules to monitor an agent during the learning/deployment process. The approach can therefore be used both during the learning process and during the deployment process because the agent itself is not directly modified. If a chosen action violates a rule it is adjusted accordingly. In Figure 1, this safety layer is shown in green. The edges show input and output as well as access to the rule set. Algorithm 2 depicts how the safety layer works. In case of continuous actions, these must first be converted into discrete values in order to be checked with the safety rules in the next step.

Algorithm 2: Safety Layer

Input: set of rules C, state s, action a
Output: action a

```
 1: for rule in $C$ do
 2:     if if features of s fulfil all conditions of $c$ then
 3:         if $a$ == action of rule then
 4:             break
 5:         else
 6:             $a$ = adapt action of rule
 7:             break
 8:         end if
 9:     else
10:         continue {threshold not surpassed}
11:     end if
12: end for
13: return $a$
```

Evaluation

In this section, the approach is used in different scenarios to evaluate it in terms of function and versatility. The evaluation is performed in three steps:

(1) setting up a suitable test environment,
(2) evaluating the framework to derive safety rules, and
(3) evaluating the safety layer.

Setup of Test Environment

For the evaluation, environments from Open AI (Brockman et al. 2016) are used. Expert demonstrations serve as input for the algorithm. Trained agents are used as experts. For the following evaluations, the *PPO2* algorithm was used together with a *MlpPolicy* (Hill et al. 2018). The number of time steps t determines the number of trajectories $|\Omega|$ included in the expert demonstrations and is kept unchanged at $15,000$ for all following experiments. The agent is then deployed in the environment until the specified number of time steps has been recorded.

Evaluation of the Derivation of Safety Rules

In this section, the framework for deriving safety rules described in algorithm 1 is evaluated. Three different investigations are carried out.

Influence of the Hyperparameters on the number of Rules in a Rule Set. First, the influence of the minimum values of support and confidence as termination criteria on the **number of safety rules** is investigated. For this, the framework from algorithm 1 is applied to the created expert demonstrations of the different environments. Depending on the set parameters, rule sets are generated as a result. Figure 2 shows the results for the CartPole environment.

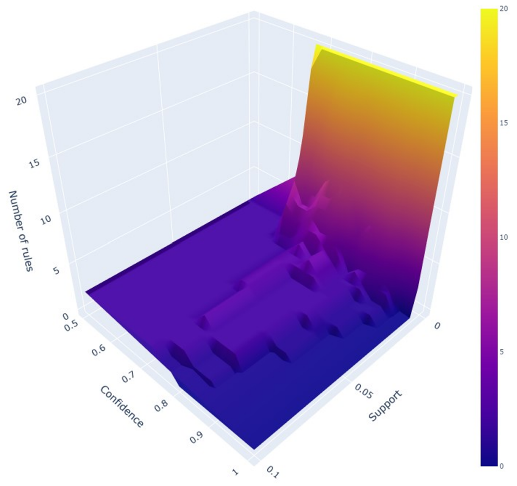

Figure 2: Comparison of rule sets by number of rules contained. The data is based on the CartPole-v1 environment.

At the point $(\text{supp}_{\text{min}} = 0, \text{conf}_{\text{min}} = 1)$, the rule set is unfiltered and thus has the highest number of rules. If the algorithm terminates before the subset of a node is unambiguously assigned, the confidence of the rule is less than one. This becomes clear when the course of the graph for $\text{supp}_{\text{min}} = 0$ is considered. The number of rules decreases continuously as the minimum value for confidence is reduced. This clearly shows that if a higher uniqueness of the rule is demanded (high confidence conf_{min}), the number of rules that are identified is lower. The same applies to the request of a more frequent occurrence of a rule (support supp_{min}).

Influence of the hyperparameters on the average length of the rules of a rule set. Comparable to the previous procedure, the influence of the hyperparameters on the average **length of safety rules** is now examined. Figure 3 illustrates the result of the investigation using the CartPole environment.

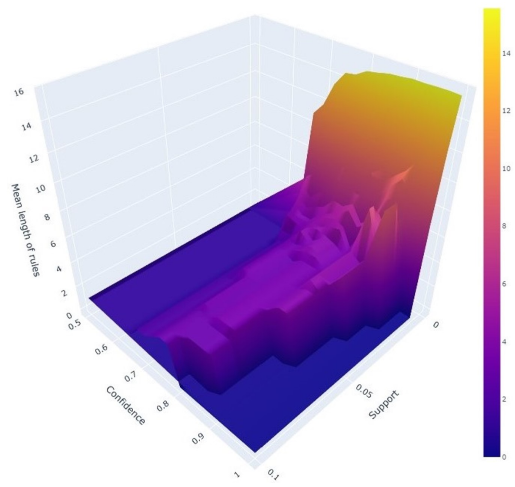

Figure 3: Comparison of rule sets according to the average length of the rules they contain, using the CartPole-v1.

At first glance, parallels to Figure 2 can be seen. Here, too, the maximum lies at the point $(\text{supp}_{\text{min}} = 0, \text{conf}_{\text{min}} = 1)$. For $\text{supp}_{\text{min}} > 0$ the maxima move to the "middle" of the confidence scale. From the investigations it can be concluded that the two hyperparameters influence both the average length and the absolute number of rules of a rule set in the same way.

Relevance of a rule set for the behaviour of agents during the learning process. In order to draw conclusions about the relevance of the rules of a rule set, the learning process of an agent is considered. The agent goes through a learning and a test phase alternately. It is first deployed in the environment without prior knowledge and tested for $20,000$ time steps. The number of times it enters a state for which a safety rule exists is recorded. In addition, it is documented how often it fulfils or violates these rules. This is followed by a learning phase in which the agent is trained for a certain number of time steps (set to 500). After completion of the learning process, another test phase follows. Two rule sets are examined using the CartPole environment as an example: one is the unfiltered rule set and the other is a filtered rule set. Table 1 summarises the parameters:

Hyperparameter	Value
Minimum value for support (filtered, unfiltered)	0.0050, 0
Minimum value for confidence(filtered, unfiltered)	0.95, 1

Table 1: Hyperparameter of the study: Relevance of a rule set for the behaviour of agents during the learning process.

The data were collected during a test phase for different training progress. The solid lines show the progression for the filtered, the dashed lines for the unfiltered rule set (cf. Table 1). Figure 4 shows how often a rule is adhered to (green line) or not adhered to (red line) in relative frequencies.

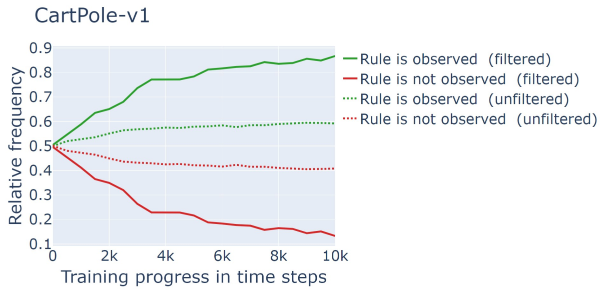

Figure 4: Relevance of a rule set for the behaviour of agents during the learning process.

For both cases (filtered and unfiltered rule set) the number of rule violations decreases with increasing training progress. A striking feature is the difference between the relative curves of the filtered and unfiltered rule sets. While the green curve of the filtered rule set converges towards $\mathrm{conf}_{\mathrm{min}} = 0.95$, this cannot be observed for the filtered rule set. One possible reason for this is the high number of rules. The rule set specifies an action for each state. If the agent chooses this action in every state, its behaviour corresponds in theory to that of the expert. However, since the agent and expert independently learn a strategy to solve the task, it is unlikely that the behaviour will match. As a result, the choice of actions differs in some states. The effect is also with the filtered rule set, but clearly smaller than with the unfiltered rule set. This suggests that the effect increases with the number of rules in a rule set.

Evaluation of Safety Layer

This section evaluates the safety layer. In the algorithms, a procedure has been formulated to monitor the agent using a set of safety rules. The framework is evaluated in common OpenAI environments.

Evaluation of the mode of operation for discrete action spaces. In order to draw conclusions about the functioning of the safety layer, different combinations of agent and rule set are compared. Object of the investigation is the aggregated reward that the agent receives at the end of an episode. For wrong decisions that lead to critical states (e.g. crashing the LunarLander), the agent receives a high negative reward. This means that the amount and variance of the reward is linked to safe behaviour of the agent. For the evaluation, an untrained agent (hereafter referred to as a novice) is used in three different ways. The novice, by its random strategy, represents the most uncertain state during the learning process. It is considered:

- Novice, unsupervised.
- Novice, monitored by safety layer with access to a filtered rule set.
- Novice, monitored by safety layer with access to an unfiltered rule set.

The hyperparameters for the rule sets (filtered/unfiltered) correspond to those in Table 1. The reference value for the final reward is the expert from whom the expert demonstrations originate. Figure 5 shows the results of the different runs using the CartPole environment as an example.

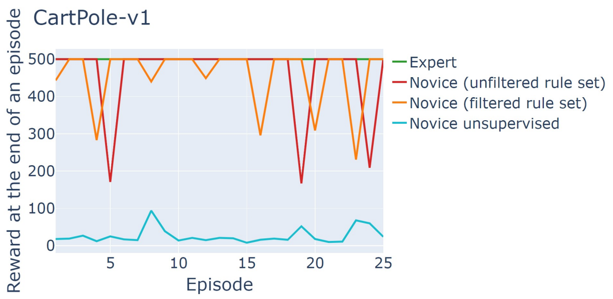

Figure 5: Evaluation for discrete action spaces based on the CartPole environment.

It can be clearly seen that the expert (green curve) has the best performance. In contrast, the reward of the unsupervised novice (blue curve) varies between 8 and 95. The comparatively poor performance can be explained by the fact that the novice chooses actions randomly. The novice has not learned a strategy for processing the given states. If this is monitored using the safety view from algorithm 2, the course changes depending on the rule set used. The red curve shows the performance when an unfiltered rule set is used. It corresponds in parts to the performance of the expert, but shows sharp dips (reward of 167) for some episodes. Nevertheless, the performance in these cases is clearly better compared to the unsupervised novice (blue curve). It can be concluded that the safety layer with access to the unfiltered rule set exerts a consistently positive influence on the novice's performance. When using a filtered rule set, the result is similar. Figure 6 presents the results using the LunarLander environment as an example.

Again the average reward of the expert (green) is the highest. In second place is the performance of the novice with unfiltered rule set (red), closely followed by the novice with filtered rule set (orange). In direct comparison, the performance of the unsupervised novice (blue) is the worst. A closer look reveals significant differences between Figure 6 and 5. The supervised novice with filtered rule set (orange) reaches the level of the expert (green) in places. The downward fluctuations are more pronounced than in the CartPole environment. This suggests that a wrong decision in

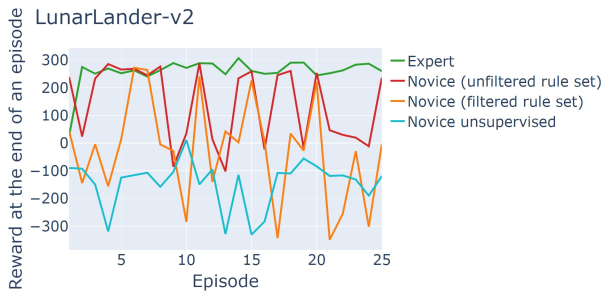

Figure 6: Evaluation for discrete action spaces using the LunarLander environment.

the LunarLander environment is difficult to compensate for. In concrete terms, this means that if the flying object gets into an unfavourable position, a controlled landing is hardly possible without a strategy. Analogous to the CartPole environment, the reward of the unsupervised novice (blue) is lowest, it increases when the novice is supervised. An increased number of safety rules improves the performance of the agent.

Evaluation of the mode of operation for continuous action spaces. To evaluate the safety layer in environments with continuous action spaces, the hyperparameters for the rule sets (filtered/unfiltered) are the same as before. The divider for the discretisation is set to two. The results are shown in Figure 7.

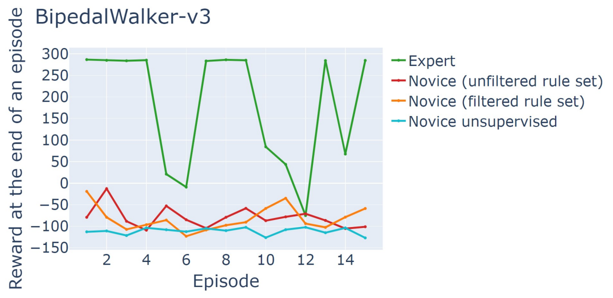

Figure 7: Evaluation for continuous action spaces using the BipedalWalker environment.

When looking at the performance of the expert (green), it is noticeable that it shows strong drops. The expert trained with the help of the PPO2 learning algorithm is not able to solve the environment. This contradicts the assumption that the expert demonstrates safe behaviour. The unsafe behaviour affects the quality of the expert demonstrations and thus the relevance of the safety rules. If the demonstrations contain errors, these are also reflected in the safety rules. The positive effect of the safety layer is dependent on the quality of the safety rules. If the curve of the unfiltered novice (red) is used, it is noticeable that it is clearly below the performance of the expert (green). In contrast to discrete action spaces, no improvement can be seen even in comparison to the unsupervised novice (blue). The same applies to the filtered novice (orange curve). However, the poorer performance of the two monitored novices (red, orange) is not solely due to the quality of the demonstrations. The two curves (red, orange) show no upward fluctuations. Even with non-optimal demonstrations, approaches of good behaviour should produce better performance than that of the unsupervised novice (blue). However, this is not the case and thus it is assumed that the complexity of the BipedalWalker environment is too high to achieve good performance without a strategy.

Lessons Learned

Current developments in technology make it possible to use advances in the field of artificial intelligence in a wide variety of areas. The focus is on tasks of high complexity where errors can have fatal consequences. In addition to the aspect of safety, trust in the intelligent systems is a hurdle that must be overcome. In order to strengthen trust, there are approaches in research that attempt to increase the explainability of systems. RL has achieved particular milestones in the past. This was made possible by using RL in conjunction with Deep Learning. Deep RL is considered a suitable candidate for complex tasks. However, the use of a neural network as a policy as well as learning it poses two problems: The black-box problem of the neural network (focus on explainability) and the trial and error approach of the learning process (focus on safety). In current research, there are several approaches that address these problems. The approach in this paper differs from existing ones in many ways. For one, both problems are addressed simultaneously. In addition, the concept can be used independently of the structure of the RL system. This means that neither the agent nor the environment itself needs to be modified. Another advantage is that the concept can be applied to different environments and agents simply by adjusting the hyperparameters. It is also possible to use existing data sets. This reduces the time needed by human experts and offers the possibility to build on existing solutions.

We developed a concrete framework for deriving safety rules from expert demonstrations. It is able to derive comprehensible rules from a set of trajectories. We used a decision tree based on the CART algorithm to derive the rules. The result of the decision tree is a rule set with a high number of rules. By using concepts from the field of association rules, the rule set can be filtered for relevant rules. We used the evaluation to show how the parameters affect the shape of the rule set and gave an intuition for the interpretation of the framework's hyperparameters. Using the metrics from the association rules domain, we developed a termination criterion that gives clear conclusions about the statistical relevance and uniqueness of the rules. With the help of the decision tree, these also have a comprehensible form. The framework is able to derive a compact rule set from expert demonstrations. The rule set reflects the behaviour of the expert in a comprehensible way. We implemented the integration of the safety rules as follows: A safety layer that monitors the agent ensures that the rules are followed at all times. The rules are derived from the expert's demonstrations and thus reflect his or her behaviour. They can thus be interpreted as safety rules, provided that the expert's behaviour is considered safe. The results of the evaluation have shown that in less complex environments, such as the LunarLander environment, the expert's performance can be achieved using the safety layer alone. In general, a positive effect regarding performance could be achieved by using the safety layer. A guarantee to avoid wrong decisions is only possible if the expert demonstrations cover all critical states. Only then can the rule set contain all the necessary rules. In order to make the framework as universally valid as possible, it was extended to include the ability to handle environments with continuous actions. Based on the significantly more complex environments, it could be determined that the application reaches its limits under the set goals of compactness and comprehensibility of a rule set.

Conclusion and Prospects

The results of this paper show the great potential of the concept *Safety Aware Reinforcement Learning by Identifying Comprehensible Constraints in Expert Demonstrations*, but also reveal first weaknesses. Only if the experts demonstrate how to deal with all critical states, it is possible that for each state there is a rule that specifies safe behaviour. Whether this is the case can only be verified by the expert. In less complex environments, such as the LunarLander environment, a rule set with a low number of rules could also achieve a high reward. However, this had a high variance. This is because the rules reflect the behaviour of the expert. The expert tries to land on the landing site even if he is far away from it. The rules resulting from this behaviour, force the supervised novice into risky manoeuvres that are difficult to intercept by the rule set alone. So instead of a simple but safe landing, what follows is an unnecessarily risky manoeuvre that leads to a crash. The results of the LunarLander environment have shown that the expert consciously takes risks. However, the risks should not be included in the safety rules. A possible solution could be the prioritisation of behaviour. In the case of the LunarLander, this would mean: An accident-free landing is necessary for safety, landing on the landing pad is secondary. One way to implement this is to use targeted demonstrations that are limited to critical situations. That is, the expert demonstrations are not composed of an arbitrary set of trajectories, but contain only those that show safety-relevant behaviour. In order to further investigate the functioning in complex environments with continuous action spaces, it should be examined whether a finer subdivision of the continuous interval into discrete actions achieves a positive effect. Other approaches could be a better performing expert or optimising the hyperparameters (amount of trajectories, support, confidence, etc.).

For a large number of the application areas commonly used in reinforcement learning, the framework developed in this work offers the possibility to learn clear and understandable rules from demonstrations. With the help of the framework, the rules can be integrated into the learning and deployment process of the agent. As with the agents themselves, the high dimensionality of environments and the continuous action spaces pose a challenge for this framework.

References

Bastani, O.; Pu, Y.; and Solar-Lezama, A. 2018. Verifiable Reinforcement Learning via Policy Extraction. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, NeurIPS'18, 2499–2509.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *CoRR*, abs/1606.01540.

Finn, C.; Levine, S.; and Abbeel, P. 2016. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *Proceedings of the 33nd International Conference on Machine Learning*, ICML'16, 49–58.

Hidber, C. 1999. Online association rule mining. *ACM Sigmod Record*, 28(2): 145–156.

Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2018. Stable Baselines.

Hussein, A.; Gaber, M. M.; Elyan, E.; and Jayne, C. 2017. Imitation Learning. *ACM Computing Surveys*, 50(2): 1–35.

Javier García; Fern; and o Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16(42): 1437–1480.

John Schulman; Filip Wolski; Prafulla Dhariwal; Alec Radford; and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Kaya, M.; and Alhajj, R. 2005. Fuzzy OLAP association rules mining-based modular reinforcement learning approach for multiagent systems. *IEEE Trans. Syst. Man Cybern. Part B*, 35(2): 326–338.

Liu, B.; Hsu, W.; and Ma, Y. 1998. Integrating Classification and Association Rule Mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, 80–86.

Longo, L.; Goebel, R.; Lecue, F.; Kieseberg, P.; and Holzinger, A. 2020. Explainable Artificial Intelligence: Concepts, Applications, Research Challenges and Visions. In *Machine Learning and Knowledge Extraction*, 1–16.

Menda, K.; Driggs-Campbell, K. R.; and Kochenderfer, M. J. 2019. EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'19, 5041–5048.

Moldovan, T. M.; and Abbeel, P. 2012. Safe Exploration in Markov Decision Processes. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12.

Puiutta, E.; and Veith, E. M. S. P. 2020. Explainable Reinforcement Learning: A Survey. In *Machine Learning and Knowledge Extraction, International Cross-Domain Conference*, CD-MAKE'20, 77–95. Springer.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M. I.; and Moritz, P. 2015. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML'15, 1889–1897.