ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Safe Exploration for Reinforcement Learning

Alexander Hans[1,2], Daniel Schneegaß[1,3], Anton M. Schäfer[1,4], and Steffen Udluft[1]

1- Siemens AG, Corporate Technology, Learning Systems,
Otto-Hahn-Ring 6, D-81739 Munich, Germany

2- Technical University of Ilmenau, Department of Neuroinformatics,
P.O.Box 100565, D-98684 Ilmenau, Germany

3- University of Luebeck, Institute of Neuro- and Bioinformatics,
Ratzeburger Allee 160, D-23538 Luebeck, Germany

4- University of Osnabrueck, Neuroinformatics Group,
Albrechtsstraße 28, D-49069 Osnabrueck, Germany

Abstract. In this paper we define and address the problem of safe exploration in the context of reinforcement learning. Our notion of safety is concerned with states or transitions that can lead to damage and thus must be avoided. We introduce the concepts of a safety function for determining a state's safety degree and that of a backup policy that is able to lead the controlled system from a critical state back to a safe one. Moreover, we present a level-based exploration scheme that is able to generate a comprehensive base of observations while adhering safety constraints. We evaluate our approach on a simplified simulation of a gas turbine.

1 Introduction

Reinforcement learning (RL) [1] is a type of machine learning to solve optimal control problems. The main goal is to find a policy that moves an agent optimally in an environment that is generally assumed to be a Markov decision process (MDP). In most RL tasks one is interested in maximising the return, i.e. the long-term reward accumulation.

While the application field of RL is manifold, in this paper we focus on the optimal control of industrial plants. Through simulations it could be shown that data-efficient RL methods are able to optimise complex industrial systems with a feasible number of interactions [2]. However, when the agent begins to interact with a real system, built for production use, the question arises if and how one can make sure that the exploration of the state-action space does not cause damage to the plant. For those environments a method is needed that does not only explore the state-action space, but does it safely.

In the following we propose a method for safe exploration. We introduce two fundamental components: a *safety function* to determine a state's degree of safety as well as a *backup policy* that is able to lead the system from a critical state back to a safe one (sec. 2). Using these two components, as it turns out, safe exploration is basically solved. Additionally, we present a level-based exploration scheme that guarantees a comprehensive base of observations and thus high-quality exploration results. In the remainder of this paper we follow the concepts and notations introduced in [1].

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

2 Safe Exploration

There already exist some contributions that deal with risk and uncertainty in RL. However, we consider safety in the sense of the existence of undesirable states or, more generally, transitions that must be avoided as they can lead to damage. On first sight, works as [3, 4, 5] seem to be similar to ours. However, if the MDP is deterministic, those approaches collapse to standard RL methods. Moreover, they assume that the system dynamics are known or they tolerate undesirable states during exploration. To the best of our knowledge, the only approach also explicitly covering exploration in safety-critical environments is apprenticeship learning [6], which relies on a teacher that must act safely and preferably near-optimally in order to achieve good policies.

We define (i) a transition (s, a, r, s') to be fatal if the corresponding reward r is less than a given safety threshold τ; (ii) an action a to be fatal in state s if it leads to a fatal transition (s, a, r, s') with positive probability; (iii) a state s to be supercritical if there exists no policy π that can, starting from s, guarantee that no fatal transition will ever occur; (iv) an action a to be supercritical in state s if it leads to a supercritical state s' with positive probability; (v) a state s to be critical if it is not supercritical and there is at least one supercritical or fatal action; (vi) an action a to be critical in state s if it is neither fatal nor supercritical but leads to a critical state s' with positive probability; (vii) a state s to be safe if it is neither critical nor supercritical; $(viii)$ an action a to be safe in state s if it is neither fatal, supercritical, nor critical, and (ix) a policy π to be safe if for all critical states s it leads to a safe state $s^{(n)}$ after a finite number n of non-fatal transitions $(s^{(i)}, \pi(s^{(i)}), r^{(i)}, s^{(i+1)})$ and applies only safe actions $a = \pi(s^{(i)})$ within safe states $s^{(i)}$.

Conventional exploration methods like random, ε-greedy, or Boltzmann exploration are not safe. Due to their random component of action selection, there is a certain chance of exploring fatal transitions. This is particularly unavoidable if the optimal policy is not safe. For an RL problem to be safely explorable, we require the following: (1) There exist safe policies w.r.t. the MDP and (2) the exploration starts in a safe state.

3 Necessary Components

In addition to the mentioned requirements, we identified two essential components for safe exploration: a safety function and a backup policy. To combine these, a level-based exploration scheme is presented.

3.1 Safety Function

Our objective is to never observe supercritical states. From a critical state the agent can return to a safe state if it acts safely. Prior to execution of an action its safety should therefore be estimated by means of a safety function. Thus, the safety function must give information on the safety of an action a in a state s.

In practical applications a safety function can hardly be specified in advance. Therefore, it is required to learn one from already collected exploration data. A possibility for this is to estimate the minimal reward $r_{\min}$ that would be observed when executing an action and afterwards following the backup policy (min-reward estimation). For this purpose, min-reward samples $(s, a, r_{\min})$, which depend on the backup policy π, are collected during exploration and used to estimate a min-reward function $R^{\pi}_{\min}(s, a)$. If its estimator's extrapolation capabilities are sufficient, it is possible to estimate $r_{\min}$ of not yet explored state-action pairs.

3.2 Backup Policy

The backup policy's task is to lead the agent back to an already known and safe area of the state space, i.e. an area with only safe states. It is activated when during exploration the agent reaches an unknown state and thus is unable to choose a safe action. It is important that the backup policy provides safe actions and hence does not lead to critical states. To take the agent back to a known area, it is useful that it tries to approach a stationary point or area.

For existing plants that are already operated without the use of RL methods often a basic controller is available. Provided this controller acts safely it may be used as a backup policy. Otherwise, one has to be learnt from observation data, i.e. (s, a, r, s') tuples.

Due to our assumption that fatal transitions are defined by the reward level, an obvious approach is to use conventional RL methods to generate a backup policy. However, an optimal policy is not necessarily a safe one. Therefore, we propose to learn the backup policy with an altered Bellman optimality equation that does not maximise the expected sum of rewards, but the minimal reward to come:

$$Q^{*}_{\min}(s, a) = \min_{s'} \min \left[R^{a}_{s,s'}, \max_{a'} Q^{*}_{\min}(s', a') \right]. \tag{1}$$

Using dynamic programming [1] one can calculate a (non-trivial) solution that fulfils equation 1 and derive a policy that acts greedily w.r.t. $Q^{*}_{\min}$.

3.3 Level-based Exploration Scheme

Utilising the safety function and the backup policy, the problem of safe exploration is essentially solved. However, to make the exploration as safe as possible, it seems advisable not to move arbitrarily through the state space, but to expand the explored area only gradually starting from a safe area. We use a level-based approach to realise this kind of gradual exploration. Every state is assigned a level l. For the starting state s_0 the level is set $l(s_0) = 0$. If a new state s_{t+1} is found resulting from exploring an action in state s_t, the level of the new state is set $l(s_{t+1}) = l(s_t)+1$. If a new state s_{t+1} is found while using the backup policy, the level is kept: $l(s_{t+1}) = l(s_t)$. Using this concept the exploration proceeds level-wise. Starting with level 0, the agent tries to explore all actions considered safe (by the safety function) of all states of the respective level. When there are

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

no more explorable actions for the current level, it is increased. This is repeated until there are no more explorable states left or some other abort criterion is met, e.g. a sufficient number of observations to derive a policy with aspired quality.

For this exploration scheme some sort of path planning is needed that allows the agent to reach a certain state with explorable actions left. A straightforward solution to this problem is a graph-based approach. While exploring, a graph is built. The graph's nodes represent the states, edges represent transitions between states, the action to trigger a transition is noted as edge label. Here, one can find paths to a certain state by means of a path search on the graph. Unfortunately, this approach is only suited for deterministic MDPs with a small number of states. To circumvent this problem one can aggregate states to clusters and use those as nodes. However, path searching on the new graph is not possible since there might be multiple edges for a specific state and action. As a solution we use standard RL for path planning: An MDP is defined with the clusters as states, actions are the same as for the original MDP, the reward function is $R^a_{s,s'} = 1$ if s' is a target state and $R^a_{s,s'} = 0$ otherwise. For the resulting MDP an optimal policy is determined and followed until a target state is reached. This approach is also applicable if the original MDP is stochastic.

4 Experiments: The BurnSim

For the analysis and evaluation of safe exploration we developed the BurnSim benchmark which is motivated by an actual problem in the context of gas turbine control. One is interested in operating a turbine with maximum power output. However, when operating at high output undesirable dynamics in the combustion chamber occur, i.e. a "humming" that can damage the turbine if becoming too heavy. Therefore, one should try to maximise the output while at the same time keeping the humming at a low level. The state space $S := \{(f,h) | f \in [0,1], h \in [0,\infty)\}$ includes fuel and humming, the available actions are $A := \{\text{decrease, keep, increase}\}$. The system dynamics are as follows:

$$
\begin{aligned}
f_{t+1} &= \begin{cases} f_t - 0.05, & \text{if } a = \text{decrease} \wedge f_t - 0.05 \geq 0 \\ f_t + 0.05, & \text{if } a = \text{increase} \wedge f_t + 0.05 < 1 \\ f_t, & \text{otherwise} \end{cases} \\
h_{t+1} &= \max\left(2 f_{t+1} h_t, \frac{f_{t+1}}{5}\right)
\end{aligned}
$$

The reward only depends on the successor state $s_{t+1} = (f_{t+1}, h_{t+1})$: $r_t = 4 f_{t+1}^2 - \left(\frac{h_{t+1} - f_{t+1}/5}{5}\right)^2$. The safety threshold is $\tau = -1$, hence $r_t \geq -1 \Rightarrow$ transition (s_t, a_t, r_t, s_{t+1}) is not fatal; $r_t < -1 \Rightarrow$ transition (s_t, a_t, r_t, s_{t+1}) is fatal. The exploration's aim is to gather enough data to determine a good policy without ever observing a reward less than -1. The particular difficulty with the BurnSim is that many states which seem safe in terms of the current reward are in fact supercritical because they inevitably lead to a fatal transition.

4.1 Implementation

We implemented a safety function, two variants of the backup policy (pre-specified and learnt), and exploration with graph-based as well as RL-based path planning. The safety function is determined by a local-quadratic least-squares estimation of the min-rewards, i.e. on the features $\Phi_i = (f_i, h_i, f_i^2, h_i^2, f_i h_i, 1)$, whenever a min-reward value for a state-action pair (s, a) must be estimated.

The pre-specified backup policy, which represents an already existing sub-optimal, but safe controller, chooses actions w.r.t. f: 'increase' if $f \leq 0.4$ and 'decrease' otherwise. In addition, a backup policy was learnt using the altered Bellman optimality equation (eq. 1). The observations used to determine the backup policy were generated by a teacher (inspired by apprenticeship learning [6]). During the subsequent autonomous exploration the backup policy was fixed.

The level-based exploration was implemented as explained previously. To be able to use the graph-based approach, the size of the MDP's state space is reduced by intrinsically rounding humming to 1/50 accuracy with each time step. For the approach that uses RL-based path planning the state space is left unchanged, but clustered by rounding humming accordingly. Fuel is already discrete as it is only changed by ± 0.05 and bounded by $[0, 1)$.

4.2 Results

Figure 1 compares the explored area of two experiments with the maximally explorable area. For the first experiment the pre-specified backup policy and graph-based path planning were used. The second experiment was conducted with a backup policy learnt from a teacher trajectory comprising 324 transitions and state clustering. In both experiments the exploration was safe and covered large parts of the state space. The results of the second experiment are not as good as those of the first one because the teacher trajectory was rather short and thus the resulting backup policy was not as safe as possible w.r.t. the min-reward. Consequently, actions were considered as unsafe unnecessarily early, resulting in a smaller explored area.

Nevertheless the observations of both experiments were sufficient to obtain good policies that were determined using dynamic programming (with nearest-neighbour generalisation for unknown states) as well as Neural Fitted Q-Iteration [7]. Table 1 shows the achieved average rewards and compares them to the optimal policy, which was obtained analytically.

	avg. reward	
RL method	graph-based	RL-based path planning
Dynamic programming	1.164	1.132
Neural Fitted Q-Iteration	1.166	1.166
Optimal policy	1.166	

Table 1: Performance of policies generated from observations of safe exploration. Neural Fitted Q-Iteration was able to identify the optimal policy due to better generalisation properties.

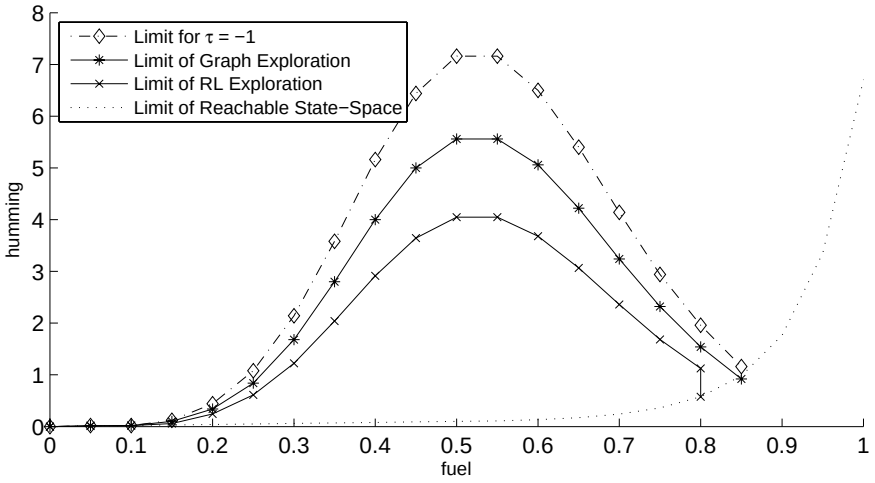

Fig. 1: Explored area of two different experiments compared to the maximum area that is safely explorable for $\tau = -1$. The area "below" a curve was explored. Due to system dynamics only states "above" the dotted line are reachable.

5 Conclusion and Future Work

In this paper we described the problem of safe exploration and pointed out possible solutions. Those can be seen as a starting point for autonomous exploration in industrial applications of RL. Unanswered questions include especially the analysis of the variant using state clustering for stochastic MDPs, dealing with high-dimensional state spaces, and error estimation of the safety function. Moreover, we aim to apply the method to a real plant.

References

[1] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[2] A.M. Schaefer, D. Schneegass, V. Sterzing, and S. Udluft. A neural reinforcement learning approach to gas turbine control. In *Proc. of the 20th Int. Joint Conf. on Neural Networks*, Orlando, 2007. MIT Press.

[3] M. Heger. Consideration of risk in reinforcement learning. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 105–111. Morgan Kaufmann, 1994.

[4] R. Neuneier and O. Mihatsch. Risk sensitive reinforcement learning. In *Proc. of the 11th Annual Conf. on Neural Information Processing Systems*, pages 1031–1037, 1998.

[5] P. Geibel. Reinforcement learning with bounded risk. In *Proc. of the 18th Int. Conf. on Machine Learning*, pages 162–169. Morgan Kaufmann, San Francisco, CA, 2001.

[6] P. Abbeel and A.Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proc. of the 22nd Int. Conf. on Machine Learning*, pages 1–8, 2005.

[7] M. Riedmiller. Neural fitted q-iteration – first experiences with a data efficient neural reinforcement learning method. In *Proc. of the 16th European Conf. on Machine Learning*, pages 317–328, 2005.