C1

REINFORCEMENT LEARNING WITH AVOIDANCE OF UNSAFE REGIONS

Jessica Vleugel (1366688), Michelle Hoogwout (1359118),

Koen Hermans (1359053) en Imre Gelens (1308750)

Summary - *We propose two new Reinforcement Learning algorithms that avoid unsafe regions of the process to be controlled. The one step-ahead algorithm uses a process model to predict the next state. It discards unsafe actions based on results of this prediction. The safety controller algorithm uses a safety controller that only overrules the learning algorithm when the agent reaches a safety margin. The two algorithms are applied to a Gridworld problem and results show that both algorithms avoid unsafe regions.*

1. Introduction

Reinforcement Learning (RL) is a generic machine learning approach that has its roots in cognitive psychology [1] and is used for (industrial) control and decision making applications [2]. By using rewards to reinforce desirable behaviour, RL is able to guide an agent to a goal while learning an optimal path through a state-space. The policy π that the agent follows determines the action $a = \pi(s)$ the agent takes at each specific state s resulting in a reward r. The rewards the agent receives are accumulated in a value function $V(s)$ or $Q(s, a)$ that is updated at each step. Since rewards get higher towards the goal, states that are in the optimal path will obtain a higher value over time, causing the agent to converge to this optimal path. Although the goal is to obtain an optimal solution, the agent should also explore new actions in order to keep learning and adapting to time-variant situations. RL can be useful in complex multivariable processes where optimal solutions are not easily calculated [3]. An issue that has remained unresolved, however, concerns the avoidance of unsafe regions.

In this paper, two algorithms to avoid unsafe regions (a set of unsafe states) are proposed and validated in an experiment. The first method uses a model to simulate the agent's actions and removes the actions that have shown to be unsafe. The second method makes use of a safety controller that overrules the learning algorithm and guides the agent back to the safe region. Note that prior knowledge is essential for both methods, because unsafe regions must be defined and the model or safety controller must be provided.

2. Methods

Both methods use a safety margin [4] in order to prevent the process from ending up in an unsafe state. The safety margin is needed to account for the system dynamics such as inertia and its size is mainly determined by these dynamic properties. In the first method, the one step-ahead algorithm, the margin is used to account for possible errors in the model, but also to guarantee the existence of an action that does not lead to an unsafe state. In the second method, the safety controller algorithm, the size of the safety margin is determined by overshoot and response time of the closed-loop controller. The size of the safety margin is determined before running the learning algorithm.

A. One Step-Ahead Algorithm

The one step-ahead algorithm simulates actions using a model $f(s, a)$ to determine what the next state s' will be and removes every action that could lead to an unsafe state from the action space. By repeating this loop until a safe action is found, the agent is guaranteed to avoid the unsafe states. When applying this method to continuous systems, one has to take into account that the action space is infinite. Therefore, rather than removing one action at a time, an interval of unsafe actions is removed from the action space. The policy then excludes the elements of this unsafe action space to avoid choosing the unsafe actions. The first method is shown in Algorithm 1 below.

Algorithm 1: One Step-Ahead Algorithm

Initialize s_0, Q_0
$t = 0$
while $s_t \neq s_{\text{goal}}$
  determine action space $A(s_t)$
    repeat
      $a_t \leftarrow \begin{cases} \text{argmax}_{a \in A(s_t)}\, Q(s_t, a) & \text{with probability } 1-\varepsilon \\ \text{random} & \text{with probability } \varepsilon \end{cases}$
      $s_t^{'} = f(s_t, a_t)$
        if $s_t^{'} \in S_u$
          Remove interval from $A(s_t)$
        else action=safe
        end
      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_t^{'}, a_{t+1}) - Q(s_t, a_t)\right)$
    until action=safe
  take action a_t, obtain reward r_t
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_t^{'}, a_{t+1}) - Q(s_t, a_t)\right)$
  $t \leftarrow t + 1$
end

To determine the unsafe actions, an alternative approach can be taken by using the inverse model f^{-1}. The equation $a = f^{-1}(s, s')$ is used to determine the action that takes the agent to a given state s'. Taking the collection of unsafe states S_u, as an input for the equation, the unsafe action space is determined. $A_u(s) = \{a \mid s' = f(s, a), s' \in S_u\}$.

B. Safety Controller Algorithm

The second method uses a safety controller C(s) that overrules the learning algorithm when the agent reaches the safety margin. The safety controller then directs the agent back to the safe region, where the learning algorithm takes over again.

The use of the safety controller is undesirable if it, e.g. uses an amount of energy that might have negative consequences for the system [4]. Although this is the case for many, but not all systems, an assumption was

made that using a safety controller is always undesirable. When no reward mechanism for the safety controller is in place the agent may incorporate the safety controller in its optimal path [4]. This is a result of the agent moving within the safety margin without any negative consequences. To avoid triggering the controller in the optimal solution, the agent is negatively rewarded for every step for which the safety-controller is used. The algorithm is shown below.

Algorithm 2: Safety Controller

initialize s_0, Q_0
$t = 0$
while $s_t \neq s_{\text{goal}}$
    if $s_t \in S_u$
        $a_t = \text{C}(s_t)$
    else
        $a_t \leftarrow \begin{cases} \text{argmax}_{a \in A(s_t)}\, Q(s_t, a) & \text{with probability } 1-\varepsilon \\ \text{random} & \text{with probability } \varepsilon \end{cases}$
    end
    take action a_t, obtain r_t
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$
    $t \leftarrow t + 1$
end

3. Validation Experiment and Results

To evaluate the performance of the safety-controller and one-step-ahead algorithms described above, they were applied to a discrete 2D Gridworld [4] with multiple unsafe regions. The task is to learn to efficiently reach the goal without ever reaching the unsafe states. The model used in the simulation is a 10 by 10 grid, with an action space that allows the agent to move up, down, left and right. This is shown in Figure 1, where the agent is a circle located at position (1,1), the goal at (9,9) and the rest of the circles are unsafe regions. Each step the agent takes is based on an ε-greedy policy [3], in which the policy of the controller or the one-step-ahead function can overrule the chosen step. Avoiding unsafe regions is the one criterion both algorithms have to uphold. It was desired that the number of times the safety algorithms interfere is minimal. The algorithms was simulated for 100 consecutive trials in each run (enough trials for the number of steps per run to converge), and repeated for 20 complete runs. Meanwhile, a record was made of the amount of 'deaths' (number of times an agent reaches an unsafe state), number of times the safety algorithms overrule, and the learning curve. Note that the one step-ahead algorithm does not have a safety margin in Gridworld due to the absence of the predict error and inertia.

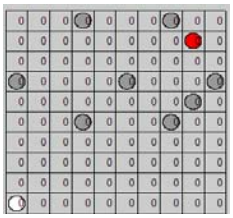

Figure 1- Used Scenario in Gridworld

A. Baseline Algorithm

Running the learning algorithm with an ε-greedy policy only and without safety mechanisms, provides a baseline for comparison. The baseline is a simulation with obstacles instead of unsafe regions, meaning that the agent cannot physically reach those regions. In order to compare the algorithms with normal RL it was assumed that running a normal RL with obstacles would be the same as running a perfect safety algorithm with unsafe regions. The one step-ahead and safety controller algorithms work best if their learning curve closely resembles that of the baseline [4].

B. One Step-Ahead Algorithm

The number of deaths using this algorithm proves to be equal to zero for all the trials, which satisfies the most important criterion. Figure 1 shows the average learning curve for 20 runs. It shows that this algorithm closely resembles the baseline, apart from a few peaks (caused by exceptionally long trials).

C. Safety Controller Algorithm

Using this algorithm, the number of deaths is also equal to zero. The safety controller algorithm converges to a less optimal path than the baseline does.

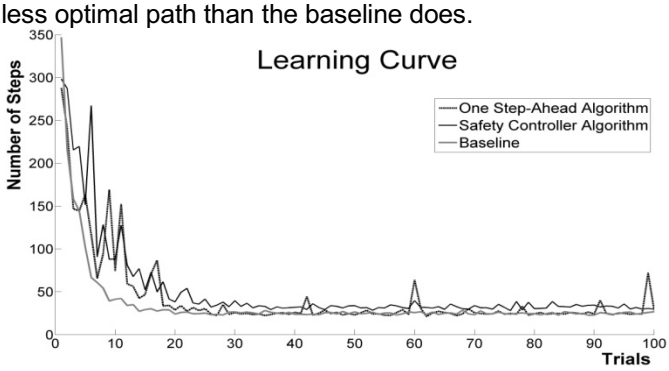

Figure 2- Learning Curve of All Three Algorithms

4. Discussion and Conclusions

Both algorithms satisfy the most important criterion of zero deaths and results show a similar number of times the safety algorithms are overruled. However, referring to the learning curve of both algorithms, neither one is optimal. The peaks in the results of the one step-ahead algorithm can be explained by the interaction between the one step-ahead algorithm and the ε-greedy policy, which sometimes results in exceptionally long trials. This can be reduced by changing the exploration rate and rewards [4]. Whether it is possible to completely eliminate this problem is a subject for further investigation. A problem with the safety controller is the convergence to a less optimal path. The reason this happens is that moving around a safety margin takes longer than moving closely past an obstacle. This happens in situations in which unsafe regions are located in the optimal path, as was proven through further testing [4].

The learning curve cannot be used as a form of comparison between the two algorithms and the baseline since its value cannot be solely attributed to the safety algorithms due to influential parameters. A choice between the one step-ahead and safety controller algorithms can therefore not be made based on Figure 2.

5. References

[1] G. D. Logan, *Cognitive Psychology*, in Elsevier, vol. 28, 1995
[2] R.S. Sutton and A.G. Barto, R.J. Williams, *Reinforcement Learning is Direct Adaptive Optimal Control,* Control Systems Magazine IEEE, pgs 19-22, vol 12, issue 2, 1992
[3] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998
[4] K. Hermans, M. Hoogwout, I. Gelens, J. Vleugel, *Robot Reinforcement Learning with Avoidance of Unsafe Regions*, Extended Research Report