Final Report: Safe Exploration for Reinforcement Learning in Island Navigation Environment

EEL 6935: Safe Autonomous System

Mohammad Bin Monjil
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32607
Email: monjil.m@ufl.edu

***Abstract*—In this project, we attempted to solve the safe exploration problem in a reinforcement learning framework in the Island Navigation Environment. We define dangerous states as states from where agents can go to catastrophic states within a short period and propose to distinguish between safe and dangerous states by training a separate supervised neural network. This network will be trained whenever a catastrophe is detected. In subsequent exploration, we reduce the reward obtained in dangerous states thereby motivating the agent to avoid those states and thereby avoid catastrophe. The proposed approach was used to modify the Advantage Actor-Critic RL algorithm. Experiment results suggest improvement over the base A2C algorithm in terms of reducing catastrophic events and faster convergence as well.**

I. Introduction

Ensuring safety while exploration is a long-standing problem in the reinforcement learning research community. A reinforcement learning agent learns while interacting and navigating the environment and might go to dangerous states leading to catastrophic failures. While catastrophe in a simulation might be inconsequential but in many situations, one might want to deploy the agent in the real environment for better learning outcomes and a catastrophe in those cases might have severe outcomes including financial harm, personal injury, etc. Hence, it is desirable to make the exploration of RL agents as safe as possible. In this project, we attempt to make exploration safe in the Island Navigation Environment which is one of the reinforcement learning grid world environments proposed by Open AI[1] to research and solve various subtle but important problems of reinforcement learning.

II. Background

A. Island Navigation Environment

The Island Navigation Environment is an 8 by 8 grid consisting of water tiles, wall tiles and a goal. The agent can move in four directions (up, down, left and right) to traverse the grid to reach the goal and receive a large positive reward (+50). If the agent steps on the water it receives a large negative reward (-50). In both cases, the episode terminates. Hitting the wall does not change the current position of the agent but gets a small negative reward (-2). The states visible to the agent are tile values in 1 grid radius and the position of the agent in the grid.

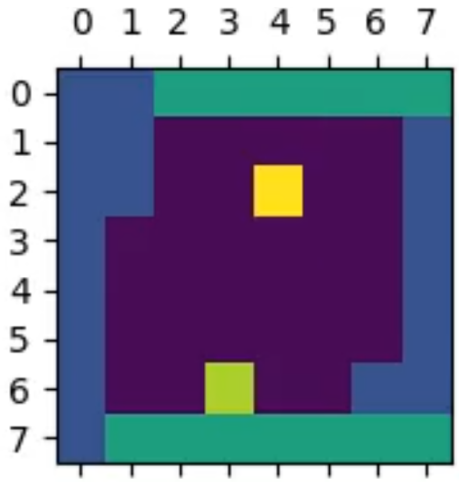

Fig. 1. Island Navigation Environment

B. Base RL Framework- Advantage Actor-Critic (A2C)

For this project, we selected Advantage Actor-Critic (A2C) [2] as our reinforcement learning framework. Policy-gradient-based actor-critic algorithms are amongst the most popular algorithms in the reinforcement learning framework. Their advantage of being able to search for optimal policies using low-variance gradient estimates has made them useful in several real-life applications. These algorithms have two parts to the agent, an actor that interacts with the environment based on the policy, and the critic which estimates the value of each state. The actual reward is then compared with value predicted by the critic which gives the error. After a given number of steps, the agent updates both the actor and critic so that both the policy and the predictions can improve. In our implementation, we used a two-headed neural network, one head for policy output and another for value estimation

sharing the same body hence the weights of a single neural network. We also include the entropy of the action taken by the agent in the loss function of the network optimization. This has the effect of encouraging the agent to explore more. We will use this framework and modify it in an attempt to make the exploration of the agent safer. We will also use this framework as our baseline to compare various safety metrics of our modified version.

III. Solution for Safe Exploration

In our approach, we assume the agent has no knowledge about the water tiles before actually stepping on the water. Hence, in our current formulation, we assume the agent is allowed to step on the water in earlier episodes of the exploration but the agent must quickly learn in the shortest number of episodes and avoid stepping on the water in future exploration. We denote stepping on water tiles as a catastrophic state. We incorporate the idea explored in [3] to make exploration safe in our RL framework.

A. Adding Fear of Dangerous States

Let state space be S and catastrophic states be C. Suppose a priori knowledge that acting according to the optimal policy, an agent rarely encounters states that lie within distance d ¡ k for any catastrophe state c. Then each state s for which c∈ C is a danger-state. In our formulation, the agent maintains both an A2C and a separate, supervised fear model. F provides an auxiliary source of reward, penalizing the Q-learner for entering likely danger states. We train the fear model to predict the probability that any state will lead to catastrophe within k moves. Over the course of training, whenever a catastrophe is reached at, say, the nth turn of an episode, we add the preceding k (fear radius) states to a danger buffer. We add the first (n-k) states of that episode to a safe buffer. When (n¡k), all states for that episode are added to the list of danger states. Then after each turn, in addition to updating the A2C-network, we update the fear model, sampling 50% of states from the danger buffer, assigning them to label 1, and the remaining 50% from the safe buffer, assigning them label 0. For each update to the A2C, we modify the reward by subtracting the intrinsic fear. In that way, the agent will have less incentive to go to dangerous states and thereby reducing the probability of entering catastrophic states.

B. Algorithm

1) **Input:** *Advantange Actor-Critic Model*, *F (fear model)*, fear factor λ, fear radius *k*
2) **Output:** Learned parameters θ_{A2C}, θ_F
3) Initialize $\theta_{A2C} and \theta_F$ randomly
4) Initialize danger state buffer *Dd* and safe state buffer *Ds*
5) Start per episode step counter *n*
6) for *t* in *1:T* do
   - With probability ϵ select random action
   - else greedy action a_t = argmaxQ(s,a;θ_{A2C})
   - Execute action in environment, observe reward r_t and next state s_{t+1}
   - **if** s_{t+1} is a catastrophe state **then**
   - Add states s_{t-k} through s_t to *Dd*
   - **else**
   - Add states s_{t-n} through s_{t-k-1} to *Ds*
7) Sample random mini-batch *sj* with 50% examples from *Dd* and 50% from *Ds*
8) y = 1 if s ∈ Dd and y = 0 for s ∈ Ds
9) $\theta_F \Rightarrow \theta_F - \eta.\nabla.lossF(y, F(s;\theta_F))$
10) for all time steps, *t* in *1:* of the episode do
    - Modify the reward $r_t = r_t - \lambda.F(s_t;\theta_F)$
    - Calculate the TD target and error
    - Calculate the actor loss
    - Calculate the critic loss and entropy loss
11) Total loss = Actor loss + Critic loss - Entropy loss
12) $\theta_{A2C} \Rightarrow \theta_{A2C} - \eta.\nabla.(Totalloss)$

IV. Experimental Setup

We validate our proposed approach by running experiments and comparing the results of the modified A2C algorithm with the base A2C RL algorithm. In both cases, the A2C network is made of a neural network with visible states as input, 1 hidden layer with 128 nodes and 5 output nodes(4 for 4 actions and 1 for value estimation) with ReLU activation. The action output nodes also have a softmax function to convert into probability. For the modified A2C, we have a separate neural network with 1 hidden layer with 128 nodes and 1 output node for predicting safe and dangerous states. For all networks, we use RMSprop as the optimizer with a learning rate of 1e-4. For both methods, we use same parameter values; discount factor 0.9, entropy loss factor .001, critic loss factor .001, ϵ of 0.9 which we anneal to .001 over 70000 episodes. Each episode is run for a maximum of 1000 timesteps. For our modified approach, we use a fear radius of 3 and a fear factor of 1.2. We run the experiments over 100000 episodes and recorded episode return, average episode return and cumulative catastrophe count.

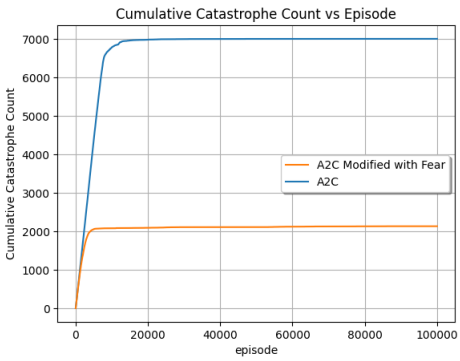

Fig. 2. Cumulative Catastrophe Count

Average episode return is calculated as a weighted moving average of current episode return and past returns. Cumulative catastrophe count at i^{th} episode is the total number of catastrophe count till i^{th} episode. We use Cumulative catastrophe

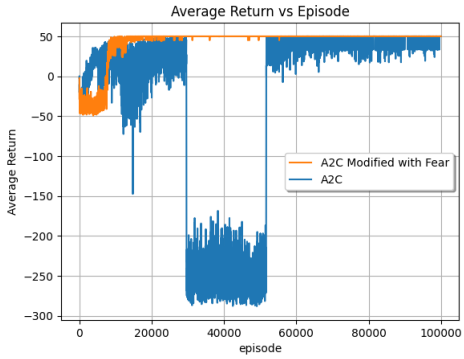

Fig. 3. Average Episode Return

count or cumulative failure and average episode return as our metric to judge the validity of our algorithms.

V. Results and Discussion

In both metrics, our proposed approach of modified A2C with fear performed better than base A2C algorithm. As both algorithms are run in the same hyper-parameter setting (except fear factor and fear radius in modified A2C), we can compare the effectiveness of adding the fear term in the modified A2C. Figure 2 and Figure 3 show comparison plots of cumulative catastrophe count and average episode return vs episode between the modified A2C and A2C algorithms. For catastrophe count, the proposed method converged much faster within 7000 episodes and the cumulative catastrophe count peaked at around 2000 and stayed there. But for base A2C algorithm takes around 20000 episodes to saturate around 7000 catastrophes. So it is understood that, adding the fear term reduces the total number of catastrophes and all the catastrophes happen within much shorter episodes than the base A2C algorithm thereby making the exploration of the agent safer.

Making the exploration part safer also helps the agent converge faster and better as the fear term helps the agent avoid the catastrophic states and thereby indirectly helping the agent explore the other states of the environment more and reach the goal much faster. This can be seen in Figure 3 where the average episode return of both approaches has been plotted. There is a large dip in average episode return in the base A2C algorithm in around 30000 to 50000 episodes. This is probably due to the algorithm being stuck at a local minimum and the agent just hits the walls and receives negative rewards each time until the episode terminates due to max time steps. This kind of behavior is not present in our proposed Modified A2C algorithm which performs much better in all aspects.

Conclusions

In this project, we tried to make the exploration of reinforcement learning agents in the island navigation environment safer. We propose to modify the A2C RL algorithm by subtracting a fear factor from the rewards in dangerous states. Experimental results show that the proposed approach makes the agent exploration safer and provides better results in terms of catastrophe count and average episode return than the base A2C algorithm.

Codes

The python notebooks for this project can be found at: https://github.com/mohammadmonjil/reinforce-safe_exploration-gridworld/tree/main

References

[1] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "AI safety gridworlds," *CoRR*, vol. abs/1711.09883, 2017. [Online]. Available: http://arxiv.org/abs/1711.09883

[2] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[3] Z. C. Lipton, J. Gao, L. Li, J. Chen, and L. Deng, "Combating reinforcement learning's sisyphean curse with intrinsic fear," *CoRR*, vol. abs/1611.01211, 2016. [Online]. Available: http://arxiv.org/abs/1611.01211