# Adversarial Windy Gridworld

Alicia Alarie[1] and Jonathan Wapman[2]

*Abstract*— **In this contribution, we extend the windy gridworld reinforcement learning problem by adding the wind as an adversarial opponent with its own reinforcement learning policy. During a typical windy gridworld problem, an agent attempts to move from a starting state to a destination state while avoiding obstacles, with added disturbances that modify the agent's predicted state change. Typically, the wind has fixed strength and direction. In this project, we allow the wind to change its direction with the goal of increasing the moving agent's time to reach the destination state, or by giving the moving agent a large negative reward through an action such as pushing it into a trap. We evaluate the SARSA and Q-Learning reinforcement learning algorithms for both the moving and windy agents. We also evaluate the effectiveness of pretraining the moving agent with an adversary, then changing the agent's environment.**

## I. INTRODUCTION

Often, a given environment cannot be completely described with a simple scenario of a computerized agent interacting with an outside environment. In the real-world, there are many situations where a RL actor must be aware of outside adversaries that will attempt to deliberately degrade its performance. Adversarial reinforcement learning has been used previously in several multi-agent environments. Our such reference is [1], which focuses on applying adversarial reinforcement learning to physics simulations where the adversary is allowed to change physical parameters such as the environment's coefficient of friction. Other papers, such as [2], apply adversarial reinforcement learning to a two-player soccer game, where both agents compete against one another following the same rules. Common adversarial strategies are explored in [3]. These include strategically-timed attacks with the goal of surprising the agent, and enchanting attacks, where the adversary attempts to lure the agent into an undesirable state. Finally, [4] contains the single-agent SARSA and Q-Learning algorithms and analysis of their performance in gridworld environments.

Our work presents the following contributions: 1) We derive adversarial forms of the SARSA and Q-Learning algorithms, 2) We analyze the performance of these algorithms in 2D gridworld environments, and 3) we experiment with applying these algorithms to changing environments, where we first train the agent in the original environment with and without the adversary, then change the environment to observe how well the agents adapt to the new environment with the goal of analyzing whether pretraining with an adversary produces faster relearning than pretraining with no adversary present.

[1]Department of Electrical and Computer Engineering, UC Davis, Davis, CA 95616, USA aalarie@ucdavis.edu

[2]Department of Electrical and Computer Engineering, UC Davis, Davis, CA 95616, USA jdwapman at ucdavis.edu

## II. MULTI-AGENT ALGORITHMS

With the addition of an adversary to the gridworld environment, the single-agent reinforcement learning algorithms are no longer adequate. Both the agent and adversary must now update their respective state-action pair values.

### A. SARSA

The first algorithm to modify is SARSA. The primary difference in this implementation is that the agent waits until after the adversary has modified the agent's position to evaluate its reward and update its state-action value. In Fig. 1, State 1 represents the agent's starting state and State 3 represents the agent's state after being modified by the adversary. The multi-agent algorithm is given in Algorithm 1.

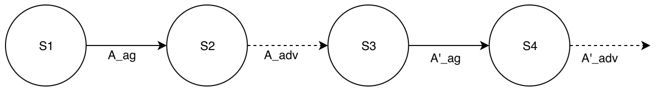

Fig. 1. Agent and Adversary State Transitions.

**Algorithm 1** Adversarial SARSA

```
Require:
    Algorithm parameters: step size $\alpha \in (0,1]$, small $\epsilon > 0$
 1: procedure SARSA($\alpha, \epsilon$)
 2:     Initialize $Q_{ag}(s,a)$, $Q_{adv}(s,a)$ for all $s \in S^+$, $a \in A(s)$, arbitrarily except that Q(terminal) = 0
 3:     for each episode do
 4:         Initialize $S_1$
 5:         while $S$ is not terminal do
 6:             Choose $A_{ag}$ from $S_1$ using $\epsilon$-greedy policy
 7:             Take action $A_{ag}$, observe $S_2$
 8:             Choose $A_{adv}$ from $S_2$ using $\epsilon$-greedy policy
 9:             Take action $A_{adv}$, observe R, $S_3$
10:             $Q_{ag}(S_1, A_{ag}) \leftarrow Q_{ag}(S_1, A_{ag}) + \alpha[-|R| + \gamma Q(S_3, A'_{ag}) - Q_{ag}(S_1, A_{ag})]$
11:             $Q_{adv}(S_2, A_{adv}) \leftarrow Q_{adv}(S_2, A_{adv}) + \alpha[|R| + \gamma Q(S_4, A'_{adv}) - Q_{adv}(S_3, A_{adv})]$
12:             $S_1 \leftarrow S_3$
```

### B. Q-Learning

Similar to the SARSA algorithm, the Q-Learning algorithm must be modified so that the agent's reward is evaluated after it is moved by the adversary, with its next state being the state it is moved to by the adversary. The modified algorithm is given in Algorithm 2.

**Algorithm 2** Adversarial Q-Learning

```
Require:
    Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
 1: procedure QLEARNING($\alpha, \epsilon$)
 2:     Initialize $Q_{ag}(s, a)$, $Q_{adv}(s, a)$ for all $s \in S^+$, $a \in A(s)$, arbitrarily except that Q(terminal) = 0
 3:     for each episode do
 4:         Initialize $S_1$
 5:         while $S$ is not terminal do
 6:             Choose $A_{ag}$ from $S_1$ using $\epsilon$-greedy policy
 7:             Take action $A_{ag}$, observe $S_2$
 8:             Choose $A_{adv}$ from $S_2$ using $\epsilon$-greedy policy
 9:             Take action $A_{adv}$, observe R, $S_3$
10:             $Q(S_1, A_{ag}) \leftarrow Q(S_1, A_{ag}) + \alpha[-|R| + \gamma * max_a Q(S_3, a) - Q(S_1, A_{ag})]$
11:             $Q(S_2, A_{adv}) \leftarrow Q(S_d, A_{adv}) + \alpha[|R| + \gamma * max_a Q(S_4, a) - Q(S_2, A_{adv})]$
12:             $S_1 \leftarrow S_3$
```

### C. *Environment*

The algorithms in this paper are tested using gridworld examples adapted similar to those in [4]. The rules of the environment are described below:

1) The agent can move in any coordinate direction (Up, Down, Left, Right) once per time step.
2) After the agent moves, the adversary can modify the agent's new position by one state up, down or right, with the constraint that it cannot return the agent to its previous position. It can also choose to take no action. (Note that left is not an action since this would prevent the agent from ever reaching the end state).
3) The agent receives a -1 reward for each time step when it has not reached a destination state (adversary receives +1).
4) The agent receives a -100 reward when it enters a trap. Entering a trap resets the agent to its starting state (adversary receives +100).

In this environment, the agent's states are its (row, column) coordinates and its actions are (Up, Down, Left, Right). The adversary's states are the agent's current (row, column) coordinates and the agent's previous action, giving (row, column, agent action) as the adversary state values.

## III. Results

### A. *2-Dimensional Simultaneous*

The gridworld used in this analysis is shown in Fig. 2 with the learned multi-agent paths for the SARSA and Q-Learning algorithms overlayed. The agent starts from the blue state at (5,1) and terminates at (5,9). The red square at (5,5) is a trap which transitions the agent back to its start state at (5,1) with a reward of -100. Otherwise, rewards at each time step are -1. Agent moves are shown with blue arrows, and the adversary's moves are shown with red arrows In this example, the agent using SARSA tends to avoid the trap much more effectively than the agent using Q-learning. Because of this, the SARSA adversary tends to focus more on increasing the number of steps the agent must take by continuously pushing it down (with the agent responding by taking an extra step up to avoid nearing the trap). In contrast, the Q-Learning agent tends to travel much nearer to the trap, so the adversary is more likely to attempt to push the adversary into the trap.

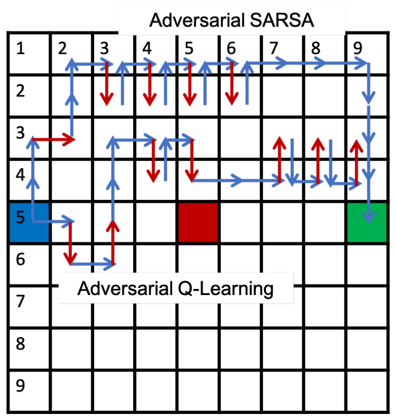

Fig. 2. 2D Gridworld with Single Agent (100 runs).

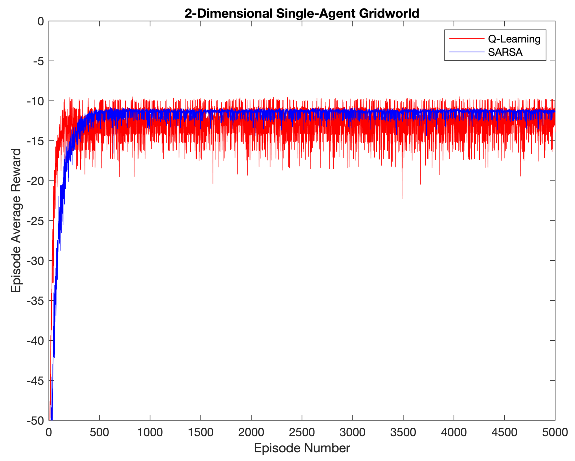

Fig. 3. 2D Gridworld with Single Agent (100 runs).

Performance of the agent without the adversary present can be seen in Fig. 3. The algorithms perform as expected, with Q-Learning performing slightly worse than SARSA since it tends to prefer travelling closer to the trap, similar to the cliff gridworld example in [4]. In contrast, Fig. 4 shows the average rewards over time with the agent and adversary trained simultaneously in the previously-described gridworld environment. Here, the adversary is able to lower the agent's average returns over time, visualized by the downward slope in the plots. Additionally, the adversary is able to degrade the agent's performance much more significantly using Q-Learning, since it takes advantage of Q-learning's tendency to travel closer to the trap (the ”optimal” route in the case where there is no adversary and $\epsilon = 0$), and when the agent travels closer to the trap, the adversary has a higher chance of being able to push it into the trap.

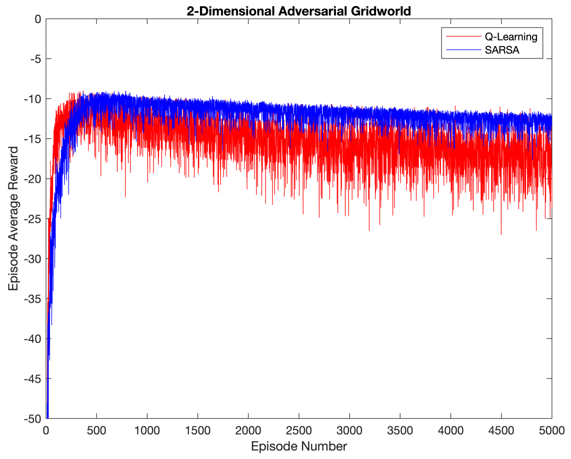

Fig. 4. 2D Gridworld with Simultaneous Training (100 runs).

## B. *Performance Across Environments*

After an optimal policy was learned in the environment from Fig. 2, all four agents for Q-learning and SARSA with and without being exposed to an adversary were placed in the new environment shown in Fig. 6. The new environment contains constant wind forces and cliffs in a different configuration than the first environment where the cliff state is now the goal state. The cumulative rewards of each agent were tracked after being placed in the new environment and are shown in Fig. 5. As in the results from training in the first environment, the Q-learning agents converge to an optimal path slightly faster than the SARSA agents. In this case, all agents converge on the same optimal path. Additionally, the agents with prior exposure to adversarial wind agents were able to adapt to the new environment faster than the agents that had been trained without the adversary.

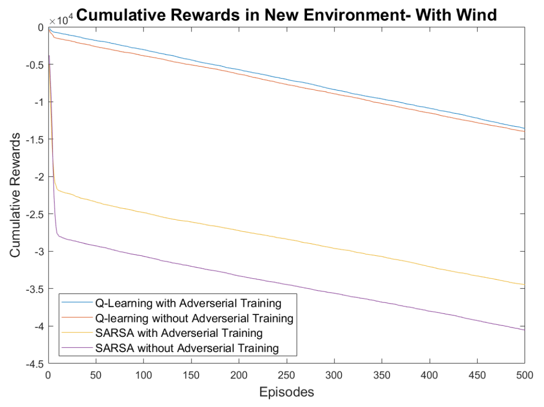

Fig. 5. Cumulative Rewards in a new windy environment for agents trained in a first environment with and without an adversary present (20 runs).

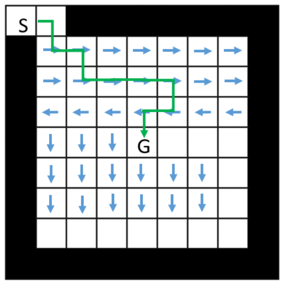

Fig. 6. New Environment agents were placed in after training in the first environment from figure 2. Black spaces indicate a ”cliff” with a reward of -100, and blue arrows indicate a fixed ”wind” which adds one movement in the direction of the arrow when the space is landed on. The optimal path is shown in green.

Performance in the new environment without wind in the same environment was also tested and similar results were observed. Fig. 7 shows the cumulative results observed in this scenario. The Q-learning agents converged on the correct path more quickly than SARSA agents and the agents that had been trained with adversaries converged on the quickest path sooner than their counterparts that had not been trained with the adversary present. However the difference in performance between agents trained with and without adversaries was smaller than in the other environments. This is expected because the adversary acts as a simulation of a dynamic wind force.

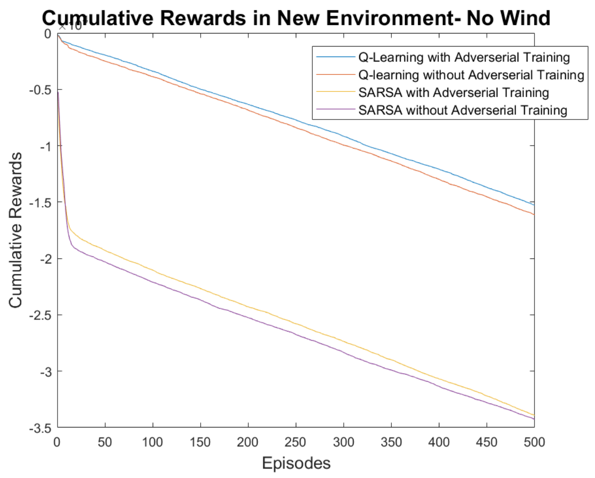

Fig. 7. Cumulative Rewards in a new windy environment for agents trained in a first environment with and without an adversary present (20 runs).

Promising results were also observed in the new environment with wind, but without any cliffs. Cumulative rewards for this case are shown in Fig. 8 and the environment setup is shown in Fig. 9. In this environment, Q-learning still outperforms SARSA in terms of hitting the optimal path the quickest, but there is less difference in performance between the Q-learning agent trained with the adversary and without the adversary present in the first environment.

Similar results were observed when the agents were placed in a different new environment without cliffs and where the

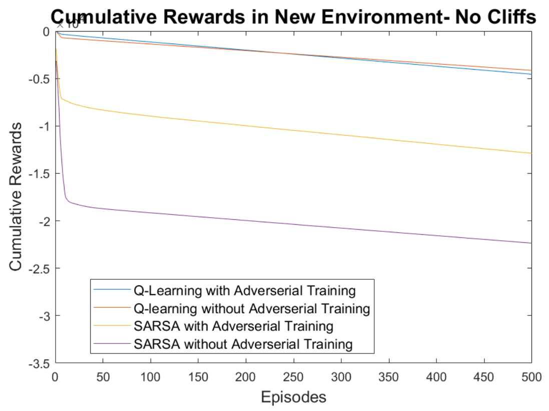

Fig. 8. Cumulative Rewards in a new windy environment with no cliffs for agents trained in a first environment with and without an adversary present (20 runs).

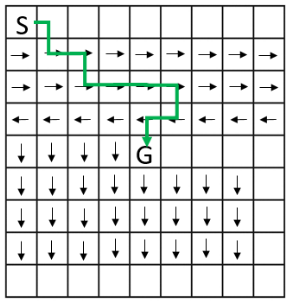

Fig. 9. New Environment agents were placed in after training in the first environment from Fig. 2. Blue arrows indicate a fixed ”wind” which adds one movement in the direction of the arrow when the space is landed on. The optimal path is shown in green.

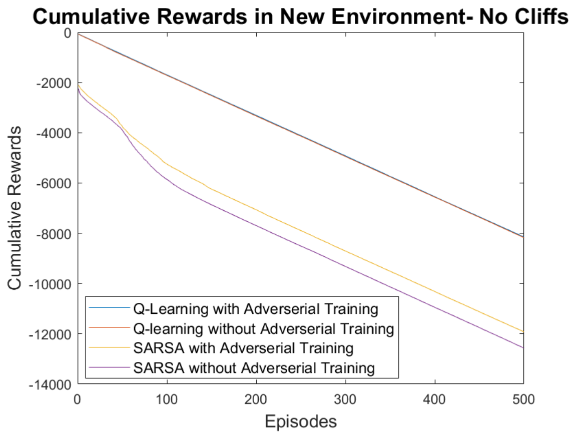

Fig. 10. Cumulative Rewards in a new windy environment with no cliffs for agents trained in a first environment with and without an adversary present (20 runs).

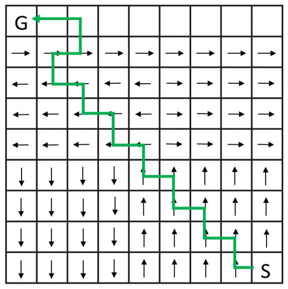

Fig. 11. New Environment agents were placed in after training in the first environment from figure 2. Blue arrows indicate a fixed ”wind” which adds one movement in the direction of the arrow when the space is landed on. The optimal path is shown in green.

orientation of the start and goal states was reversed. As shown in Fig. 10, both Q-learning agents were able to adapt more quickly than the SARSA agents, and the SARSA agent that had been trained with the adversary adapted quicker to the changed environment shown in Fig. 11. In general the Q-learning agents achieved higher cumulative rewards without the cliff present since the performance of Q-learning is degraded by taking a path closer to the cliff spaces.

## IV. CONCLUSIONS

As shown in this work, adversarial reinforcement learning can be both very effective at destabilizing an agent's optimal path, and can also be a useful method of training an agent to be more robust. First, we derived multi-step, adversarial forms of the SARSA and Q-Learning algorithms, which are particularly useful in environments where the agent's next state and reward are not known until the adversary has also acted. Simulation on a simple gridworld environment shows that the adversary can effectively reduce the agent's average reward over time. (One important note is that the amount of decay is highly dependant on the environment). Finally, analysis of these adversarial algorithms applied to a changing environment shows that training the agent with an adversary in the original environment allows the agent to more quickly adapt to the new environment. This behavior has many real-world applications in scenarions such as self-driving vehicles (which often need to account for windy conditions) or competitive games. In general, this paper shows that adversarial reinforcement learning can be highly effective at destabilizing a reinforcement learning agent, but also that training such an agent against an adversary can lead to notable performance improvements.

## REFERENCES

[1] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, Robust Adversarial Reinforcement Learning, arXiv:1703.02702 [cs], Mar. 2017.
[2] W. Uther and M. Veloso, Adversarial Reinforcement Learning, p. 18.
[3] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, Tactics of Adversarial Attack on Deep Reinforcement Learning Agents, arXiv:1703.06748 [cs, stat], Mar. 2017.
[4] R. S. Sutton and A. Barto, Reinforcement learning: an introduction, Second edition. Cambridge, MA London: The MIT Press, 2018.