GradSlam + RL Assignment, Winter 2020

Abhishek Jain, *MS in Robotics Engineering, Worcester Polytechnic Institute*

Abstract

The report contains results and discussion of a *slight modification in the reward function* for implementation of Reinforcement Learning(RL) algorithms on a custom gym (SEVN:Sidewalk Environment for Visual Navigation) based environment: *SEVN-Test-AllObs-Shaped-v1*

I. Introduction

The given environment is based on *SEVN*, a visually realistic Deep Reinforcement Learning (DRL) simulator for pedestrian navigation in a typical urban neighborhood. The state space of the agent contains 4 types of data: panorama Images, GPS data, street names, house numbers. The action space comprises of a discrete set of 5 actions: [sharp left(-67.5°), slight left(-22.5°), forward, slight right(+22.5°), sharp right(+67.5°)]. An episode is marked as complete if the agent either attains the goal configuration or reaches the maximum limit of steps(253).

II. Motivation

On careful observation of the performance of the best performing RL agent(PPO) in the given environment, it was found that the agent successfully learns to reach the goal node but at times fails to orient itself in the correct heading. The reward function of the environment was studied and it was found that if the agent is very close to the goal configuration, it no longer receives a +0.1 or +1 reward for moving closer to the direction of the goal. This could be the possible reason for the agent to not successfully find the terminal state as it reaches very close to the goal. This provided a motivation to experiment incentivizing the agent to reach the terminal state by giving a higher reward for attaining the goal configuration.

III. Reward Function

State	Original	Modified
Moving towards goal position	+1	+1
Moving away from goal position	-1	-1
Turning in direction towards goal	+0.1	+0.1
Turning away from direction towards goal	-0.2	-0.2
Reaching goal configuration	+0.1	**+10**

TABLE I: Original and Modified reward functions of the environment setup

Table I shows the original and modified reward functions for the given environment. There was just a slight modification in the reward that was setup for successfully reaching the goal configuration. Under the modified regime, the agent was given a +10 reward for successfully completing the trajectory against a +0.1 in the original setup.

IV. Results and Discussion

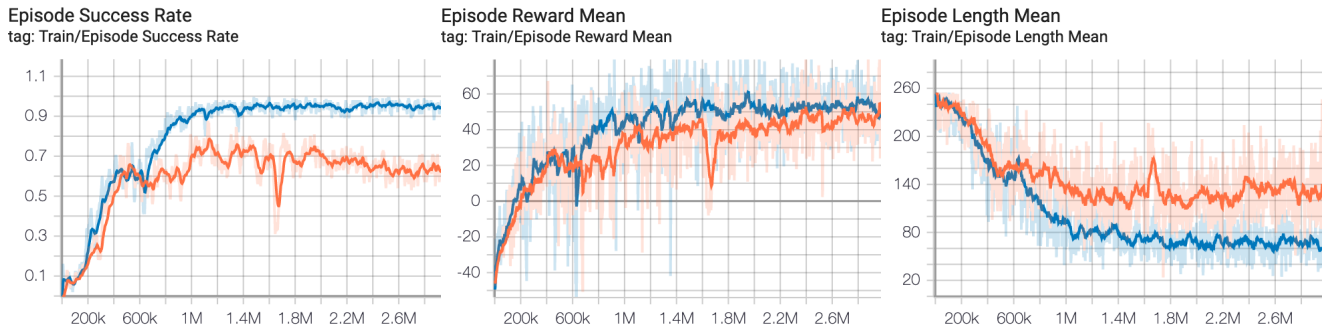

Fig. 1: Training graphs for Original(Orange) and Modified(Blue) reward function

Figure 1 shows the comparison of the training graphs for the original and modified reward function. The agent under both the regimes was trained for 3 M episodes. While the agent under the original regime reaches a top success rate of $\sim 90\%$ and declines to reach a stable success rate of $\sim 65\%$, the agent under the modified regime registers a much better performance both in terms of stability and success rate as it smoothly continues to grow to reach $\sim 99\%$ success rate.

Reward Function	Mean Reward	Mean Success rate	Mean Trajectory Length
Original	43.89 (±37.33)	57.6% (±20.14)	139.897 (±40.22)
Modified	43.11 (±35.11)	**92.4% (±7.33)**	**70.168 (±20.12)**

TABLE II: Test Results for Original(Orange) and Modified(Red) reward function

Table II shows the test results for both the reward functions for the policy after 3 M steps of experience. The results were averaged over 10 different seeds with 100 episodes per seed. The standard deviation is across the seeds and the evaluated episodes. While the difference in the total reward earned in an episode was expected, there is a significant improvement in the success rate as well as the episode length if the modified reward function is used. This shows that incentivizing the agent to reach the terminal state by giving a high reward for attaining the goal configuration drives the agent to successfully navigate to the goal state and avoid getting stuck in the episode.

V. Additional Results

The modified reward function was also tested with other agents and environments. A D3QN(Double Dueling Deep-Q Network) agent was trained on the same environment (SEVN-Test-AllObs-Shaped-v1) while the PPO agent was trained on the SEVN-Train-AllObs-Shaped-v1 environment to check the impact of the change in the reward function.

A. *D3QN for SEVN-Test-AllObs-Shaped-v1 environment*

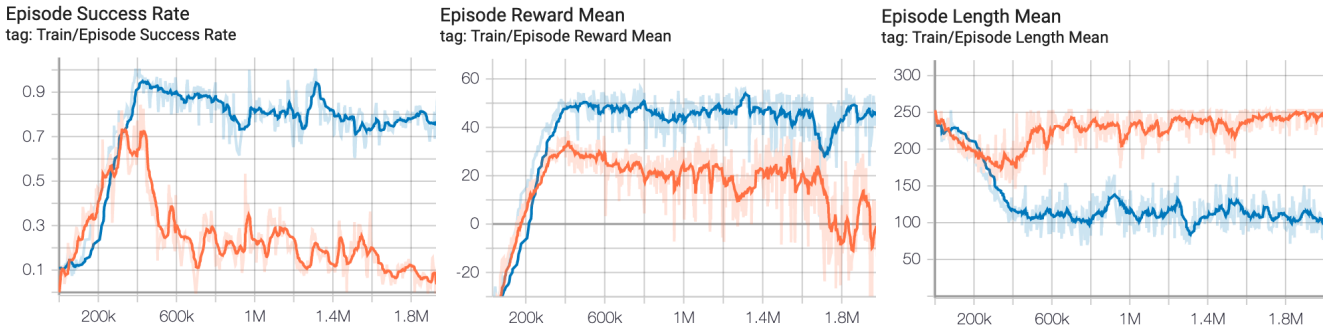

Fig. 2: Training graphs for Original(Orange) and Modified(Blue) reward function for D3QN agent

Figure 3 shows the comparison of the training graphs for the original and modified reward function for the D3QN agent. Clearly, there is significant improvement in the performance of the D3QN agent where previously under the original reward function the agent showed a huge dip in the performance with epsilon-greedy policy as the value of epsilon(a hyper-parameter for exploration) decayed to zero (in 2 M steps). Whereas in the case of the modified reward function, the agent learns to reach the goal configuration more often ($\sim$ 80% success rate) even after the epsilon decays to zero.

B. *PPO for SEVN-Train-AllObs-Shaped-v1 environment*

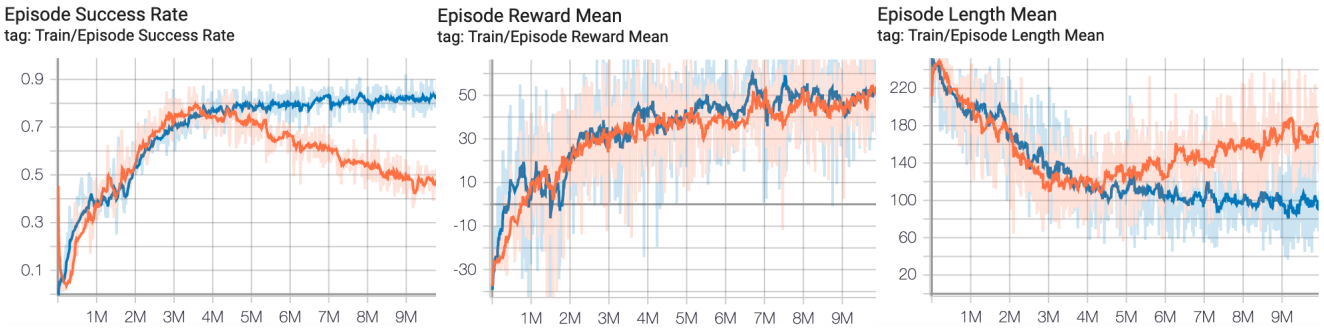

Fig. 3: Training graphs for Original(Orange) and Modified(Blue) reward function for PPO agent on SEVN-Train-AllObs-Shaped-v1 environment

The modified reward function was also tested with the PPO agent on a different environment (SEVN-Train-AllObs-Shaped-v1 environment). Here, the agent was trained for 10 M steps. Similar results were observed as in the case of the given environment signifying that the modified reward function helped in achieving better results.