Deep RL Arm Manipulation Writeup

Shingo Mori

June 27, 2018

This writeup addresses the rubric points for the fourth project of the Udacity Robotics Software Nanodegree Program Term 2.

1 Introduction

In this project, a robotic agent is created in a Gazebo environment and trained to accurately touch an object of interest within a certain amount of movements. The training is done in a reinforcement learning approach [1]. The Deep Recurrent Q-Network (DRQN) [2], a variant of the Deep Q-Network (DQN) [3], is applied to determine the agent's action policy.

The main tasks in this project is to define a reward function and tune hyperparameters to carry out the following two primary objectives,

1. Have any part of the robot arm touch the object of interest, with at least a 90% accuracy for a minimum of 100 runs.
2. Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy for a minimum of 100 runs.

The environment is composed of an object of interest, a robotic arm to touch the object and a camera to observe the state of the environment. An image of the environment setup is shown in Figure 1.

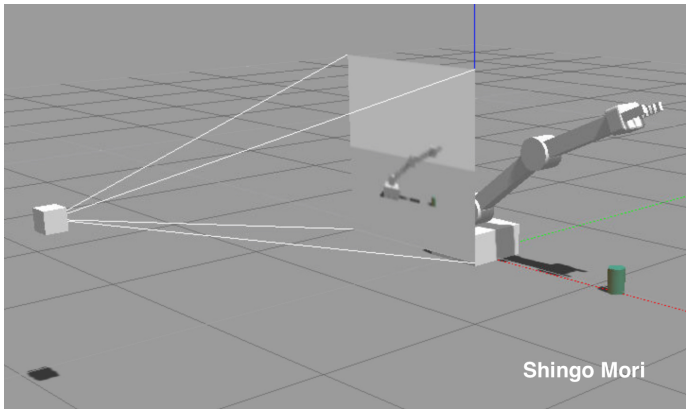


Figure 1: Gazebo environment setup.

2 Reward Function

2.1 Action Command

The arm has three revolute joints which are controlled by the agent sending an action command every frame. There are two options supplied for what type of action command to send. We can either send a velocity-control action command to specify the angular velocity of a joint, or send an angle-control action command to specify the angle offset of a joint. The latter is selected in this assignment to achieve an accurate joint control.

The angle-control command has two directions, increase or decrease. Hence, the number of actions the agent can select sums up to 6 (3 joints x 2 directions).

2.2 Reward Value

The agent selects one of the actions every frame. The selection is made by the action-value function of the DRQN or simply by a random choice based on an ϵ-greedy exploration [3]. For each frame the parameters of the DRQN are updated using a reward value calculated by the reward function.

The reward function takes as input the last state of the environment and outputs the corresponding reward value. We define winning and losing scenarios in which the function returns a reward value of $r_w = 100$ and $r_l = -10$, respectively. For the first objective, the agent wins when any part of the robot arm touches the object. For the second objective, the agent wins when only the gripper base of the robot arm touches the object, and loses if any other part touches it. In the both cases, the agent loses when any part of the arm hits the ground or the number of frames elapsed exceeds the limit value (100). If the agent wins or loses, the episode (sequence of frames) ends and a new episode starts with the environment reset to the initial state.

For each frame the function returns a interim reward based on the latest state of the agent. The function returns a positive reward if the gripper base is approaching the object, and returns a negative reward proportional to the number of frames elapsed normalized by the maximum number of episodes. This encourages the agent to move towards the object and end the episode as fast as possible. Additionally, a negative reward is returned if the amount of change in the distance between the gripper base and the object is less than 10cm. This prevents the agent from selecting an action that does not yield any progress. All the reward is multiplied by a certain coefficient. The following pseudo code shows the process of calculating the interim reward.

```
ALPHA = 0.4
delta = lastGoalDistance - currentGoalDistance
avgDelta = avgDelta * ALPHA + (delta * (1 - ALPHA))

reward = 2.0f * avgDelta - 0.5 * currentEpisodeFrame / maxEpisodeFrame
if abs(avgDelta) < 0.01:
        reward -= 0.5
```

3 Hyperparameters

List of hyperparameters and corresponding descriptions are shown in Table 1.

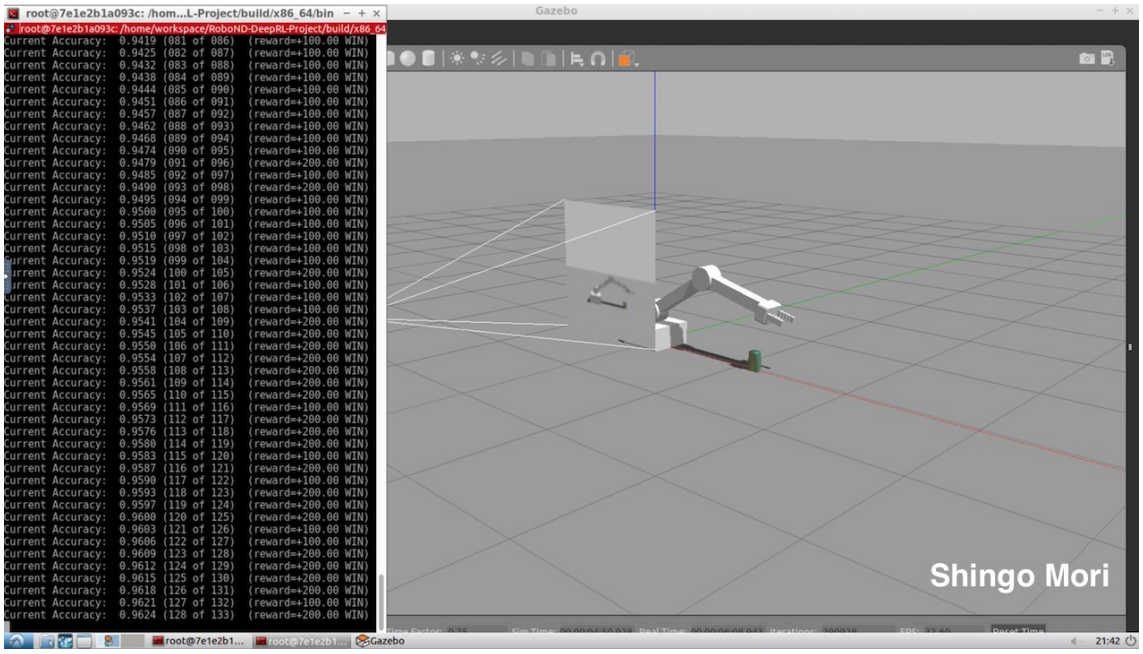
4 Results

The reward function and hyperparameters are configured either for the first objective or for the second objective before conducting each experiment. With the configuration for the first objective, the accuracy of the arm touching the object recorded 96% for 125 episodes, which meets the requirement. With the configuration for the second objective, the accuracy of only the gripper base touching the object recorded 81% for 390 episodes, which also meets the requirement. The screen-shot of the results are shown in Figure 2.

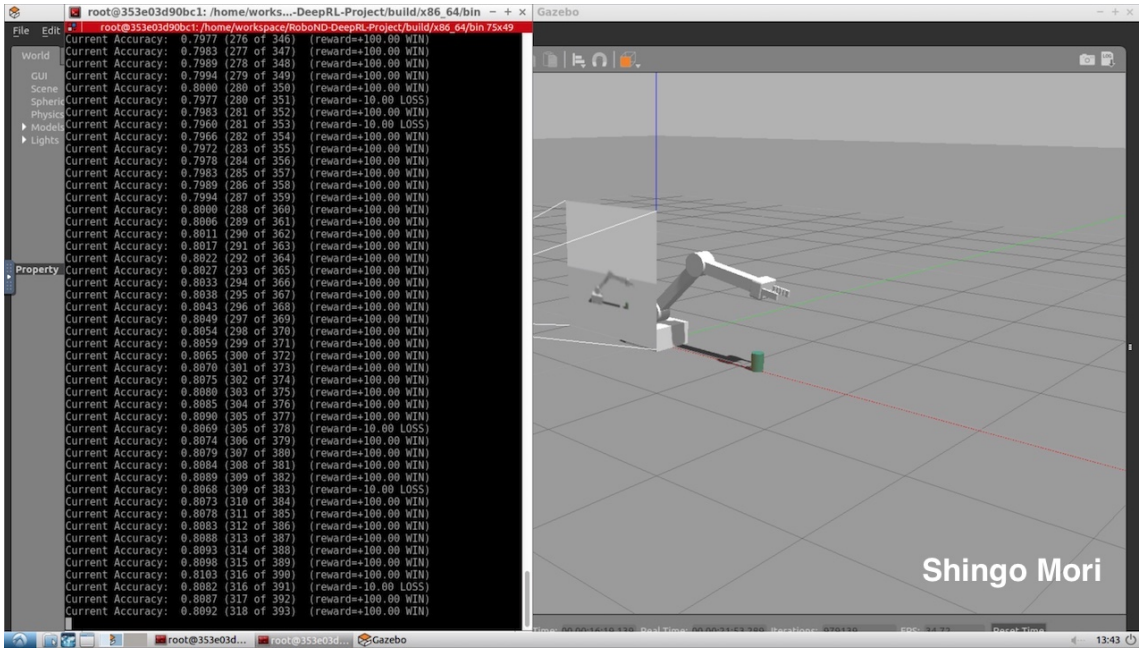
Due to the large learning rate, the agent finds an optimal route quite quickly (within 20 episodes at the fastest). However, the performance after the convergence highly depends on the experiences in the early frames. Because the Adam optimizer decays its effective learning rate over time, experiences in the later frames does not change the network parameter as much as those in the earlier frames. This causes the overall accuracy very low ($< 60\%$) when the agent fails to find an optimal solution in the earlier stage. Changing the optimizer and learning rate and/or tuning the parameters for ϵ-greedy exploration might solve this problem.

Hyperparameter	Value	Description
GAMMA	0.9 (0.9)	Discount factor used in the Q-lerning [4] update. Lowering the value makes the agent more greedy. The default is kept to prevent from not valuing the future rewards.
EPS_START	0.9 (0.9)	Initial value of ϵ in ϵ-greedy exploration. Unchanged to encourage the agent to explore in a wide range during the early frames.
EPS_END	0.001 (0.05)	Final value of ϵ in ϵ-greedy exploration. Lowered significantly. Since the environment is deterministic, random explorations are no longer needed once the agent discovers a path.
EPS_DECAY	(200) 100	Number of frames over which ϵ decays. The EPS_DECAY value for the first objective. The agent explores enough within 100 episodes for the first objective.
	150	The EPS_DECAY value for the second objective. The agent explores enough within 150 episodes for the second objective.
INPUT_WIDTH	64 (512)	Image width of the camera.
INPUT_HEIGHT	64 (512)	Image height of the camera.
LSTM_SIZE	256 (32)	Number of frames fed to LSTM [5] in the DRQN model.
OPTIMIZER	"Adam" ("None")	The name of the optimizer used for training. Adam [6] is used to allow higher learning rates.
LEARNING_RATE	0.1 (0.0)	Learning rate used by the optimizer. A large value is set to speedup the learning process.
REPLAY_MEMORY	10000 (10000)	Number of recent frames stored for experience replay [3]. The default value is enough to ensure randomness in the sample distribution.

Table 1: List of hyperparameters



(a) The result for the first objective.



(b) The result for the second objective.

Figure 2: A screen shot of the result for each objective.

5 Future Work

Although the agent learns an optimal route quickly, it fails to find one if it does not explore enough in the earlier frames. This is due to the nature of ϵ-greedy exploration. The EPS_DECAY value can be increased to allow the agent for longer exploration. However this trades off the convergence speed. Future works include addressing the problem of dependence on earlier frames by changing the EPS_DECAY, OPTIMIZER, LEARNING_RATE or other parameters. One may try to even change the reward function and/or the network architecture.

Future works also include dealing with additional challenges such as randomizing the object position and/or increasing the arm's degree of freedom.

References

[1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[2] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," 2015.

[3] Y. Zhan, H. B. Ammar, and M. E. Taylor, "Human-level control through deep reinforcement learning," 2016.

[4] C. J. C. H. Watkins and P. Dayan, "Q-learning," in *Machine Learning*, pp. 279–292, 1992.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.