Deep RL ARM Manipulator

Pramod Kumar

1 Introduction

This project about the robotic arm manipulation using rainforce learning. A agent is created based on Deep Q-learning Network (DQN). The two objectives of robot is required to achieve the goal. first, robotic arm would touch the object with atleast 90% accuracy. Second, robotic gripper would touch the object with atleast 80% accuracy. Robot learn using deep rainforce learning algorithm. reward functions and hyperparameters is tuned to achieve the goal.

2 Reward functions

There are two objectives of this project to achieve the goal:

2.1 Objective 1:

First objective of the this project is to touch the any part of ARM to the object. The velocity control approach is used to achieve this goal and velocity control is checked with the even or odd action. based on this even or odd action, actionVelDelta is added or subtracted. Positive or negative rewards define as numeric constant REWARD_LOSS and REWARD_WIN. REWARD_LOSS and REWARD_WIN is adjusted to avoid the ARM collision. When arm touching the ground level then issued a loss and also exceeds the 100 length steps. Since it's using velocity control to complete the first objective, the reward function setup for the process in between to control the robotic arm's movement will defer to the one that will be used for the second objective. average delta is calculated just as stated in the tasks lesson.

```
float distDelta = lastGoalDistance - distGoal ;
avgGoalDelta = (avgGoalDelta * ALPHA) +
               (distDelta * (1.0f *ALPHA));
```

During simulation velocity control is enabled and make linear function proportional to the moving average. negative reward is proportional to the goal distance.

```
rewardHistory = INTERIMREWARD * avgGoalDelta ;
```

2.2 Objective 2:

Second objective of this project is to touch the gripper of robot to the object. The position control approach is used to achieve this goal and position control is checked with the even or odd action. based on this even or odd action, actionJoinDelta is added or subtracted. Positive or negative rewards define as numeric constant REWARD_LOSS and REWARD_WIN. REWARD_LOSS and REWARD_WIN is adjusted to avoid the gripper collision. Robot arm should not collide with object it touch it loss the REWARD_LOSS with 1*20 and if griiper touch with object then it get 1*100 for REWARD_WIN.

The average delta is calculated just as stated in the tasks lesson.

```
avgGoalDelta = (avgGoalDelta * ALPHA) +
               (distDelta * (1.0f *ALPHA));
```

Smooth average moving is used to achieve the goal.

```
if(distGoal > 0.0f){
rewardHistory = REWARD_WIN * avgGoalDelta;
}
else if (distGoal == 0.0f){
 rewardHistory = REWARD_WIN * 10.0f;
}
if(distGoal <= 0.0f){
rewardHistory = REWARD_LOSS * distGoal;
}
```

This approach encourage the movement of robotic arm so if it move more then it will get more reward points. if it distance is zero for robotic arm then it will get 10 times more rewards. If it is not moving towards object the it will loss rewards distGoal times.

3 Hyperparameters

The parameters have tuned based on experiments and have tried based on some inputs from udacity slack community.

TABLE 1
Hyperparameters

Hyperparameters	
Objective 1 Parameters	Objective 2 Parameters
#define INPUT CHANNELS 3	#define INPUT CHANNELS 3
#define ALLOWRANDOM true	#define ALLOWRANDOM true
#define DEBUGDQN false	#define DEBUGDQN false
#define GAMMA 0.9f	#define GAMMA 0.9f
#define EPS START 0.7f	#define EPS START 0.7f
#define EPS END 0.02f	#define EPS END 0.02f
#define EPS DECAY 200	#define EPS DECAY 200
#define INPUTWIDTH 64	#define INPUTWIDTH 64
#define INPUT HEIGHT 64	#define INPUT HEIGHT 64
#define OPTIMIZER "RMSprop"	#define OPTIMIZER "Adam"
#define LEARNING RATE 0.1f	#define LEARNING RATE 0.1f
#define REPLAYMEMORY 10000	#define REPLAYMEMORY 10000
#define BATCH SIZE 32	#define BATCH SIZE 32
#define USE LSTM true	#define USE LSTM true
#define LSTM SIZE 64	#define LSTM SIZE 256

4 Results

Both objectives is achieved and shown in figure 1 and 2. The better performance is achieved with linear reward function. Here it observed that higher the accuracy if higher the alpha value. Objective 1 get achieve better at REWARD_WIN 10 and for objective 2 it is 20 as well REWARD_LOSS also with same negative value. DQN value slightly decrease for objective 1 compared to objective 2.

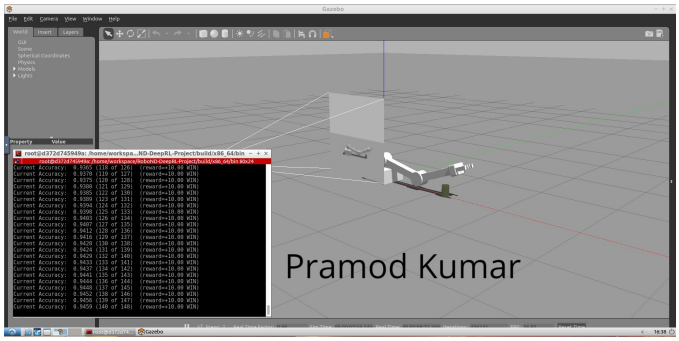

Fig. 1. Objective 1 Accuracy

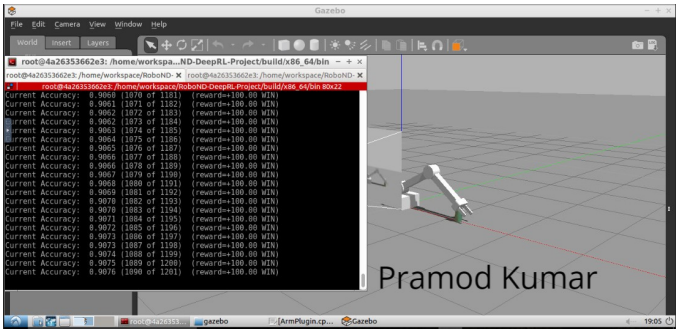

Fig. 2. Objective 2 Accuracy

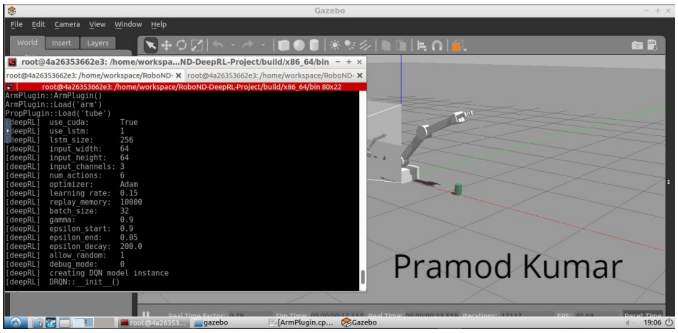

Fig. 4. Objective 2 Parameters

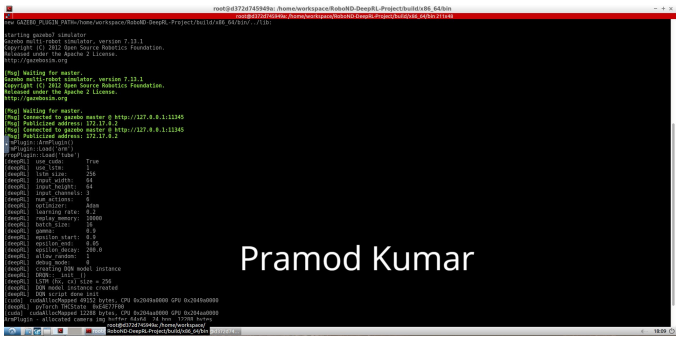

Fig. 3. Objective 1 Parameters

5 Future work

To get 100% accuracy, the other DQN hyperparameters value also tuned like discount factor and reply memory. To make it fast also required for real life and make the learner also fast will be significant improvement. It should not collide with ground because in real life scenario robot arm can break. Optimize in such a way in normal system can simulate where GPU is not available in the system.