Deep RL Arm Manipulator Write-up

1. Introduction

In this project we design a Q-value based reinforcement-learning agent to learn control of a 2 DOF arm manipulator without any computation of arm kinematics. We design a deep neural network to estimate the optimal policy based on Q-value approach. The Deep Q Network (DQN) is trained by performing a sequence of random actions at current states and computing the reward corresponding to the state action pair. We design a reward function so that it would be maximized if appropriate actions are selected for each state. The network is then, after training, capable of perform a suitable action based on the state of the environment captured by the sequence of images fed into the network. Our work is divided into two main parts, DQN design and reward function design.

2. Type of Control

To move an arm manipulator we control its joint variables which may be angles of these joints to get a certain configuration (i.e. position control strategy) or velocities of joints to control a speed profile until reach the desired configuration (i.e. velocity control strategy). In our work, we adopt the first approach i.e. joint positions control. Thus the output of the DQN represents the change in position of each joint so that the arm can reach the desired configuration.

3. Reward Function Design

We design our reward function following the simple logic introduced in the guidelines but with slight modification. The same reward logic is used for the two objectives with slight difference in parameters. Our logic in calculating reward goes as follows:

1. Punishment (i.e. Reward Loss).
    a. When the maximum number of episodes is exceeded, we set a negative value for the reward equals: -20.0 and end the episodes.
    b. When the gripper touches the ground, we set a negative value for the reward equals: -20.0 and end the episodes.
2. Winning (i.e. Reward Win).
    a. When a part of the robot touches the prob. in the case of 1st objective or when the gripper touches the prob. in the case of 2nd objective, we set a positive value for the reward equals: +20.0 and end the episodes.
3. Successive Rewards for continuing without loss.
    a. Reward is given when the arm approaches the goal (i.e. the distance to goal in this time is less than the previous: distDelta = lastGoalDistance - distGoal

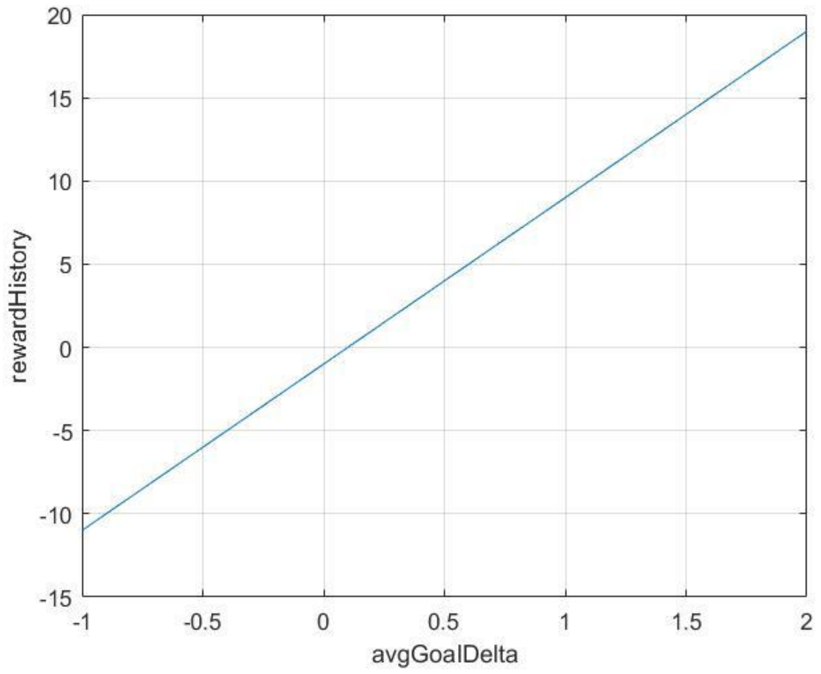
b. The Reward is calculated based on the average change in distance not the "distDelta" itself. "avgGoalDelta" is calculated as follows:

   avgGoalDelta = alpha * avgGoalDelta + (1-alpha) * distDelta

c. The reward is then calculated as a linear function of avgGoalDelta:

   rewardHistory = beta * (avgGoalDelta + bias)

d. alpha , beta and bias are tuning parameters have values equal: 0.7, 10.0 for 1st objective / 12.0 for 2nd objective, and -0.1 respectively
e. The logic behind the negative value for bias is to punish the agent if it consumes time without approaching the target.
f. The curve of the rewardHistory vs. avgGoalDelta is shown below:



4. DQN Architecture and Hyper Parameter Tuning

We use the given DQN design with LSTM enabled to capture history of the scene (i.e. modeling scene dynamics), the Network is fed with images with dimensions: 64x64x3 and is trained using "RMSprop" learning algorithm since it achieves better results. The learning rate is selected after some trials to reach better response. We achieved the desired accuracy with learning rate = 0.25 in 1st objective and learning rate = 0.5 in 2nd objective. The output of the network represents the actions to be taken. We have six actions: two for each joint of the three joints, one for increasing joint angle and the other is for decreasing it.

For other parameters of the network we use:

- Batch size = 64 images in 1st objective which is not sufficient to achieve desired accuracy for 2nd objective, the desired accuracy is achieved by making batch size = 128 images
- We use replay memory (for state action reward tuples storage) of length 10000
- When the agent wins it get REWARD_WIN = 20
- When the agent losses it gets reward: REWARD_LOSS = -20

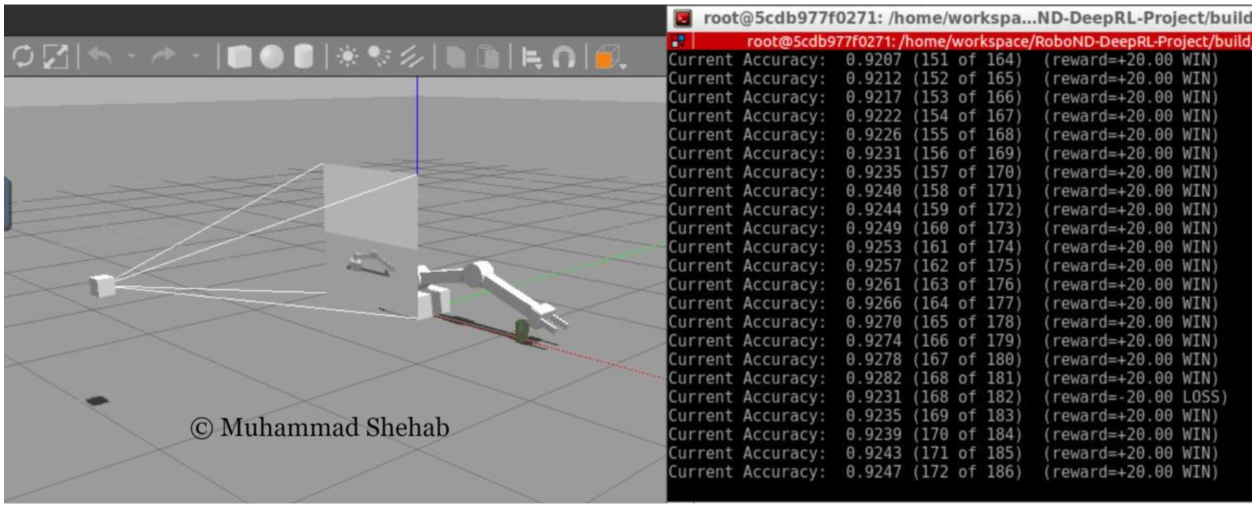
Reward win and loss are adjusted to be more enough than successive reward or punishment to push the networks towards deciding actions that make the agent wins and avoid and collision.

5. Results

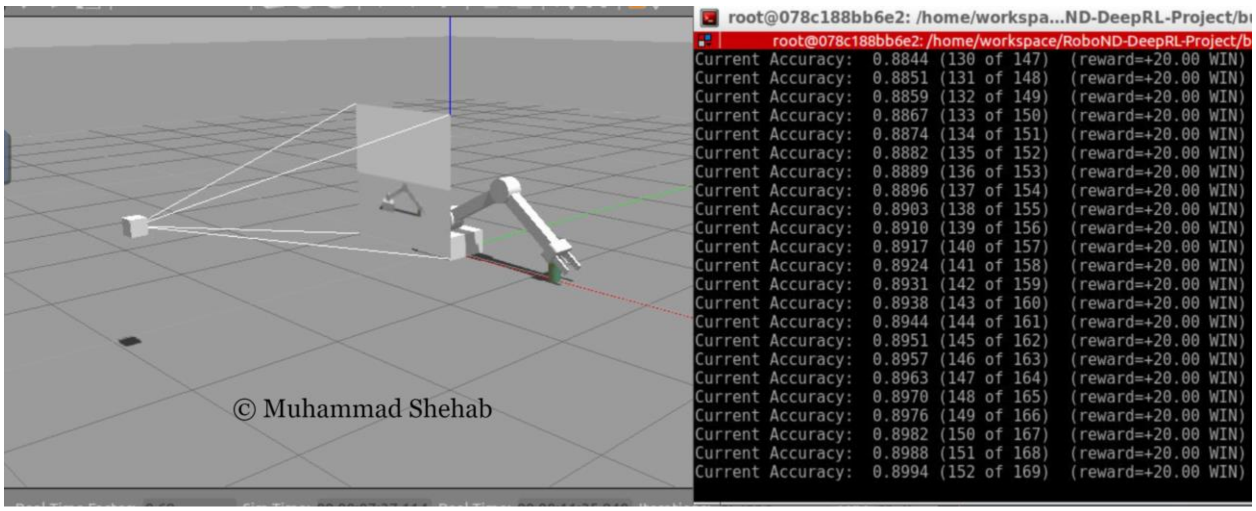
After designing our reward function and adjusting hyper parameters we achieved the desired goal. Which is:

- A part of the arm touches the prop with accuracy >= 90%
- The gripper touches the prop with accuracy >=80%

The two figures below show these results respectively



```
root@5cdb977f0271: /home/workspa...ND-DeepRL-Project/build
root@5cdb977f0271: /home/workspace/RoboND-DeepRL-Project/build
Current Accuracy:  0.9207 (151 of 164)  (reward=+20.00 WIN)
Current Accuracy:  0.9212 (152 of 165)  (reward=+20.00 WIN)
Current Accuracy:  0.9217 (153 of 166)  (reward=+20.00 WIN)
Current Accuracy:  0.9222 (154 of 167)  (reward=+20.00 WIN)
Current Accuracy:  0.9226 (155 of 168)  (reward=+20.00 WIN)
Current Accuracy:  0.9231 (156 of 169)  (reward=+20.00 WIN)
Current Accuracy:  0.9235 (157 of 170)  (reward=+20.00 WIN)
Current Accuracy:  0.9240 (158 of 171)  (reward=+20.00 WIN)
Current Accuracy:  0.9244 (159 of 172)  (reward=+20.00 WIN)
Current Accuracy:  0.9249 (160 of 173)  (reward=+20.00 WIN)
Current Accuracy:  0.9253 (161 of 174)  (reward=+20.00 WIN)
Current Accuracy:  0.9257 (162 of 175)  (reward=+20.00 WIN)
Current Accuracy:  0.9261 (163 of 176)  (reward=+20.00 WIN)
Current Accuracy:  0.9266 (164 of 177)  (reward=+20.00 WIN)
Current Accuracy:  0.9270 (165 of 178)  (reward=+20.00 WIN)
Current Accuracy:  0.9274 (166 of 179)  (reward=+20.00 WIN)
Current Accuracy:  0.9278 (167 of 180)  (reward=+20.00 WIN)
Current Accuracy:  0.9282 (168 of 181)  (reward=+20.00 WIN)
Current Accuracy:  0.9231 (168 of 182)  (reward=-20.00 LOSS)
Current Accuracy:  0.9235 (169 of 183)  (reward=+20.00 WIN)
Current Accuracy:  0.9239 (170 of 184)  (reward=+20.00 WIN)
Current Accuracy:  0.9243 (171 of 185)  (reward=+20.00 WIN)
Current Accuracy:  0.9247 (172 of 186)  (reward=+20.00 WIN)
```



```
root@078c188bb6e2: /home/workspa...ND-DeepRL-Project/b
root@078c188bb6e2: /home/workspace/RoboND-DeepRL-Project/b
Current Accuracy:  0.8844 (130 of 147)  (reward=+20.00 WIN)
Current Accuracy:  0.8851 (131 of 148)  (reward=+20.00 WIN)
Current Accuracy:  0.8859 (132 of 149)  (reward=+20.00 WIN)
Current Accuracy:  0.8867 (133 of 150)  (reward=+20.00 WIN)
Current Accuracy:  0.8874 (134 of 151)  (reward=+20.00 WIN)
Current Accuracy:  0.8882 (135 of 152)  (reward=+20.00 WIN)
Current Accuracy:  0.8889 (136 of 153)  (reward=+20.00 WIN)
Current Accuracy:  0.8896 (137 of 154)  (reward=+20.00 WIN)
Current Accuracy:  0.8903 (138 of 155)  (reward=+20.00 WIN)
Current Accuracy:  0.8910 (139 of 156)  (reward=+20.00 WIN)
Current Accuracy:  0.8917 (140 of 157)  (reward=+20.00 WIN)
Current Accuracy:  0.8924 (141 of 158)  (reward=+20.00 WIN)
Current Accuracy:  0.8931 (142 of 159)  (reward=+20.00 WIN)
Current Accuracy:  0.8938 (143 of 160)  (reward=+20.00 WIN)
Current Accuracy:  0.8944 (144 of 161)  (reward=+20.00 WIN)
Current Accuracy:  0.8951 (145 of 162)  (reward=+20.00 WIN)
Current Accuracy:  0.8957 (146 of 163)  (reward=+20.00 WIN)
Current Accuracy:  0.8963 (147 of 164)  (reward=+20.00 WIN)
Current Accuracy:  0.8970 (148 of 165)  (reward=+20.00 WIN)
Current Accuracy:  0.8976 (149 of 166)  (reward=+20.00 WIN)
Current Accuracy:  0.8982 (150 of 167)  (reward=+20.00 WIN)
Current Accuracy:  0.8988 (151 of 168)  (reward=+20.00 WIN)
Current Accuracy:  0.8994 (152 of 169)  (reward=+20.00 WIN)
```

6. Future Work

This project is supposed to have further modifications for performance enhancement and more reachability in the environment. These modifications can be summarized below:

- Unlock the base so that the arm have 3 DOF and can move in 3D not a planner only
- Modify the reward function to include successive punishment for taking long time to reach the goal.
- Make the reward function a nonlinear function of the avgGoalDelta. Since the pose of end effector is related to joint variables by nonlinear functions, it is thought that this would give better performance.
- Make a punishment on the case where the agent take an action that moving the gripper far away the prop to guarantee convergence of end effector pose to the desired pose
- The hyper parameters are supposed to receive more attention for fine tuning