Deep Reinforcement Learning Homework

Jacob Blevins

February 28, 2025

1 Part 0: Warm up questions

1. How does the Q function $Q(s_t, a_t)$ relate to sum of discounted rewards $\sum_{t=0}^{T} \gamma^t r_t$?

   Ans: The Q function is defined as the expected sum of discounted rewards, given that we are starting at some state and with some initial action, meaning that we average the discounted rewards over many trajectory rollouts:

   $Q(s_t, a_t) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t | s_t, a_t]$

2. How does the Q function $Q(s_t, a_t)$ relate to $Q(s_{t+1}, a_{t+1})$?

   Ans: We can separate the immediate reward and the future reward as follows:

   $Q(s_t, a_t) = \mathbb{E}[r_t + \gamma Q(s_{t+1}, a_{t+1})]$

3. When Q is accurate are these definitions equivalent?

   Ans: Yes. The Q value is defined as the sum of future discounted rewards so the first/immediate reward can be pulled out and the remaining rewards are γ times the Q value at the next step as shown in 2).

4. Whats the loss for a neural approximation to the Q network?

   Ans: The loss for a Q network (value-based RL algorithm) is the squared difference of the target/true Q value and the estimated Q value. This is often represented as MSE loss for batch sizes great than 1. We see this is subtracting the definition in 1) from that in 2). This difference is defined as the temporal difference error.

   $L(\theta) = (\text{Predicted Q value} - \text{Target Q value})^2 = (Q(s, a; \theta) - (r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^{target})))^2$

5. In the discrete case, how do you select actions given an accurate Q network?

   Ans: Actions for discrete Q networks are chosen by taking the argmax of the Q network outputs which represents choosing the action with the maximum expected reward.

   $a_t = argmax(Q(s_t, a))$

6. In DQN, for an environment with 5 continuous states and 3 discrete action choices: Whats the input and output size of the Q network?

   Ans: In the discrete action space, the Q network determines the value of each possible action so the input and output sizes of the network are 5 and 3, respectively.

7. In DDPG, for an environment with 5 continuous states and 3 continuous action: Whats the input and output size of the Q network?

   Ans: In the continuous case, a Q network determines the value of taking an action in a state so the output is size 1, the value. The input to the network is the state and action, meaning the input is size $5 + 3 = 8$.

8. In DQN, what is a target network and why do we need it?

   Ans: A target network is a slowly updating network which updates lesser than the primary Q network, allowing for more stable training. Instabilities can come from fast big updates.

   Hard update: $\theta_{target} \leftarrow \theta$

   Target network soft update with small τ value: $\theta_{target} \leftarrow \tau\theta + (1 + \tau)\theta_{target}$

9. In DQN, what is a replay buffer and why do we need it?

   Ans: A replay buffer is storage of past experiences which the model can sample from for future training. We need it to stabilize training by decorrelating the current training by using those past experiences during policy updates.

10. Explain this inequality $\mathbb{E}[\max(C_1, C_2)] \geq \max(\mathbb{E}[C_1], \mathbb{E}[C_2])$, assuming C_1 and C_2 are random variables representing the probability of getting heads when flipping two fair coins. (This is the basis for double Q learning in Double DQN and SAC.)

    Ans: This inequality is saying that in standard Q learning, overestimation happens in the loss because the max is selected before the expected value is taken. On the other hand (right side), now the expected value is taken and then the max which reduces the loss function output.

11. Do off policy algorithms use a replay buffer?

    Ans: Off policy algorithms do use a replay buffer. This is because the policy is updated after rollouts using rollout data and the buffer. On policy algorithms just use the recent rollouts to update.

12. What does 'with torch.nograd()' do and why should you use it when calling target networks but not regular networks?

    Ans: This python command causes the called target network to not update. We do not want to update the target network during backpropagation.

13. Why do you need a policy network in DDPG but not DQN, and what is the DDPG policy loss?

    Ans: DDPG is a policy based model so a policy is updated/created to directly give action outputs. DQN is a value-based model so no explicit policy is formed, but rather values are estimated from the network

    $\mathcal{L}_{DDPG}(\theta_p) = -\mathbb{E}[Q_\theta(s, a)]$

14. Compare and contrast hard and soft target network updates.

    Ans: Hard target network updates are direct updates of the target network. Soft updates are utilization of small updates with a target network which blends the old and new networks for more stable training

15. In [InvertedPendulum-v5] what are the physical meanings of states and actions and are they discrete or continuous?

    Ans: In this specific environment, the states are the translational position, translational velocity, pole angle, and pole angular velocity. These states are continuous. The one action is a continuous force applied colinear with the sliding rail.

2 Part 1: DQN

1. DQN uses target networks to stabilize training. DQN Loss:

$$\mathcal{L}(\theta) = \mathbb{E}[\{Q_\theta(s, a) - (r_t + \gamma \max_{a'} Q_{\theta_{\text{target}}}(s', a') \cdot \text{not\_done})\}^2]$$

But, what if you didn't use a target network:

$$\mathcal{L}(\theta) = \mathbb{E}[\{Q_\theta(s, a) - (r_t + \gamma \max_{a'} Q_\theta(s', a') \cdot \text{not\_done})\}^2]$$

(Optionally, make this actual change in code and observe training results for yourself. Its a one line change. Make sure to revert or comment it before submitting. Any code change is not for credit, but may aid understanding. You can change the name of the tensorboard run for direct visual comparison.)

In 1 or 2 sentences, what could happen to your Q loss over the course of training if you modified the DQN loss equation so that a target network is not used? Why?

Ans: The loss over the epochs of training without a target network would induce instability in training whereas the slowed target network training keeps the training from jumping too fast and falling into instability.

2. The loss function for Double DQN improves upon standard DQN by using the Q-network to select the best action and the target Q-network to evaluate it:

$$\mathcal{L}_{\text{double dqn}}(\theta) = \mathbb{E}\{(Q_\theta(s,a) - [r_t + \gamma Q_{\theta_{\text{target}}}(s', \arg\max_{a'} Q_\theta(s',a')) \cdot \text{not\_done}])^2\}$$

(Optionally, implement this in code and observe the training, but comment or revert changes before submitting.)

In 1 or 2 sentences, explain the intuition behind why this might improve performance.

Ans: The double DQN uses the target network to determine the action and an online network to determine the value. This method decouples the action and value determinations which reduces over-estimation bias.

3 Part 2: DDPG

1. In HW1 we used a single combined optimizer for both value and policy nets. For DDPG, we need separate optimizers for Q and Policy nets. Why is that? (hint: policy loss)

   Ans: In PPO, we used a single loss function for both actor and critic networks, but in DDPG, we have two separate loss functions and thus need to update each individually, especially because the critic is minimizing the error and the policy is maximizing the reward.

2. Every policy rollout (defined in boiler plate code) uses 32 parallel environments simulated for 128 timesteps.

```
class DRL:
    def __init__(self):
        self.n_envs = 32
        self.n_steps = 128
```

   Additionally, our replay buffer (defined in boiler plate code) is large enough to hold 1,000,000 transitions.

```
class ReplayBuffer:
    def __init__(self):
        self.buffer = deque(maxlen=1_000_000)
        self.batch_size = 32
```

   In 1 or 2 sentences, what might happen to our training speed and stability if we collected less data per rollout and used a smaller replay buffer? Why? Lets say 1 environment, 32 steps, size 32 replay buffer. (Optionally, make these changes in code and observe training results yourselves. Note, if you test lower than 32 transitions you need to reduce ReplayBuffer.batch_size aswell. Comment or revert changes before submitting.)

   Ans: The reduced rollout steps would speed up training but remove ability to explore, and the reduced buffer size would lose valuable training knowledge very quickly, reducing the learning stability.