Name: Nayan Panda

Reg. No: 20BAI10188

Reinforcement Learning

Assignment

Q1. What is actor-critic process.

Ans. The actor-critic process is a popular approach in reinforcement learning (RL) that combines elements of both policy-based methods (actor) and value-based methods (critic) to improve learning efficiency and stability. It is a way to learn both a policy and a value function simultaneously.

In the actor-critic process, there are two main components:

1. Actor: The actor is responsible for learning and updating the policy. It takes the current state as input and outputs an action based on its learned policy. The policy can be deterministic (directly maps states to actions) or stochastic (outputs a probability distribution over actions). The actor explores the environment, collects experiences, and updates its policy to maximize the expected cumulative reward. This is typically done using techniques like policy gradient methods, where the gradients of the policy parameters are used to update the policy in the direction of higher rewards.

2. Critic: The critic is responsible for estimating the value function or the expected cumulative reward. It takes the current state as input and outputs an estimate of the expected future reward from that state. The critic provides feedback to the actor by evaluating the quality of the actor's actions and policy. It helps in learning the value of different states or state-action pairs. The critic can use various value-based methods like temporal difference learning or Q-learning to update its value estimates based on the observed rewards and state transitions.

The actor-critic process combines the advantages of policy-based methods (direct policy optimization, better handling of continuous action spaces) with value-based methods (efficient estimation of the value function, better handling of delayed rewards). The actor learns to make better decisions by receiving feedback from the critic, and the critic learns from the actor's interactions with the environment.

Overall, the actor-critic process provides a way to jointly learn the policy and value function in reinforcement learning, leveraging the strengths of both policy-based and value-based approaches.

Q2. Derive the formulae for bellman expectation equation.

Ans. The Bellman expectation equation is a fundamental equation in reinforcement learning that expresses the relationship between the value function and the expected value of the next state.

Essentially, the Bellman Equation breaks down our value functions into two parts

- Immediate reward
- Discounted future value function

State-value function can be broken into:

$$\begin{aligned}\mathcal{V}_\pi(s) &= \mathbb{E}[\mathcal{G}_t|\mathcal{S}_t = s] \\ &= \mathbb{E}[\mathcal{R}_{t+1} + \gamma\mathcal{R}_{t+2} + \gamma^2\mathcal{R}_{t+3} + \ldots|\mathcal{S}_t = s] \\ &= \mathbb{E}[\mathcal{R}_{t+1} + \gamma(\mathcal{R}_{t+2} + \gamma\mathcal{R}_{t+3} + \ldots)|\mathcal{S}_t = s] \\ &= \mathbb{E}[\mathcal{R}_{t+1} + \gamma\mathcal{G}_{t+1}|\mathcal{S}_t = s] \\ &= \mathbb{E}[\mathcal{R}_{t+1} + \gamma\mathcal{V}_\pi(s_{t+1})|\mathcal{S}_t = s]\end{aligned}$$

Action-value function can be broken into:

$$\mathcal{Q}_\pi(s,a) = \mathbb{E}[\mathcal{R}_{t+1} + \gamma\mathcal{Q}_\pi(s_{t+1}, a_{t+1})|\mathcal{S}_t = s, \mathcal{A} = a]$$

Bellman Expectation Equations

Basic: State-value function $V_\pi(s)$

- Current state S
- Multiple possible actions determined by stochastic policy $\pi(a|s)$
- Each possible action is associated with an action-value function $Q\pi(s,a)$
- returning a value of that particular action
- Multiplying the possible actions with the action-value function and summing them gives us an indication of how good it is to be in that state
    - Final equation: $V_\pi(s)=\sum_{a\in A}\pi(a|s)Q(s,a)$
    - Loose intuitive interpretation:
        - state-value = sum (policy determining actions * respective action-values)

Basic: Action-value function $Q_\pi(s,a)$

With a list of possible multiple actions, there is a list of possible subsequent states s' associated with:

- state value function $V\pi(s')$
- transition probability function Pa/ss' determining where the agent could land in based on the action
- reward Ras for taking the action

Summing the reward and the transition probability function associated with the state-value function gives us an indication of how good it is to take the actions given our state

- Final equation: $Q_\pi(s,a)=R_{as}+\gamma\sum_{s'\in S}P_{ass'}V_\pi(s')$
- Loose intuitive interpretation: action-value = reward + sum (transition outcomes determining states * respective state-values)

Q3. Deep Deterministic Policy Gradient (DDPG) in RL.

Ans. Deep Deterministic Policy Gradient (DDPG) is an algorithm in reinforcement learning (RL) that combines the concepts of deep learning and policy gradient methods to handle continuous action spaces in continuous control problems. It is an extension of the deterministic policy gradient algorithm for continuous action spaces.

In DDPG, both the policy (actor) and the value function (critic) are approximated using deep neural networks. The algorithm utilizes a combination of actor-critic methods and experience replay to improve learning stability and sample efficiency.

Here's a high-level overview of how DDPG works:

1. Actor-Critic Architecture:
   - Actor: The actor network takes the state as input and outputs a deterministic action. It learns to approximate the optimal policy by directly mapping states to actions.
   - Critic: The critic network takes the state-action pair as input and estimates the expected cumulative reward (Q-value). It learns to evaluate the quality of actions based on the current policy.

2. Experience Replay:
   - DDPG employs experience replay, where experiences (state, action, reward, next state) are stored in a replay buffer.
   - During training, a batch of experiences is randomly sampled from the replay buffer to decorrelate the data and stabilize learning.

3. Target Networks:
   - To address the problem of non-stationary targets, DDPG utilizes target networks that are updated slowly and provide target values for training.
   - The target actor network is a copy of the actor network with delayed updates.
   - The target critic network is a copy of the critic network with delayed updates.

4. Training Process:
   - The critic network is trained using a temporal difference (TD) learning algorithm, such as the Q-learning update rule, to minimize the difference between the predicted Q-value and the target Q-value.
   - The actor network is updated by performing gradient ascent on the expected cumulative reward estimated by the critic network.
   - The target networks are updated periodically by slowly blending the weights of the target networks with the weights of the online networks.

DDPG is particularly effective in continuous control tasks, such as robotic control or autonomous vehicle navigation, where the action space is continuous and requires precise control. It leverages the advantages of deep neural networks to handle high-dimensional state spaces and provides a deterministic policy, making it suitable for tasks that require precise actions.

Overall, DDPG combines the actor-critic framework, deep neural networks, experience replay, and target networks to learn policies for continuous control problems in RL.

Q4. What is a policy gradient method?

Ans. A policy gradient method is a class of reinforcement learning (RL) algorithms that directly optimize the policy, which is a mapping from states to actions, to learn optimal decision-making strategies. Unlike value-based methods that estimate the value function or action-value function, policy gradient methods focus on learning the policy directly.

In RL, the policy represents the behaviour of an agent in an environment. It defines the action the agent should take given a particular state to maximize the expected cumulative reward over time. Policy gradient methods aim to find the parameters of the policy that maximize the expected cumulative reward.

Here's a high-level overview of how policy gradient methods work:

1. Policy Representation:

- The policy is typically represented by a parametric function, such as a neural network, that takes a state as input and outputs a probability distribution over actions. This allows for both deterministic and stochastic policies.

2. Objective Function:

- The objective function in policy gradient methods is often defined as the expected cumulative reward, also known as the return or the performance measure. The goal is to find the policy parameters that maximize this objective.

3. Gradient Ascent:

- Policy gradient methods use gradient ascent to update the policy parameters in the direction of higher expected cumulative reward. The gradient of the objective function with respect to the policy parameters is computed and used to update the policy.

4. Policy Update:

- The policy is updated iteratively by taking steps in the direction of the gradient. Different optimization techniques like stochastic gradient ascent or natural gradient methods can be employed to update the policy parameters.

5. Exploration and Sampling:

- Policy gradient methods often incorporate exploration to balance between exploiting the learned policy and exploring new actions to discover better strategies. This can be achieved by adding noise to the policy during action selection or by using exploration policies like epsilon-greedy or Boltzmann exploration.

Policy gradient methods have several advantages, including their ability to handle continuous action spaces and to learn stochastic policies. They have been successfully applied in various domains, such as robotics, game playing, and natural language processing.

Overall, policy gradient methods provide a way to directly optimize the policy to learn optimal decision-making strategies in reinforcement learning. They offer flexibility in handling different types of policies and are effective in settings where the action space is continuous or when explicit knowledge of value functions is not available.