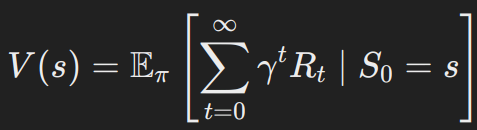
In Value Function Approximation when we should use state-value function and when state-action function? Explain clearly with one agent example. B) explain why we have Multiple forward pass problem in Deep neural network part of Deep Q-learning algorithm and how we can solve it?

A): State-Value Function vs. State-Action Function in Value Function Approximation

State-Value Function (V(s)):

Definition: Represents the expected return starting from the state s, following a specific policy π:

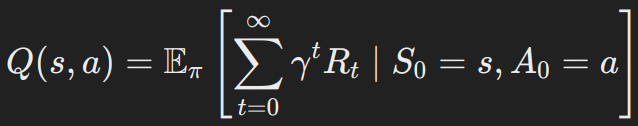


Use Case: Use V(s) when the goal is to evaluate the quality of states without explicitly considering actions. Typically used in:

Policy-based methods: Where actions are determined directly by a policy π(a∣s), and only the state value matters for policy improvement (e.g., Policy Gradient, Actor-Critic methods).

State-Action Function (Q(s, a)):

Definition: Represents the expected return starting from the state s, taking action a, and following a specific policy π thereafter:



Use Case: Use Q(s,a) when it is necessary to explicitly evaluate actions to derive an optimal policy. Typically used in:

Action-value-based methods: Such as Q-Learning and Deep Q-Networks (DQN), where the policy is derived by choosing actions that maximize Q(s,a).

Agent Example:

Consider an agent navigating a maze to reach a goal.

Using V(s): If the agent has a pre-defined policy (e.g., it knows how to act based on state (s), we evaluate V(s) to estimate how promising a state is in leading to the goal. For example, states closer to the goal will have higher values.

Using Q(s,a): If the agent needs to explore actions and optimize its policy, Q(s,a) is crucial. For instance, at an intersection, Q(s,a) can evaluate whether going left or right has a higher expected return, enabling the agent to choose optimally.

B): Multiple Forward Pass Problem in DQN

Why Does the Problem Occur?

In Deep Q-Learning, the Multiple Forward Pass Problem arises because the agent needs to evaluate the Q-values for all possible actions a at a given state s:

The Q-network outputs Q(s,a) for each action a, which requires multiple forward passes through the neural network when the action space is large or continuous.

During training, this becomes computationally expensive since evaluating Q(s,a) for all actions at each update step increases complexity.

Solution:

Single Forward Pass with All Actions (DQN Implementation):

1. Instead of calculating Q(s,a) separately for each action a, modern DQN implementations structure the output of the neural network to return Q-values for all actions simultaneously:



This allows the agent to select the best action (argmax a Q(s,a)) or compute the loss for a specific action using just one forward pass.

1. Actor-Critic Methods: These methods separate the policy (actor) and value function (critic). The actor directly outputs the best action, bypassing the need to evaluate multiple Q-values.
2. Continuous Action Space Solutions: For continuous actions, techniques like Deep Deterministic Policy Gradient (DDPG) or Twin Delayed DDPG (TD3) avoid computing Q(s,a) for multiple actions by using an actor-network to propose the best action and a critic network to evaluate it.

A) In both DQL and Actor-critic we saw how to define two deep networks. What are the differences of the target and predicted networks in DQL with actor and critic networks in Actor-critic approaches? Explain the 2 main differences.

B) Consider a 2 DoFs robotics arm and explain what is the exactly output of our DRL algorithm for continues action space (output of actor network)? Shortly, you need to explain what is the output by stating numerical range and it’s meaning first, then how agent executes that output clearly. You can show example with values or draw in word file if needed.

A): Differences between Target and Predicted Networks in DQL and Actor-Critic Networks

1. Purpose:

DQL (Deep Q-Learning):

Target Network: Provides stable Q-value targets during training by keeping its weights fixed for several iterations before being updated to the weights of the predicted (main) Q-network.

Predicted Network: Computes the Q-values used to choose actions and minimize the temporal difference (TD) loss. It is updated frequently during training.

Actor-Critic:

Actor Network: Outputs the policy (actions to take) based on the state. It directly maps states to actions, optimized to maximize expected returns.

Critic Network: Estimates the value function Q(s,a) or V(s), which helps the actor improve its policy by providing feedback on how good an action is.

1. Type of Output:

DQL:

Both the target and predicted networks output Q(s,a) values for all possible actions in a discrete action space.

Actor-Critic:

The actor network outputs actions (specific values for continuous action spaces).

The critic network outputs a single value: either V(s) or Q(s,a).

B): Output of a DRL Algorithm for Continuous Action Space (2 DoFs Robotic Arm)

Output of the Actor Network:

Numerical Range:

For a 2 DoFs robotic arm, the actor network outputs continuous action values for each degree of freedom (DoF). These values typically lie within a normalized range, such as [-1, 1].

Each output corresponds to the control signal for one joint (e.g., angular velocity or torque).

Meaning:

The output values represent the agent's decision on how to adjust each joint of the robotic arm:

A value of -1 might represent the maximum negative rotation or force for a joint.

A value of 1 might represent the maximum positive rotation or force for a joint.

A value of 0 represents no action (neutral position).

Example:

1. Actor Network Output:

Suppose the actor network outputs: [0.5, -0.8].

For joint 1 (DoF 1): 0.5 means applying moderate positive force/rotation.

For joint 2 (DoF 2): −0.8 means applying strong negative force/rotation.

1. Execution by the Agent:

The robot's control system translates these normalized outputs into actual motor commands. For instance:

If joint 1 can rotate between −90 and +90 , an output of 0.5 might translate to +45 .

If joint 2 applies torques in the range [−10Nm,+10Nm], an output of −0.8 might translate to −8Nm.

1. Real Execution:

The robot sends these computed motor commands to its actuators, which adjust the joints accordingly.

Example after conversion: Joint 1: Rotate to +45 , Joint 2: Apply −8Nm torque.