Giorgio Mendoza

RBE595-S24-S04

Dr. Dadkhah Terani

Reinforcement Learning

DRL Assignment #7

1- What are the two sources of error in Deep RL with function approximation?

In RL with function approximation, there are two primary sources of error:

Bias: This arises from the algorithm's inability to represent the true value function or policy accurately. In deep RL, this can occur due to several reasons, such as the choice of neural network architecture, approximation errors introduced during training, or limitations in the representational capacity of the neural network compared to the true value or policy function.

Variance: This refers to the variability in the estimates of the value function or policy that occurs due to the stochasticity inherent in the environment, the exploration strategy, or the learning algorithm itself. In deep RL, variance can be worsened by factors such as noisy gradients during training, non-stationarity of the data distribution, or the inherent randomness in the environment's dynamics.

2- In TD (Temporal Difference) learning with a neural network, we are typically trying to minimize the temporal difference error or TD error.

The TD error represents the discrepancy between the predicted value of a state or action and the actual observed value, which is updated based on the observed rewards and the estimated values of subsequent states. The TD error is often defined as:

$$\text{TD error} = R + \gamma \cdot V(S') - V(S)$$

Where:

- R is the immediate reward received after taking an action.
- γ is the discount factor, which determines the importance of future rewards.

- V(S) is the estimated value of the current state S.
- V(S') is the estimated value of the next state S' after taking an action.

By minimizing the TD error, the goal is to reduce the discrepancy between the predictions and the observed rewards, leading to more accurate value function estimates and better decision-making.

In RL, the objective is to maximize the expected cumulative reward over time. In TD learning with a neural network, the focus shifts to indirectly maximizing the expected cumulative reward by refining the value function approximation. This enhancement enables more informed decision-making, resulting in higher expected returns.

3- What is the main benefit of deep neural networks in function approximation compared to linear method? What is the linear function approximation useful for?

DNNs excel in capturing complex, nonlinear relationships in data due to:

Nonlinearity: DNNs learn intricate, nonlinear mappings, making them suitable at approximating complex functions, crucial in scenarios like RL where environments exhibit nonlinear dynamics.

Hierarchical Representation: DNNs' multiple layers extract abstract features, enabling them to model intricate patterns, particularly advantageous for tasks with hierarchical structures or multiple levels of abstraction.

Automatic Feature Learning: DNNs automatically learn feature representations from raw data, eliminating the need for manual feature engineering and often leading to superior performance on tasks with high-dimensional or unstructured data.

Linear function approximation is valuable for:

Simplicity and Interpretability: Linear models are simpler and more interpretable, making them suitable for scenarios where transparency and understanding are essential.

Low-Dimensional Data: Linear models provide sufficient accuracy for tasks with low-dimensional data and linear relationships, without the computational complexity of DNNs.

Computational Efficiency: Linear models are computationally efficient to train and deploy, especially compared to DNNs, which may require significant computational resources for training and inference.

So, the main benefit of DNNs compared to linear methods is their ability to capture complex and nonlinear relationships within data. DNNs achieve this through their nonlinear activation functions and hierarchical representation, enabling them to automatically learn intricate patterns from raw data.

Linear function approximation is useful for tasks where relationships between variables are simple and linear. It's preferred for its simplicity, interpretability, and computational efficiency, especially when dealing with low-dimensional data or when transparency is essential.

4- In DQN, what is the purpose of the target network and value network?

In Deep Q-Network, the target network and value network serve different purposes but work together to stabilize and improve the training process. Here's the breakdown:

Value Network (Q-Network):

- The value network, often referred to as the Q-network, is the primary neural network responsible for approximating the Q-values (action values) for each state-action pair.
- It takes the state as input and outputs the Q-values for all possible actions in that state.
- The value network is trained to minimize the temporal difference (TD) error between its predicted Q-values and the target Q-values.

Target Network:

- The target network is a separate neural network used to stabilize the training of the value network.
- It has the same architecture as the value network but with a delayed update schedule.
- The parameters of the target network are updated less frequently compared to the value network.

- The purpose of the target network is to provide stable target Q-values during training, reducing the potential for divergence or oscillations in the learning process.
- Target networks are especially crucial in DQN to address the problem of overestimation bias, where the value estimates may become overly optimistic due to the maximization step in the Bellman equation.

In summary, the value network (Q-network) learns to approximate Q-values, guiding the agent's decision-making process, while the target network provides stable target Q-values for training, contributing to the stability and convergence of the learning algorithm, particularly in the presence of non-stationarity or overestimation bias.

5- In the Deep Q-learning method, which are true:

a. epsilon-greedy is required to ensure exploration.

b. exploration this taken care of by the randomization provided by experience replay?

a. True: Epsilon-greedy exploration is typically employed to ensure exploration. Epsilon-greedy is a strategy where the agent selects the action with the highest estimated Q-value most of the time (exploitation), but with probability ε, it chooses a random action (exploration). This balance between exploitation and exploration is crucial for the agent to discover new states and actions that may lead to higher rewards.

b. True: Exploration is also facilitated by experience replay. Experience replay involves storing experiences (state, action, reward, next state) in a replay buffer during interactions with the environment. During training, batches of experiences are randomly sampled from the replay buffer, providing a diverse set of transitions for the agent to learn from. This randomization helps the agent explore different state-action pairs and prevents it from getting stuck in local optima.

So, both statements are true. Epsilon-greedy ensures exploration during action selection, while experience replay enhances exploration by providing randomization through the diverse set of experiences sampled during training.

6- Value function-based RL methods are oriented towards finding deterministic policies whereas policy search methods are geared towards finding stochastic policies

a. True

b. False

Value function-based RL methods, such as Q-learning and Deep Q-learning, are not inherently oriented toward finding deterministic policies. These methods focus on learning value functions (e.g., state-action values in Q-learning) that represent the expected return of following a particular policy. However, they do not directly specify the policy itself. Policies can be derived from value functions using techniques like greedy policy improvement, where the action with the highest value is selected in each state, but this doesn't necessarily result in a deterministic policy.

Policy search methods, on the other hand, explicitly target to find stochastic policies, where the agent's actions are probabilistically determined based on the current state. These methods optimize the parameters of a policy directly to maximize expected return, often using techniques such as gradient ascent or evolutionary algorithms.

So, the statement is False. Value function-based RL methods are not exclusively oriented toward finding deterministic policies, and policy search methods are not solely geared toward finding stochastic policies.

7- What makes the optimization space smooth in policy gradient methods?

In policy gradient methods, the optimization space is smooth due to a differentiable policy parameterization. Meaning the policy function, mapping states to actions, is defined by

continuous parameters. These parameters can be updated using gradient-based optimization, ensuring smooth transitions in the optimization process.

Key factors contributing to this smoothness include:

Differentiable Policy Parameterization: The policy function, often represented by a neural network, allows for smooth changes in output with parameter adjustments.

Smooth Objective Function: The objective function, typically the expected return, is smooth with respect to policy parameters. This smoothness ensures small parameter changes lead to small objective function changes.

Continuous Action Spaces: In environments with continuous action spaces, policy gradients naturally produce smooth changes in action probabilities as parameters adjust.

This combination enables efficient optimization of policy parameters, facilitating effective exploration and learning in policy gradient methods.

8- How does actor-critic architecture improve the “vanila” policy gradient?

The actor-critic architecture enhances the "vanilla" policy gradient method in multiple ways, for example:

Value Function Estimation:

The actor-critic setup involves two components: the actor (policy) and the critic (value function). The critic estimates the value function, reflecting the expected return from a state under the current policy. This estimation provides feedback to the actor, improving policy updates.

Reduced Variance:

Critic's value function estimation lowers the variance of policy gradient estimates.

By subtracting learned value estimates from observed returns, policy updates become less noisy and more stable, fostering quicker and more reliable learning.

Temporal Difference Learning:

The critic often employs temporal difference (TD) learning methods like TD(0) or TD(λ) to update value function estimates.

TD learning allows the critic to learn from incomplete experience sequences, enhancing data utilization and sample efficiency compared to vanilla policy gradient methods.

Bootstrapping:

Critic's value estimates are often bootstrapped, updated based on other value estimates rather than solely relying on observed returns.

This enables more frequent critic updates, leading to faster adaptation and learning in dynamic environments.

Asynchronous Updates:

Actor and critic networks can be updated asynchronously in some setups, maximizing computational resources and hastening convergence.

In summary, by combining policy learning with value function learning, the actor-critic architecture overcomes some limitations of vanilla policy gradient methods, resulting in more stable, efficient, and effective RL algorithms.

9- What is the Advantage function, $A\pi(st, at)$, in actor-critic architecture? Write down the equation for monte-carlo-based Advantage function and TD-based advantage function and comment on the pro and cons of each one.

1. The Monte Carlo-based Advantage function is given by:

- $A_{MC}^{\pi}(S_{t,}\, a_t) \;=\; G_t - V^{\pi}(S_t)$
- G_t is the sampled return (total discounted reward) following the execution of action a_t from state S_t.
- $V^{\pi}(S_t)$ is the value function estimate of state S_t under policy π.

The advantage function is calculated as the difference between the sampled return and the value function estimate.

2. TD-based Advantage Function:

- $A_{MC}^{\pi}(S_{t,}\, a_t) = r_{t+1} + \gamma V^{\pi}(S_t + 1) - V^{\pi}(S_t)$
- r_{t+1} is the reward received after taking action a_t in state S_t.
- γ is the discount factor.
- $V^{\pi}(S_{t+1})$ is the value function estimate of the next state S_{t+1} under policy π.

Some pros and cons of the Monte Carlo-based Advantage Function:

- Pros: Provides unbiased estimates and captures long-term effects.
- Cons: High variance, especially in stochastic environments or long episodes. Requires waiting until the episode ends to compute returns, potentially slowing down learning.

Some pros and cons of the TD-based Advantage Function:

- Pros: Lower variance, allowing for more frequent updates. Can be computed online after each timestep.
- Cons: Prone to bias if value function estimates are inaccurate. May struggle in environments with sparse rewards or long time horizons.

10- Can you find a resemblance between actor-critic architecture and generalized policy iteration in chapter 4?

The actor-critic model in RL mirrors the Generalized Policy Iteration principle, featuring two main components: the actor (policy) and the critic (value function). Similar to GPI's policy evaluation and improvement, the critic assesses actions, and the actor updates the policy accordingly. Both aim for an optimal balance of action selection and policy evaluation, using iterative updates and bootstrapping methods. As such, actor-critic can be viewed as an application of

GPI, converging towards optimal policy and value function through continual interaction and refinement of the actor and critic components.

11- In the actor-critic architecture described in the lecture, assuming the use of differentiable function approximators, we can conclude that the use of such a scheme will result in:

a) convergence to a globally optimal policy

b) convergence to a locally optimal policy

c) cannot comment on the convergence of such an algorithm.

When using function approximators, convergence to a globally optimal policy cannot be guaranteed. This is attributed to the non-convex nature of the problem space when utilizing neural networks or similar approximators for policy and value function estimation. Actor-critic methods with differentiable function approximators can explore the policy space and often discover effective policies, yet they are susceptible to becoming trapped in local optima. This limitation stems from the inherent characteristics of gradient-based optimization methods utilized in neural networks, commonly found in actor-critic architectures. While these methods follow gradients towards reward maximization, they may overlook superior solutions beyond local gradient information.

12- Assume that we are using the linear function approximation for the critic network and SoftMax policy with linear function approximation for the actor network. Write down the parameter update equations for the actor and the critic network (use on-line update equations similar to the algorithm in page 332)

The parameter update equations for both the actor and the critic network using linear function approximation for the critic and a SoftMax policy with linear function approximation for the actor are as follows:

For the Critic (with weights w):

$w \leftarrow w + \alpha^{w} \delta \phi(S)$

where δ is the temporal difference (TD) error:

$\delta = R + \gamma w^{T} \phi(S') - w^{T} \phi(S)$

$\phi(S)$ is the feature vector for state S, γ is the discount factor, and α^{w} is the step size for the critic.

For the Actor (with parameter θ for the policy):

$\theta \leftarrow \theta + \alpha^{\theta} \delta \nabla_{\theta} ln\pi(A \mid S, \theta)$

Where $\nabla_{\theta} ln\pi(A \mid S, \theta)$ is the gradient of the log probability of the action *A* taken in state *S* with respect to the policy parameter *θ* and α^{θ} is the step size for the actor.

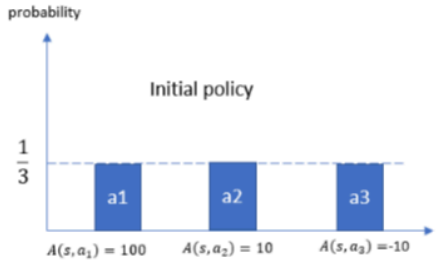
The gradient of the log policy for a SoftMax policy with linear function approximation is:

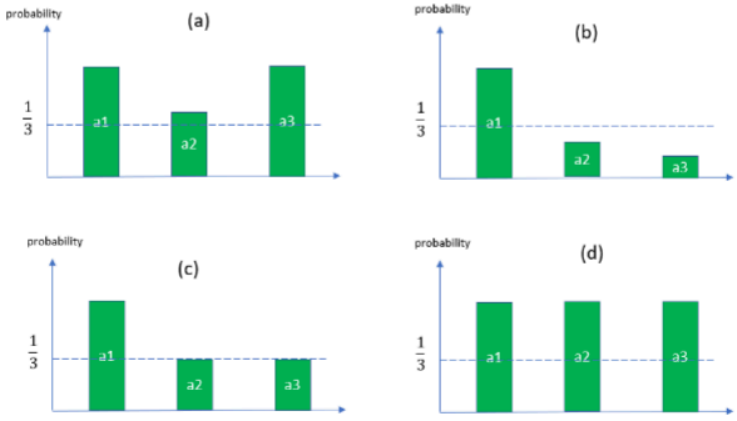
$\nabla_{\theta} ln\pi(A \mid S, \theta) = \phi(S, A) - \sum_{b} \pi(b \mid S, \theta) \phi(S, b)$

Here $\phi(S, A)$ is the feature vector for the state-action pair (S, A) and $\pi(b \mid S, \theta)$ is the probability of taking action *b* in state *S* under the SoftMax policy.

In both equations, δ is the signal that drives the learning, informing both the actor and the critic of the quality of the action taken. The feature vectors ϕ(*S*), ϕ(*S, A*) and ϕ(*S, b*) must be defined as part of the function approximation architecture. These equations will adjust the parameters *w* and θ incrementally, following the trajectory of experience the agent goes through.

13- Consider a trajectory rollout of (s**,** a**1,** s**,** a**2,** s**,** a**3). The initial policy depicted in the figure below for state** s**.The horizontal line also shows the value of the Advantage function for each (state,action) pair. After applying the policy gradient update, what do you expect the probability of each action to change to?**





1- For action a_1: since $A(s,\ a_1) = 100$ which is a high positive value, the probability of selecting $a_1$will increase significantly because the policy gradient update increases the probability of actions with positive advantage.

2- For action a_2: the advantage $A(s,\ a_2) = 10$ is also positive, but much smaller than $A(s,\ a_1)$. The probability of selecting a_2 will increase but not as much as a_1.

3- For action a_3: the advantage $A(s,\ a_2) = -10$ is negative, indicating that taking action a_3 is worse than the average action in state s. Therefore, the probability of selecting $a_3$will decrease.

The exact new probabilities will depend on the learning rate and the specific functional form of the policy (i.e. softmax), but the qualitative changes would expected to be:

- The probability of a_1 increases substantially.
- The probability of a_2 increases slightly.
- The probability of a_3 decreases substantially.

So based on the information given and typical gradient methods, the expected changes in probabilities could look similar to option (b) as shown above: a significant increase for a_1, a slight increase for a_2 and a decrease for $a_3$while maintaining a valid probability distribution (probs are non-negative & sum to one).