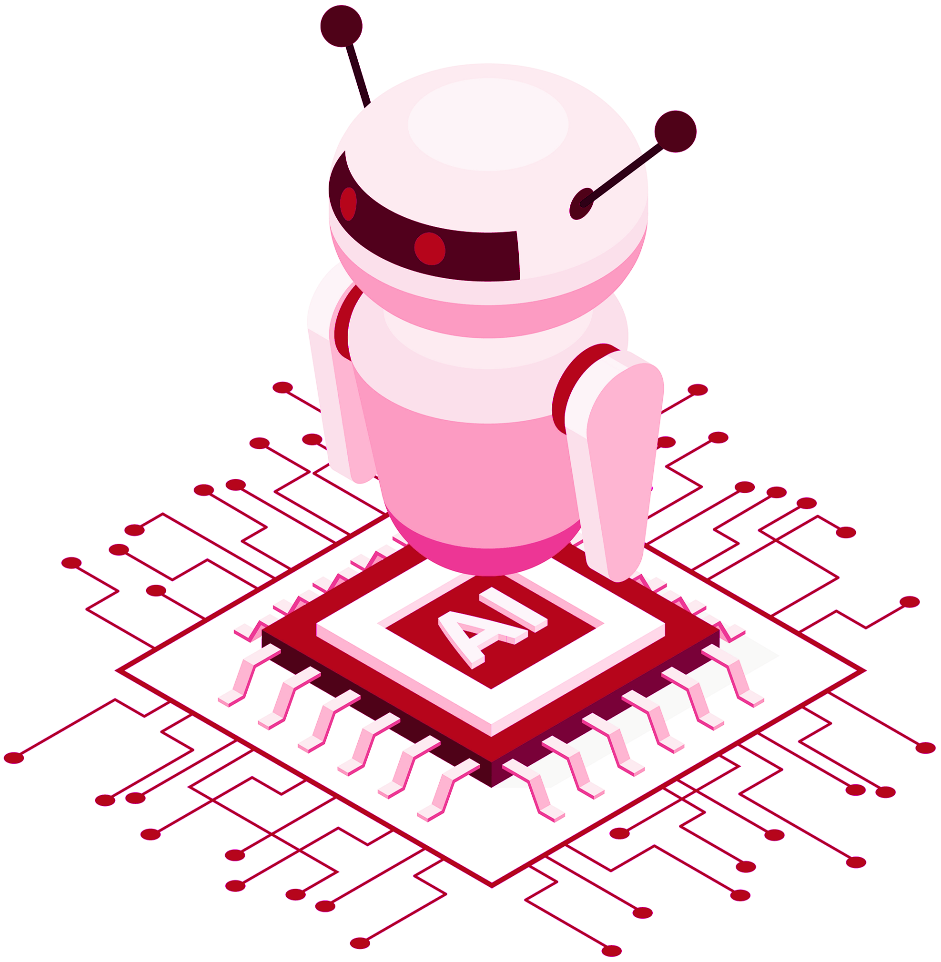

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 4:

Advanced Methods in RL

By:

[Asemaneh Nafe]

[400105285]

Spring 2025

Contents

Grading

The grading will be based on the following criteria, with a total of 100 points:

Task	**Points**
Task 1: PPO	25
Task 2: DDPG	20
Task 3: SAC	25
Task 4: Comparison between SAC & DDPG & PPO	20
Clarity and Quality of Code	5
Clarity and Quality of Report	5
Bonus 1: Writing your report in Latex	10

1 Task 1: Proximal Policy Optimization (PPO) [25]

1.1 Question 1:

What is the role of the actor and critic networks in PPO, and how do they contribute to policy optimization?

1. Role of Actor and Critic Networks in PPO In Proximal Policy Optimization (PPO), the actor network represents the policy π_θ, which determines the agent's actions based on the current state. The critic network estimates the value function V_ϕ, which helps evaluate how good a given state is in terms of expected future rewards.

The actor improves policy by maximizing expected rewards while ensuring stable updates using the clipped objective function. The critic guides the actor by reducing variance in policy updates, providing an estimate of how good a state-action pair is. PPO alternates between sampling data using the actor and updating the critic to provide more accurate value estimates. Together, they optimize the policy while preventing drastic updates that could destabilize training.

1.2 Question 2:

PPO is known for maintaining a balance between exploration and exploitation during training. How does the stochastic nature of the actor network and the entropy term in the objective function contribute to this balance?

PPO ensures a balance between exploration (trying new actions) and exploitation (using the best-known actions) using two mechanisms:

Stochastic Actor Network:

The actor outputs a probability distribution over actions, making decisions based on sampling rather than always choosing the highest-value action. This randomness allows the agent to explore new strategies instead of getting stuck in suboptimal ones.

Entropy Bonus in the Objective Function:

PPO includes an entropy term to encourage randomness in action selection, preventing the policy from becoming too deterministic too soon.

High entropy $\rightarrow$ More exploration, Low entropy $\rightarrow$ More exploitation. so by adding -0.1 * entropy to loss function minimizing the loss function lead to more entropy and it means the agent is acting more randomly, which encourages it to explore different strategies rather than sticking to a single one. As training progresses, entropy naturally decreases, shifting focus toward exploitation.

This ensures the agent explores efficiently early in training while refining its strategy as it learns.

1.3 Question 3:

When analyzing the training results, what key indicators should be monitored to evaluate the performance of the PPO agent?

Episode Reward (Return): Tracks cumulative rewards per episode; a steady increase indicates learning.

Policy Loss: Measures how much the policy is changing; large fluctuations may indicate instability.

Value Loss: Evaluates how well the critic predicts future rewards; high values may suggest inaccurate value estimates.

Entropy: Indicates exploration level; should gradually decrease but not collapse too early. Advantage Estimation: Ensures the agent is learning the correct action-state advantage values; erratic values may signal instability.

Clipping Ratio: Ensures PPO's constraint is working properly; too many clipped updates suggest over-aggressive policy updates.

2 Task 2: Deep Deterministic Policy Gradient (DDPG) [20]

2.1 Question 1:

What are the different types of noise used in DDPG for exploration, and how do they differ in terms of their behavior and impact on the learning process? In Deep Deterministic Policy Gradient (DDPG), noise is added to the action space for exploration since it is an off-policy algorithm that learns a deterministic policy. The two main types of noise used are:

Ornstein-Uhlenbeck (OU) Noise:

This is a temporally correlated noise process, meaning that the noise in the current step is influenced by the previous step. It is useful for environments with inertia, such as robotics, where actions should evolve smoothly rather than change drastically. It encourages exploration by adding momentum to actions, preventing the agent from getting stuck in local optima too soon.

Gaussian Noise:

This is a simpler alternative, where random noise is drawn independently at each step from a normal distribution. It does not have temporal correlation, meaning each action is perturbed independently. It is often preferred in modern implementations of DDPG as it is easier to tune and sufficient for exploration in many tasks. The choice of noise affects learning speed and stability. OU noise helps with smooth exploration in physics-based simulations, while Gaussian noise is more straightforward and widely used in deep RL.

2.2 Question 2:

What is the difference between PPO and DDPG regarding the use of past experiences?

The key difference between Proximal Policy Optimization (PPO) and DDPG in handling past experiences lies in their approach to policy updates:

DDPG is an off-policy algorithm, meaning it stores past experiences in a replay buffer and samples from it randomly during training. This helps stabilize learning by breaking correlations between consecutive experiences. However, it requires careful tuning of noise and hyperparameters.

PPO is an on-policy algorithm, meaning it learns directly from the most recent interactions rather than storing past experiences for later use. It optimizes a surrogate loss function while enforcing a constraint on how much the policy can change in each update, preventing drastic policy shifts.

In summary, DDPG reuses past experiences, making it sample-efficient but requiring careful replay buffer management, whereas PPO discards old data, leading to more stable updates at the cost of sample inefficiency.

3 Task 3: Soft Actor-Critic (SAC) [25]

3.1 Question 1:

Which algorithm performs better in the `HalfCheetah` environment? Why?
Compare the performance of the PPO, DDPG, and SAC agents in terms of training stability, convergence speed, and overall accumulated reward. Based on your observations, which algorithm achieves better results in this environment?

As you can see in my notebook ppo get the lowest reward and then ddpg and then sac.

Overall Accumulated Reward:

PPO performs well in terms of accumulated reward, but often requires more training time and careful hyperparameter tuning. It's robust and often achieves competitive rewards, especially when training for longer periods. DDPG may achieve good rewards faster in some cases but is prone to instability. If it converges successfully, it can perform well, but it's more likely to underperform or fail in environments like HalfCheetah.

SAC tends to perform the best in terms of accumulated reward in environments like HalfCheetah. Its combination of off-policy learning and entropy regularization gives it a good balance of exploration and exploitation, often leading to higher rewards compared to PPO and DDPG.

as you can see from my plots the ppo is more stable in training an do not have up and down move compare to SAC and DDPG.

Training Stability: PPO is generally considered more stable than DDPG and SAC, as it uses a clipped objective function to prevent large updates that might destabilize training. This makes it more reliable, especially in environments where exploration and exploitation must be carefully balanced. DDPG, being an off-policy algorithm, often suffers from instability issues, particularly in environments with high-dimensional action spaces like HalfCheetah. It can have problems with overestimation of Q-values and may require more careful tuning of hyperparameters. SAC, like DDPG, is off-policy, but it uses entropy regularization, which encourages exploration. This can result in better stability than DDPG, as it doesn't rely solely on value function approximation. SAC is often more stable than DDPG, but can still be less stable than PPO in some environments.

as you can see from my num episodes in my note book:

PPO tends to have slower convergence compared to SAC and DDPG. While PPO is stable, it may require more iterations to achieve high performance because it conservatively updates policies. DDPG can converge faster in certain environments since it's able to directly optimize a deterministic policy using experience replay, but this is at the cost of stability. It may require careful tuning of hyperparameters and exploration strategies to prevent overfitting or poor convergence. SAC generally converges faster than PPO due to its use of off-policy learning and entropy regularization, which helps it explore the environment more effectively and avoid premature convergence to suboptimal solutions.

3.2 Question 2:

How do the exploration strategies differ between PPO, DDPG, and SAC?
Compare the exploration mechanisms used by each algorithm, such as deterministic vs. stochastic policies,

entropy regularization, and noise injection. How do these strategies impact learning in environments with continuous action spaces?

The exploration strategies in reinforcement learning algorithms like PPO, DDPG, and SAC are crucial for how these agents balance exploration (trying new actions) and exploitation (choosing actions known to yield high rewards). These strategies differ in their approach and impact learning, especially in environments with continuous action spaces like HalfCheetah. Here's how each algorithm tackles exploration:

1. PPO (Proximal Policy Optimization): Exploration Mechanism: PPO uses a stochastic policy that relies on the concept of probability distributions (typically Gaussian) over actions. This means that even after learning a good policy, the agent will still choose actions probabilistically, encouraging exploration by occasionally selecting actions that are suboptimal according to the current policy.

Entropy Regularization: PPO uses entropy regularization implicitly by encouraging exploration in its objective function. The policy is optimized to not only maximize rewards but also to maintain a certain level of entropy (i.e., randomness in action selection). This helps ensure that the agent does not prematurely converge to a deterministic policy, thus maintaining exploration.

Impact on Learning: The stochastic nature of the policy allows PPO to explore the environment more diversely compared to deterministic algorithms. This is especially important in continuous action spaces, as it allows the agent to cover more of the action space. However, it can also slow convergence since the agent continues to explore even as it gets closer to an optimal policy.

2. DDPG (Deep Deterministic Policy Gradient): Exploration Mechanism: DDPG uses a deterministic policy, where the action selection is based on the output of the deterministic policy network. This means the agent consistently selects the same action for a given state once the policy is learned. To encourage exploration, DDPG adds noise to the action during training. Typically, Ornstein-Uhlenbeck noise is injected into the action at each timestep, which allows the agent to explore different regions of the action space.

Entropy Regularization: DDPG does not use entropy regularization, which can lead to less diversity in exploration compared to stochastic methods. The addition of noise helps to counteract this, but the exploration is still less diverse and more targeted toward regions of the action space that the agent has already explored.

Impact on Learning: The noise injection helps in exploration but is often less efficient than stochastic exploration mechanisms. As the agent continues learning, the noise is typically reduced over time, and the agent moves toward more deterministic actions. This means DDPG can struggle to maintain sufficient exploration over long training periods, especially in complex continuous environments.

3. SAC (Soft Actor-Critic): Exploration Mechanism: SAC uses a stochastic policy similar to PPO, where actions are sampled from a Gaussian distribution. However, SAC goes a step further by integrating entropy regularization into the objective function. This ensures that the agent is encouraged to maintain a high level of entropy throughout training, preventing premature convergence to suboptimal deterministic policies.

Entropy Regularization: SAC directly optimizes for maximum entropy alongside reward maximization, promoting exploration by encouraging the policy to remain stochastic even as training progresses. The entropy term ensures that the agent does not become too deterministic too early, which is especially important in continuous action spaces where the exploration of diverse actions can help discover better policies.

Impact on Learning: The combination of a stochastic policy and entropy regularization in SAC leads to very efficient exploration, as the agent tends to explore the action space more effectively and less randomly than PPO. SAC's exploration strategy is generally more robust, allowing it to perform better in complex environments with continuous action spaces by balancing exploration and exploitation well throughout training.

3.3 Question 3:

What are the key advantages and disadvantages of each algorithm in terms of sample efficiency and stability?
Discuss how PPO, DDPG, and SAC handle sample efficiency and training stability. Which algorithm is more sample-efficient, and which one is more stable during training? What trade-offs exist between these properties?

PPO (Proximal Policy Optimization):

Advantages: PPO is relatively stable due to its clipped objective function, which prevents large, unstable updates. It works well in many environments without requiring complex tuning.

Disadvantages: PPO is less sample-efficient compared to off-policy algorithms like DDPG and SAC. It requires more data to converge, as it updates policies based on batches of experience rather than individual transitions.

Trade-off: Stability comes at the cost of sample efficiency, as PPO's conservatism slows down learning.

DDPG (Deep Deterministic Policy Gradient):

Advantages: DDPG is sample-efficient in environments with continuous action spaces because it is off-policy, learning from past experiences stored in a replay buffer.

Disadvantages: It can be unstable during training, especially in high-dimensional environments. It is prone to issues like overestimation of Q-values and poor exploration due to the lack of entropy regularization.

Trade-off: DDPG's sample efficiency is offset by potential instability and sensitivity to hyperparameters.
SAC (Soft Actor-Critic):

Advantages: SAC is highly sample-efficient and stable, combining off-policy learning with entropy regularization, which encourages better exploration and smoother learning curves.

Disadvantages: SAC requires more computational resources and hyperparameter tuning to balance exploration and exploitation effectively.

Trade-off: SAC provides a good balance between sample efficiency and stability but at the cost of increased complexity and computation.

Conclusion: SAC is the most sample-efficient and stable, while PPO is stable but less efficient, and DDPG is sample-efficient but less stable.

3.4 Question 3:

Which reinforcement learning algorithm—PPO, DDPG, or SAC—is the easiest to tune, and what are the most critical hyperparameters for ensuring stable training for each agent?
How sensitive are PPO, DDPG, and SAC to hyperparameter choices, and which parameters have the most significant impact on stability? What common tuning strategies can help improve performance and prevent instability in each algorithm?

PPO:

Ease of Tuning: PPO is relatively easy to tune compared to DDPG and SAC due to its more stable nature. It generally requires fewer adjustments and works well out of the box for many environments.

Critical Hyperparameters: The most important ones include the clip range (for the objective function), learning rate, and discount factor.

Sensitivity: PPO is less sensitive to hyperparameters but can become unstable with large clip ranges or improper learning rates.

Tuning Strategy: Focus on a moderate clip range (usually 0.1-0.3) and use a learning rate scheduler to ensure steady progress.

DDPG:

Ease of Tuning: DDPG is challenging to tune due to its sensitivity to hyperparameters and instability during training.

Critical Hyperparameters: Key parameters include the learning rate, target smoothing coefficient, exploration noise, and batch size.

Sensitivity: DDPG is highly sensitive to the noise schedule and the learning rate, and poorly tuned noise can lead to poor exploration.

Tuning Strategy: Gradually decrease the noise level over time and ensure the learning rate is not too high to prevent divergence.

SAC:

Ease of Tuning: SAC is more complex to tune than PPO but easier than DDPG. It requires balancing exploration (entropy) and exploitation carefully. Critical Hyperparameters: Important ones are the learning rate, alpha (entropy regularization weight), target entropy, and Q-function update frequency.

Sensitivity: SAC is less sensitive than DDPG but still requires careful tuning of entropy parameters (alpha and target entropy) to maintain exploration.

Tuning Strategy: Start with default values for alpha and target entropy, and adjust based on the stability of the reward curve.

Conclusion: PPO is the easiest to tune, followed by SAC, while DDPG is the most challenging. For stable training, focus on the learning rate, exploration strategies, and entropy regularization for each algorithm.

4 Task 4: Comparison between SAC & DDPG & PPO [20]

4.1 Question 1:

Which algorithm performs better in the `HalfCheetah` environment? Why?
Compare the performance of the PPO, DDPG, and SAC agents in terms of training stability, convergence speed, and overall accumulated reward. Based on your observations, which algorithm achieves better results in this environment?

4.2 Question 2:

How do the exploration strategies differ between PPO, DDPG, and SAC?
Compare the exploration mechanisms used by each algorithm, such as deterministic vs. stochastic policies, entropy regularization, and noise injection. How do these strategies impact learning in environments with continuous action spaces?

4.3 Question 3:

What are the key advantages and disadvantages of each algorithm in terms of sample efficiency and stability?
Discuss how PPO, DDPG, and SAC handle sample efficiency and training stability. Which algorithm is more sample-efficient, and which one is more stable during training? What trade-offs exist between these properties?

4.4 Question 4:

Which reinforcement learning algorithm—PPO, DDPG, or SAC—is the easiest to tune, and what are the most critical hyperparameters for ensuring stable training for each agent?
How sensitive are PPO, DDPG, and SAC to hyperparameter choices, and which parameters have the most significant impact on stability? What common tuning strategies can help improve performance and prevent instability in each algorithm?