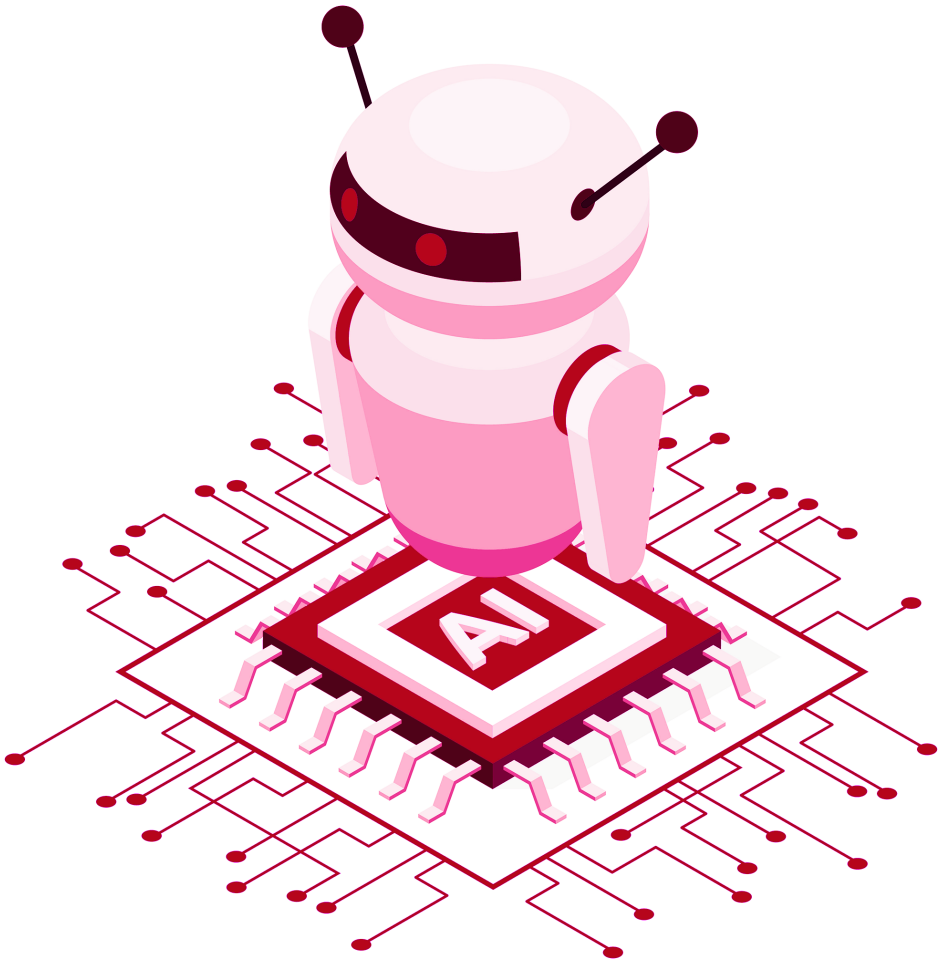

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 4:

Advanced Methods in RL

By:

Amir Kooshan Fattah Hesari

401102191

RIML

Deep RL Course

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. Actor Network:

The actor network changes the **policy function**, which is typically represented as a probability distribution over possible actions given a state. The policy function, denoted as $\pi_\theta(a|s)$, is parameterized by the actor network with parameters θ. The actors role is to update this policy to maximize the expected cumulative reward.

$$\pi_\theta(a|s)$$

This function defines the probability of selecting action a given state s. The actor modifies this function by adjusting its parameters θ based on feedback from the critic.

In PPO, the actor is updated through a **clipped objective function** that ensures that the new policy doesn't deviate too drastically from the old policy, avoiding large policy updates that could destabilize training. The update is typically done using the following objective:

$$L(\theta) = \mathbb{E}_t \left[\min \left(\hat{r}_t(\theta)\hat{A}_t, \text{clip}(\hat{r}_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where:

- $\hat{r}_t(\theta)$ is the importance sampling ratio, which measures how much the new policy deviates from the old policy.
- $\hat{A}_t$ is the advantage estimate, which measures how good the action was compared to the average action.
- ϵ is a small hyperparameter controlling the clipping range.

2. Critic Network:

The critic network changes the **value function**, which estimates the expected cumulative reward (or return) from a given state. The critic is responsible for updating the value function $V_\phi(s)$, where ϕ represents the parameters of the critic network.
In PPO, the critic is typically trained to minimize the **mean squared error (MSE)** between the predicted value and the actual return (the target value), which is calculated as:

$$\mathcal{L}_{\text{value}} = \mathbb{E}_t \left[(V_\phi(s_t) - R_t)^2 \right]$$

where:

- R_t is the actual return, often computed using Generalized Advantage Estimation (GAE).
- $V_\phi(s_t)$ is the predicted value from the critic for state s_t.

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?

In Proximal Policy Optimization (PPO), the balance between exploration and exploitation is maintained through the **stochastic nature of the actor network** and the **entropy term** in the objective function. The actor network outputs a probability distribution over actions, introducing **stochasticity** in action selection. This allows the agent to explore different actions (exploration) while also exploiting actions that are known to yield high rewards (exploitation) as training progresses.

The entropy term in the objective function encourages exploration by penalizing deterministic policies. A higher entropy encourages **exploration** by maintaining uncertainty in the action selection, while a lower entropy focuses on **exploitation**. The entropy term is weighted by a hyperparameter c to control the balance between exploration and exploitation.

$$L(\theta) = \mathbb{E}_t \left[\min \left(\hat{r}_t(\theta)\hat{A}_t, \text{clip}(\hat{r}_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] + c \cdot \mathbb{H}(\pi_\theta)$$

Together, the stochasticity of the actor network and the entropy term help the agent balance exploration and exploitation during training.

1.3 Question 3:

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

While trying to train the agent using the PPO algorithm, I saw that the algorithm is extremely sensitive to learning rate and finding an optimal learning rate can greatly influence the outcome of the training. Also the λ for the GAE calculation played a significant role in reducing the variance of the results.

On another note, the value and policy loss help us ensure that the update of the policy is making it better, not worse.

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), two types of noise are used for exploration:

1. Ornstein-Uhlenbeck (OU) Noise:

OU noise is a temporally correlated noise process, where the noise at time t depends on the previous noise at $t-1$. This smoothness is useful for continuous action spaces.

$$\xi_t = \theta(\mu - \xi_{t-1}) + \sigma \cdot \mathcal{N}(0,1)$$

where: - ξ_t is the noise at time t, - μ is the mean, - θ controls the decay rate, - σ is the noise scale.

Behavior and Impact: OU noise enables smooth exploration with gradual changes in actions, ideal for continuous control tasks (e.g., robotic control).

2. Gaussian Noise:

Gaussian noise is independent at each time step and is sampled from a standard Gaussian distribution.

$$\epsilon_t \sim \mathcal{N}(0, \sigma^2)$$

Behavior and Impact: Gaussian noise is more random and less smooth, leading to more erratic exploration. It can be used for random exploration when smoothness isn't necessary.

Differences:

- **OU noise** is temporally correlated, promoting smooth, continuous action adjustments, making it suitable for continuous control tasks.
- **Gaussian noise** is independent at each time step, leading to more random exploration but can be less effective for environments requiring smooth transitions in actions.
In DDPG, **OU noise** is preferred for continuous control problems due to its smoother exploration behavior.

2.2 Question 2:

What is the difference between PPO and DDPG regarding the use of past experiences?
The difference between PPO and DDPG regarding the use of past experiences lies in their approach to experience handling:

- PPO (Proximal Policy Optimization)

  - **On-policy**: PPO uses only *current experiences* generated by the latest policy to update the policy.
  - It does not store or reuse past experiences for training.
- DDPG (Deep Deterministic Policy Gradient)
  - Off-policy: DDPG stores experiences in a *replay buffer* and can reuse past experiences for training.
  - It improves sample efficiency by learning from past interactions.

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

3.1 Question 1:

Why do we use two Q-networks to estimate Q-values?
The algorithm that we are using here is the twin Delayed DDPG.
TD3 concurrently learns two Q-functions, Q_{ϕ_1} and Q_{ϕ_2}, by mean square Bellman error minimization, in almost the same way that DDPG learns its single Q-function. To show exactly how TD3 does this and how it differs from normal DDPG, well work from the innermost part of the loss function outwards.

First: target policy smoothing. Actions used to form the Q-learning target are based on the target policy, $\mu_{\theta_{\text{targ}}}$, but with clipped noise added on each dimension of the action. After adding the clipped noise, the target action is then clipped to lie in the valid action range (all valid actions, a, satisfy $a_{\text{Low}} \leq a \leq a_{\text{High}}$). The target actions are thus:

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}\right), \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

Target policy smoothing essentially serves as a regularizer for the algorithm. It addresses a particular failure mode that can happen in DDPG: if the Q-function approximator develops an incorrect sharp peak for some actions, the policy will quickly exploit that peak and then have brittle or incorrect behavior. This can be averted by smoothing out the Q-function over similar actions, which target policy smoothing is designed to do.

Next: clipped double-Q learning. Both Q-functions use a single target, calculated using whichever of the two Q-functions gives a smaller target value:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s'))$$

and then both are learned by regressing to this target:

$$L(\phi_1, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{\phi_1}(s,a) - y(r,s',d)\right)^2\right]$$

$$L(\phi_2, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{\phi_2}(s,a) - y(r,s',d)\right)^2\right]$$

Using the smaller Q-value for the target, and regressing towards that, helps fend off overestimation in the Q-function.

Lastly: the policy is learned just by maximizing Q_{ϕ_1}:

$$\max_{\theta} \mathbb{E}_{s\sim\mathcal{D}}\left[Q_{\phi_1}(s, \mu_\theta(s))\right]$$

which is pretty much unchanged from DDPG. However, in TD3, the policy is updated less frequently than the Q-functions are. This helps damp the volatility that normally arises in DDPG because of how a policy

update changes the target.
OpenAI Spinning Up gives a good explanation of the inner working of different implementations of DDPG

3.2 Question 2:

What is the temperature parameter(α), and what is the benefit of using a dynamic α in SAC?
In Soft Actor-Critic (SAC), the temperature parameter α controls the trade-off between exploration and exploitation by adjusting the weight of the entropy term in the objective function. The objective is:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_t r_t - \alpha \cdot \mathcal{H}(\pi(\cdot|s_t)) \right]$$

α determines how much the entropy (exploration) influences the learning process. A higher α encourages more exploration, while a lower α focuses on exploitation.

Benefit of Dynamic α: In SAC, α is dynamically adjusted during training. This allows:

- Early stages: Higher α for more exploration.
- Later stages: Lower α for more exploitation.

This dynamic adjustment helps balance exploration and exploitation, improving learning stability and efficiency.

3.3 Question 3:

What is the difference between evaluation mode and training mode in SAC?
In Soft Actor-Critic (SAC), the difference between evaluation mode and training mode can be understood as follows:

- **Training Mode:**
  - In training mode, the agent interacts with the environment and collects experiences that are used to update the policy and value networks.
  - The policy is stochastic, meaning the agent explores the environment by taking actions based on the probability distribution over actions, which is encouraged by the entropy term.
  - The agent learns from both exploration and exploitation, gradually improving its policy by balancing between these two through the entropy mechanism.
- **Evaluation Mode:**
  - In evaluation mode, the agent's performance is assessed using the learned policy without exploration (no randomness in action selection).
  - The policy is typically deterministic (greedy), meaning it selects the action that maximizes the expected reward, based on the learned knowledge.
  - Evaluation mode is used to measure the agents performance after training, with no random exploration influencing the results.

Key Difference: In training mode, exploration is encouraged through a stochastic policy, while in evaluation mode, the agent acts deterministically to evaluate its learned policy.

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?

In the HalfCheetah environment, the performance of the algorithms PPO,DDPG, and SAC can be compared based on the following criteria:

- **Training Stability:**
  - **PPO:** PPO tends to offer more stable training due to its clipped objective function, which prevents excessively large policy updates. This stability is crucial in continuous control environments like HalfCheetah.
  - **DDPG:** DDPG can experience unstable training, especially due to its reliance on off-policy updates and experience replay. The volatility of the policy update can lead to instability, particularly in complex environments.
  - **SAC:** SAC is generally very stable due to the combination of off-policy updates, entropy regularization, and the use of both value and policy networks. The entropy term helps maintain exploration, preventing premature convergence to suboptimal policies.
- **Convergence Speed:**
  - **PPO:** PPO typically has a moderate convergence speed, balancing exploration and exploitation. However, since it is on-policy, it may require more interactions with the environment to converge compared to off-policy methods.
  - **DDPG:** DDPG can sometimes converge faster due to off-policy learning, especially in simpler environments. However, it may struggle with environments requiring fine-tuned exploration, such as HalfCheetah.
  - **SAC:** SAC usually converges faster than both PPO and DDPG due to its more efficient use of experiences (off-policy) and the entropy term that encourages more exploration in the early stages of training.
- **Overall Accumulated Reward:**
  - **PPO:** PPO can achieve strong performance and is robust, but it may not always reach the highest accumulated reward in highly dynamic environments like HalfCheetah.
  - **DDPG:** DDPG may struggle with accumulated rewards in environments like HalfCheetah due to its exploration-exploitation trade-off being harder to balance and issues with training stability.
  - **SAC:** SAC tends to perform the best in terms of accumulated reward, as it combines efficient exploration with a stable, off-policy learning process. The entropy regularization promotes

consistent exploration, leading to a better policy over time.

Conclusion: Based on the observations, SAC performs the best in the HalfCheetah environment. It achieves superior results due to its off-policy nature, faster convergence, better stability, and consistent exploration. While PPO offers stability and DDPG can converge faster in simpler tasks, SAC is more effective in environments that require fine-tuned control, like HalfCheetah.

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?

The exploration strategies used by PPO, DDPG, and SAC differ in how they balance exploration and exploitation, particularly in environments with continuous action spaces.

- **PPO (Proximal Policy Optimization):**
  - **Exploration Mechanism:** PPO uses a stochastic policy, where actions are selected based on a probability distribution (e.g., Gaussian distribution for continuous action spaces).
  - **Entropy Regularization:** PPO encourages exploration through entropy regularization, which adds a penalty to the objective for deterministic policies. This encourages the agent to maintain a certain level of randomness in action selection, promoting exploration.
  - **Impact on Learning:** The use of stochastic policies and entropy regularization allows PPO to explore the environment while gradually converging to optimal actions. This is effective in continuous action spaces as it helps prevent the policy from prematurely exploiting suboptimal actions.
- **DDPG (Deep Deterministic Policy Gradient):**
  - **Exploration Mechanism:** DDPG uses a deterministic policy (a single action is selected given the state) but injects Gaussian noise to the action output to promote exploration.
  - **Noise Injection:** Gaussian noise is added directly to the action to introduce randomness. In continuous action spaces, this noise is critical for exploration, but it can lead to inefficient exploration if not handled properly.
  - **Impact on Learning:** While the deterministic policy is efficient for exploitation, the noise injection can sometimes cause erratic exploration, making it harder for DDPG to find optimal policies in environments where fine-grained exploration is necessary. The method can also suffer from instability during training due to overestimation in the Q-value function.
- **SAC (Soft Actor-Critic):**
  - **Exploration Mechanism:** SAC uses a stochastic policy similar to PPO but with an additional focus on entropy regularization. The policy is trained to maximize both the expected reward and the entropy of the action distribution.
  - **Entropy Regularization:** SAC explicitly incorporates entropy into its objective function, which encourages the agent to explore more diverse actions. The entropy term increases exploration early in training and decreases over time as the agent learns the environment.

  - **Impact on Learning:** The entropy regularization in SAC helps it explore the action space more efficiently than DDPG, preventing premature convergence to suboptimal policies. This is particularly beneficial in continuous action spaces, where SAC can strike a balance between exploration and exploitation, leading to faster and more stable learning.

Key Differences and Impact on Learning:

- PPO promotes exploration through stochastic policies and entropy regularization, making it effective in maintaining a balance between exploration and exploitation.
- DDPG uses deterministic policies with noise injection, which is efficient for exploitation but can struggle with exploration, especially in environments that require more refined exploration strategies.
- SAC combines stochastic policies with entropy regularization, ensuring efficient exploration early in training and more stable learning in continuous action spaces, leading to better overall performance.

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?

- **PPO (Proximal Policy Optimization):**
  - **Sample Efficiency:** PPO is relatively less sample-efficient as it is an on-policy algorithm, requiring fresh experiences for each update. It needs more interactions with the environment to converge.
  - **Stability:** PPO is stable due to its clipped objective function that prevents large policy updates, ensuring more consistent training.
- **DDPG (Deep Deterministic Policy Gradient):**
  - **Sample Efficiency:** DDPG is more sample-efficient than PPO because it is off-policy and can reuse past experiences stored in a replay buffer.
  - **Stability:** DDPG is less stable due to the reliance on experience replay, which can lead to instability, especially when the Q-value function overestimates.
- **SAC (Soft Actor-Critic):**
  - **Sample Efficiency:** SAC is highly sample-efficient due to its off-policy nature and the use of entropy regularization, which helps improve exploration and learning speed.
  - **Stability:** SAC is very stable due to entropy regularization, which prevents premature convergence and helps maintain a good exploration-exploitation balance.

Key Trade-offs:

- **PPO:** Less sample-efficient but highly stable, making it suitable for environments where stability is more important than sample efficiency.

- **DDPG:** More sample-efficient but less stable, leading to challenges in complex environments with high variance in training.
- **SAC:** Offers the best balance, being both sample-efficient and stable, making it ideal for continuous control tasks where fast learning and stability are crucial.

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?

- **PPO (Proximal Policy Optimization):**
  - **Ease of Tuning:** PPO is relatively easy to tune compared to DDPG and SAC. It has fewer hyperparameters that need to be carefully adjusted.
  - **Critical Hyperparameters:**
    * Clipping range (ϵ): Affects the stability of policy updates.
    * Learning rate: Controls how quickly the policy network updates.
    * Mini-batch size: Influences the variance in gradient estimates.
  - **Sensitivity to Hyperparameters:** PPO is moderately sensitive to hyperparameters, but careful tuning of the clipping range and learning rate ensures stable training.
- **DDPG (Deep Deterministic Policy Gradient):**
  - **Ease of Tuning:** DDPG is more challenging to tune due to its reliance on experience replay and deterministic policy.
  - **Critical Hyperparameters:**
    * Learning rate for actor and critic networks.
    * Batch size for experience replay.
    * Noise scale for exploration (Gaussian noise).
  - **Sensitivity to Hyperparameters:** DDPG is very sensitive, especially to the learning rate and noise parameters, which can significantly impact both stability and exploration.
- **SAC (Soft Actor-Critic):**
  - **Ease of Tuning:** SAC is easier to tune than DDPG, as the entropy term helps maintain stability during training.
  - **Critical Hyperparameters:**
    * Temperature parameter (α): Controls the balance between exploration and exploitation.
    * Learning rate for actor and critic networks.

    - Target smoothing coefficient.
  - **Sensitivity to Hyperparameters:** SAC is less sensitive than DDPG, but the temperature parameter and learning rate are crucial for stability and performance.

Common Tuning Strategies:

- Use a small learning rate to prevent large, unstable updates.
- Tune exploration parameters (e.g., noise scale in DDPG, entropy in PPO and SAC) to balance exploration and exploitation.
- Gradually decay learning rates and entropy to allow for fine-tuning of the policy as training progresses.
- Monitor stability by tracking training curves and adjusting hyperparameters dynamically.