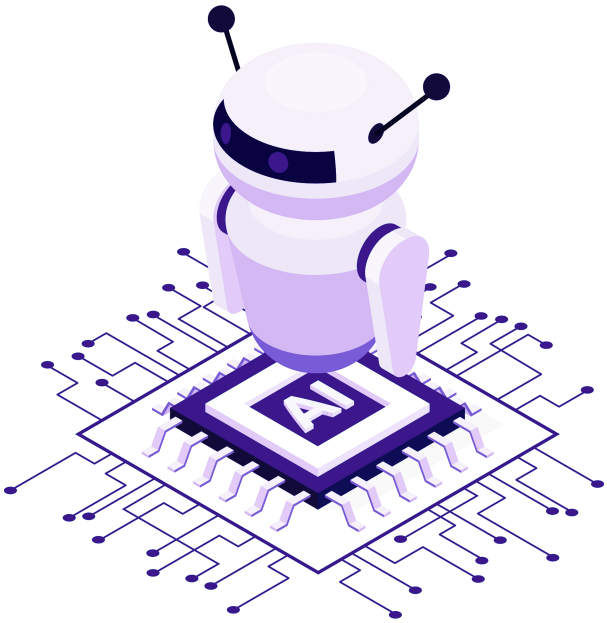


Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 4:

Advanced Methods in RL

Designed By:

Ahmad Karami

ahmad.karami77@yahoo.com

Hamidreza Ebrahimpour

ebrahimpour.7879@gmail.com

RIML

Spring 2025

Preface

Welcome to the homework!

Reinforcement Learning (RL) is a branch of Artificial Intelligence that aims to determine an **optimal policy** for an agent. The objective is to enable the agent to make decisions that maximize its **cumulative reward**, ensuring it efficiently completes a given task.

In the previous exercises, you were introduced to the fundamental concepts of RL and some basic RL methods, such as **value-based methods** and **policy-based methods**. This assignment focuses on more advanced methods that integrate value-based and policy-based methods. Typically, these algorithms select actions using their policy network and evaluate the policy based on their value network.

Three well-known algorithms in this category are **Proximal Policy Optimization (PPO)**, **Soft Actor-Critic (SAC)**, and **Deep Deterministic Policy Gradient (DDPG)**, all widely used for reinforcement learning in continuous and discrete action-space environments.

In this assignment, we focus on continuous action-space environments. We begin by implementing a PPO agent and examining its ability to optimize policies through iterative updates while maintaining stability. Next, we develop a SAC agent that leverages entropy regularization to balance exploration and exploitation. Then, we implement a DDPG agent, which selects deterministic actions and explores the environment by adding noise to those actions. Finally, we compare the performance of these methods by analyzing their cumulative rewards per episode and the behavior of their loss functions.

This assignment explores and compares the **PPO**, **SAC**, and **DDPG** algorithms for continuous action-space environments. You will:

- **PPO agent** – Implement a PPO agent for a continuous action-space environment.
- **SAC agent** – Implement a SAC agent for a continuous action-space environment.
- **DDPG agent** – Implement a DDPG agent for a continuous action-space environment.
- **Comparison between SAC, DDPG and PPO** – Analyze and compare the performance of SAC, DDPG and PPO in a specific environment.

By completing this assignment, you will gain valuable insights into the effectiveness and trade-offs of these algorithms in different reinforcement learning scenarios.

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

Submission

The deadline for this homework is 1403/12/24 (March 14th 2025) at 11:59 PM.

Please submit your work by following the instructions below:

- Zip all the files together with the following naming format:

  `DRL_HW4_[StudentNumber]_[FullName].zip`

  - Replace `[FullName]` and `[StudentNumber]` with your full name and student number, respectively. Your `[FullName]` must be in CamelCase with no spaces.
- Submit the zip file through Quera in the appropriate section.
- We provided this LaTeX template for writing your homework solution. There is a 10-point bonus for writing your solution in LaTeX using this template and including your LaTeX source code in your submission, named `HW4_Solution.zip`.
- If you have any questions about this homework, please ask them in the Homework section of our Telegram Group.
- If you are using any references to write your answers, consulting anyone, or using AI, please mention them in the appropriate section. In general, you must adhere to all the rules mentioned here and here by registering for this course.

Keep up the great work and best of luck with your submission!

Contents

1 Part 1 (Setup Instructions)

Before starting this assignment, ensure your environment is correctly set up with the required libraries and dependencies. The practical component of this homework will be completed in the provided Jupyter Notebooks:

- `PPO_continuous.ipynb`
- `SAC_DDPG_continuous.ipynb`

These notebooks are attached along with this document.

1.1 Environment Setup

The provided Jupyter Notebooks rely on several essential Python packages, which must be installed to avoid errors.

- `torch.optim (Adam)` – Implements the Adam optimization algorithm for training neural networks.
- `torch` – The core PyTorch library for tensor operations and deep learning models.
- `torch.nn` – Provides modules and classes for building neural network layers.
- `torch.distributions (Normal)` – Defines probability distributions for stochastic policy sampling.
- `random` – Provides functions for generating random numbers and controlling randomness in experiments.
- `numpy (np)` – A fundamental package for numerical computing and handling large arrays.
- `logging` – Used for tracking events and debugging information.
- `matplotlib` – A plotting library for visualizing data and training progress.
- `matplotlib.pyplot (plt)` – Provides a MATLAB-like interface for creating figures and plots.
- `imageio` – Reads and writes images and animated sequences, useful for saving training visualizations.
- `IPython.display` – Displays HTML content, often used for embedding animations.
- `base64 (b64encode)` – Encodes binary data in Base64 format, typically for embedding media files.
- `IPython.display (display)` – Enables interactive content rendering in Jupyter notebooks.
- `torch.nn.functional` – Contains functions for activation functions, loss functions, and other neural network utilities.
- `gymnasium (gym)` – A toolkit for testing reinforcement learning environments, providing standardized APIs for interaction.

To install the required libraries, use the following command in your terminal or Jupyter Notebook if necessary:

```
!pip install [libraries' name]
```

To work with the `HalfCheetah` environment, run the following command:

```
!pip install mujoco==2.3.3
```

1.2 Submission Requirements

- Ensure that your Jupyter Notebooks run without errors before submission.
- Include all code, outputs, and plots within the notebooks.
- Submit a ZIP file containing:
  - The completed notebooks:
    - `PPO_continuous.ipynb`
    - `SAC_DDPG_continuous.ipynb`
  - The appropriate reports.
- If you encounter any issues, please reach out on the course forum or contact the teaching assistants.

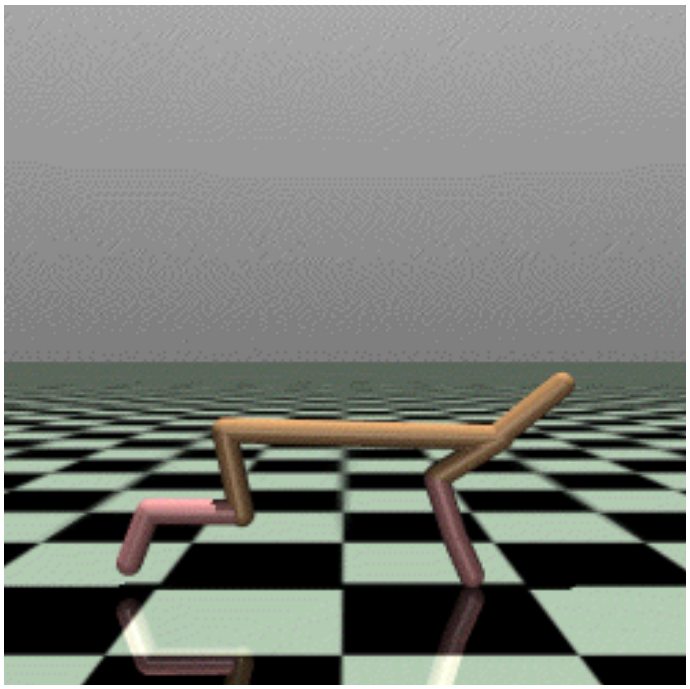
2 Part 2 (Problem Descriptions)

This assignment focuses on advanced policy-based reinforcement learning algorithms specifically designed for continuous action spaces. With step-by-step guidance, you will implement and analyze key algorithms, including Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Deep Deterministic Policy Gradient (DDPG). The tasks involve developing and fine-tuning these agents, understanding their optimization processes, and comparing their performance. Throughout the assignment, you are expected to complete the provided tasks, analyze the results, and answer conceptual questions related to the strengths and limitations of each method.

The `HalfCheetah` environment, part of the MuJoCo physics simulator, is a widely used benchmark for continuous control in reinforcement learning. It features a 2D robotic cheetah with **six controllable joints**, designed to move forward efficiently.

- **Action Space**: The environment has a **6-dimensional continuous action space**, where each dimension represents the torque applied to one of the six joints.
- **State Space**: The state is represented by a **17-dimensional vector**, including the position, velocity, and joint angles of the robot.
- **Reward Function**: The agent receives a reward based on its forward velocity while being penalized for excessive energy consumption, encouraging both speed and efficiency.
- **Challenges**: Learning a stable gait requires balancing exploration and exploitation, handling contact dynamics, and optimizing energy efficiency.

This environment is beneficial for evaluating reinforcement learning algorithms like PPO, SAC, and DDPG, as it requires precise control and policy optimization in a high-dimensional continuous action space.



You can also see more details about this environment in this link.

2.1 Task 1: Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a widely used reinforcement learning algorithm that belongs to the family of policy gradient methods. It improves policy learning stability by using a clipped surrogate objective function, which prevents excessively large policy updates that could destabilize training. Due to its simplicity, efficiency, and robustness, PPO is one of the most popular algorithms for solving reinforcement learning tasks in both continuous and discrete action spaces.

2.1.1 Task Overview

The provided notebook `PPO_continuous.ipynb` contains detailed explanations and guidance to help you complete this task. Additionally, you should give appropriate answers to a set of conceptual questions. However, you do not need to write your answers directly in the notebook due to space constraints. Instead, please submit them separately.

2.1.2 Questions

1. What is the role of the actor and critic networks in PPO, and how do they contribute to policy optimization?
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?
3. When analyzing the training results, what key indicators should be monitored to evaluate the performance of the PPO agent?

2.2 Task 2: Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) is an off-policy reinforcement learning algorithm designed for continuous action-space environments. It combines the advantages of Deterministic Policy Gradient (DPG) and Deep Q-Networks (DQN) by using an actor-critic architecture, where the actor selects actions deterministically, and the critic estimates the action-value function. To encourage exploration, DDPG introduces noise into the actions during training. Additionally, it employs target networks and experience replay to improve learning stability. Due to its ability to handle high-dimensional continuous control tasks efficiently, DDPG is widely used in robotics and other real-world applications.

2.2.1 Task Overview

The notebook `SAC_DDPG_continuous.ipynb` offers detailed explanations and guidance to assist you in completing this task. You should also answer some conceptual questions based on your implementation and results.

2.2.2 Questions

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

2.3 Task 3: Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is an off-policy, model-free reinforcement learning algorithm that combines the benefits of both value-based and policy-based methods. SAC introduces the concept of entropy regularization, which encourages exploration by maximizing both expected reward and entropy (a measure of randomness in the policy). This allows SAC to balance exploration and exploitation more effectively than traditional methods. SAC is known for its sample efficiency, stability, and ability to handle complex tasks in environments with high-dimensional action spaces.

2.3.1 Task Overview

The notebook `SAC_DDPG_continuous.ipynb` offers detailed explanations and guidance to assist you in completing this task. You should also answer some conceptual questions based on your implementation and results.

2.3.2 Questions

1. Why do we use two Q-networks to estimate Q-values?
2. What is the temperature parameter(α), and what is the benefit of using a dynamic α in SAC?
3. What is the difference between evaluation mode and training mode in SAC?

2.4 Task 4: Comparison between SAC & DDPG & PPO

In this section, we will compare the key differences and similarities between three popular reinforcement learning algorithms: Soft Actor-Critic (SAC), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO). We will focus on their approaches to policy learning, exploration strategies, and performance in continuous action spaces.

2.4.1 Questions

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?
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?
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. **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?

3 References

[1] Cover image designed by freepik

[2] Half Cheetah environment

[3] Some other continuous action-space environments

[4] Proximal Policy Optimization Algorithms

[5] Continuous control with deep reinforcement learning

[6] Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

[7] Soft Actor-Critic Algorithms and Applications