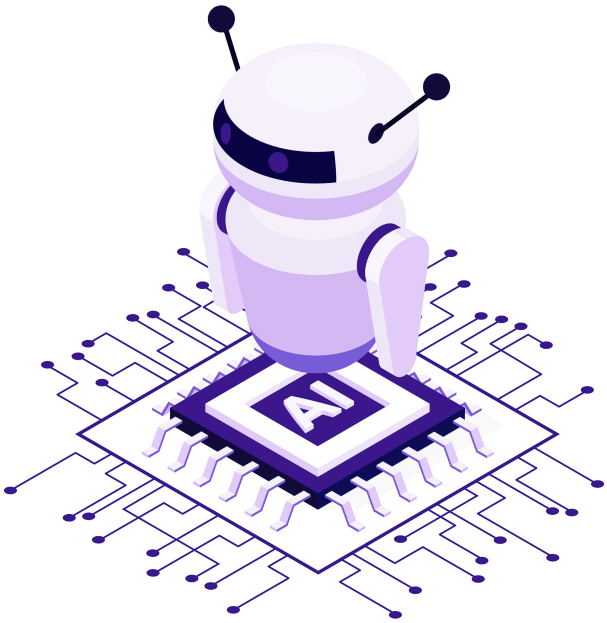


Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 3:

Policy-Based Methods

Designed By:

Nima Shirzady

shirzady.1934@gmail.com

SeyyedAli MirGhasemi

sam717269@gmail.com

RIML

Spring 2025

Preface

Welcome to the homework!

As you may know, **Reinforcement Learning (RL)** is a branch of **Artificial Intelligence** that focuses on finding an **optimal policy** for an agent. The goal is to enable the agent to take actions in a way that maximizes its **cumulative reward**, allowing it to perform a given task as efficiently as possible.

In previous exercises, you were introduced to the fundamental concepts of RL, explored various environments, and implemented **value-based methods**. In this assignment, we shift our focus to **policy-based methods**, providing an introductory exploration of their principles and applications.

One of the foundational approaches in this category is **policy gradient methods**, with **REINFORCE** being one of the simplest and earliest algorithms. We begin by comparing **policy search** using evolutionary optimization techniques, such as the **genetic algorithm (GA)**, with the **REINFORCE algorithm**. Then, we implement different variants of REINFORCE that enhance its performance and compare them to the standard version. Additionally, we examine how to adapt REINFORCE for continuous action spaces.

Finally, we compare policy gradient methods (REINFORCE algorithm) with DeepQ-Network (DQN), analyzing their strengths and weaknesses to better understand when and why each method is preferred.

The goal of this assignment is to explore **policy-based reinforcement learning methods**, with a focus on **policy gradient algorithms**. You will:

- **Compare policy search methods** – Implement and compare REINFORCE with genetic algorithm-based policy search to understand different optimization approaches.
- **Improve REINFORCE** – Implement and analyze variants of REINFORCE, such as REINFORCE with baseline, to see how they enhance learning stability and efficiency.
- **Apply REINFORCE to continuous action spaces** – Modify the algorithm to work in environments with continuous actions, highlighting key differences from discrete action spaces.
- **Compare Policy Gradient (REINFORCE) vs. DeepQ-Network (DQN)** – Evaluate the strengths and weaknesses of policy gradient methods in contrast to DeepQ-Network.

By completing this assignment, you will gain hands-on experience with **policy gradient techniques**, understand their **advantages and limitations**, and develop insights into how they are applied in different reinforcement learning problems.

Grading

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

Task	Points
Task 1: Policy Search: REINFORCE vs. GA	20
Task 2: REINFORCE: Baseline vs. No Baseline	25
Task 3: REINFORCE in a continuous action space	20
Task 4: Policy Gradient Drawbacks	25
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 1403/12/12 (March 2nd 2025) at 11:59 PM.

Please submit your work by following the instructions below:

- Place your solution alongside the Jupyter notebook(s).
  - Your written solution must be a single PDF file named `HW3_Solution.pdf`.
  - If there is more than one Jupyter notebook, put them in a folder named `Notebooks`.
- Zip all the files together with the following naming format:

  `DRL_HW3_[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 5-point bonus for writing your solution in LaTeX using this template and including your LaTeX source code in your submission, named `HW3_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) 1

1.1 Environment Setup 1

1.2 Submission Requirements 1

2 Part 2 (Problem Descriptions) 3

2.1 Task 1: Policy Search: REINFORCE vs. GA 3

2.1.1 Task Overview 3

2.1.2 Instructions 4

2.1.3 Questions 4

2.2 Task 2: REINFORCE: Baseline vs. No Baseline 4

2.2.1 Task Overview 4

2.2.2 Instructions 5

2.2.3 Questions 5

2.3 Task 3: REINFORCE in a continuous action space 5

2.3.1 Task Overview 6

2.3.2 Instructions 6

2.3.3 Questions 6

2.4 Task 4: Policy Gradient Drawbacks 6

2.4.1 Task Overview 7

2.4.2 Instructions 7

2.4.3 Questions 7

3 References 9

1 Part 1 (Setup Instructions)

Before starting this assignment, ensure that 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:

- `REINFORCE_VS_GA.ipynb`
- `CartPole_REINFORCE_baseline.ipynb`
- `MountainCarContinuous_REINFORCE.ipynb`
- `REINFORCEvsDQN.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. Below is a list of the required libraries:

- `numpy` – For numerical computations and array manipulations.
- `torch`, `torch.nn`, `torch.optim` – PyTorch library for building and training neural networks.
- `torch.distributions.Categorical` and `torch.distributions.Normal` – For handling probability distributions in reinforcement learning.
- `gym` – OpenAI Gym for creating and interacting with reinforcement learning environments.
- `matplotlib`, `matplotlib.pyplot` – For plotting and visualizing learning curves and training results.
- `base64` and `imageio` – For encoding and handling images and videos.
- `IPython` – For displaying animations and interactive outputs in notebooks.
- `logging` and `warnings` – For debugging and handling warning messages.
- `random` – For controlling randomness in experiments.

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

```
pip install numpy torch gym matplotlib imageio ipython
```

Note for Google Colab Users: If you are using Google Colab, all the required libraries are pre-installed. You do not need to install them manually; simply importing them in the notebook is sufficient.

Ensuring that all these dependencies are correctly installed will help you run the notebooks without any issues and focus on experimenting with reinforcement learning algorithms.

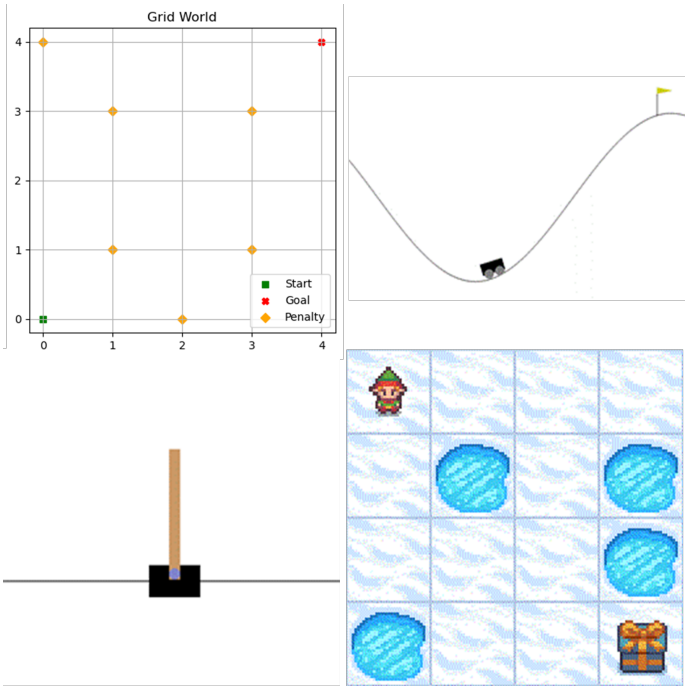
1.2 Submission Requirements

- 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:
    - `REINFORCE_VS_GA.ipynb`
    - `CartPole_REINFORCE_baseline.ipynb`
    - `MountainCarContinuous_REINFORCE.ipynb`
    - `REINFORCEvsDQN.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 consists of various tasks related to policy-based methods. With the provided guidance in each section, you are expected to appropriately complete the assigned tasks. These tasks include optimal policy search, comparison of the REINFORCE algorithm with and without a baseline, implementation of the REINFORCE algorithm in a continuous action space, and comparing REINFORCE vs. DQN. Based on the explanations provided in each section, you are required to complete the specified tasks and provide suitable answers to any conceptual questions that may arise in each section.



2.1 Task 1: Policy Search: REINFORCE vs. GA

In this assignment, you will compare two reinforcement learning approaches—**the REINFORCE algorithm** and **policy search** using **a genetic algorithm**—to solve a simple grid world navigation task.

2.1.1 Task Overview

The environment consists of a grid with a **start point** and a **finish point**, along with various **penalties** scattered across the grid. The goal of the agent is to learn an optimal policy that allows it to navigate from the start to the finish with the **fewest steps** while **avoiding penalties** as much as possible.

You will implement and train both the REINFORCE algorithm (a policy gradient method) and a genetic algorithm (an evolutionary approach) and compare their performance in terms of:

- **Convergence speed** – How quickly the algorithm learns an effective policy.
- **Final policy quality** – How optimal the learned policy is in terms of minimizing steps and penalties.
- **Handling of penalties** – How well the algorithm avoids penalties while finding the shortest path.

- **Stability of learning** – Whether the algorithm converges consistently or shows high variance in performance.
- **Sample efficiency** – How many episodes or iterations are needed to learn a good policy.
- **Exploration vs. exploitation balance** – How each method approaches searching for new paths versus refining known strategies.

By analyzing the results, you will gain insights into the strengths and weaknesses of these two different policy optimization methods.

2.1.2 Instructions

All the necessary explanations and guidance for completing this task are provided in the corresponding notebook. Additionally, the notebook includes a set of conceptual questions that you will find within the file. You are required to provide appropriate answers to these questions.

However, due to space limitations in the notebook, you do not need to write your answers directly in the notebook. Instead, please submit them separately.

2.1.3 Questions

• Based on the implementation and results from comparing policy search using Genetic Algorithm (GA) and the REINFORCE algorithm:

Question 1:

How do these two methods differ in terms of their effectiveness for solving reinforcement learning tasks?

Question 2:

Discuss the key differences in their **performance**, **convergence rates**, and **stability**.

Question 3:

Additionally, explore how each method handles exploration and exploitation, and suggest situations where one might be preferred over the other.

2.2 Task 2: REINFORCE: Baseline vs. No Baseline

In this assignment, you will compare two variations of the REINFORCE algorithm—the vanilla (simple) REINFORCE and REINFORCE with baseline—in solving the classic CartPole environment from OpenAI Gym.

2.2.1 Task Overview

CartPole is a standard reinforcement learning benchmark where an agent must balance a pole on a moving cart by applying forces to the left or right. The goal is to prevent the pole from falling for as long as possible, maximizing the total reward.

You will implement both versions of REINFORCE and analyze their performance differences based on:

- **Convergence speed** – How quickly each method learns an effective policy.
- **Policy stability** – How consistent the learned policy is over multiple runs.

- **Reward accumulation** – The total reward achieved per episode.
- **Variance in updates** – The impact of variance on policy updates.

2.2.2 Instructions

All the necessary explanations and guidance for completing this task are provided in the corresponding notebook. Additionally, the notebook includes a set of conceptual questions that you will find within the file. You are required to provide appropriate answers to these questions.

However, due to space limitations in the notebook, you do not need to write your answers directly in the notebook. Instead, please submit them separately.

2.2.3 Questions

Question 1:

How are the observation and action spaces defined in the CartPole environment?

Question 2:

What is the role of the discount factor (γ) in reinforcement learning, and what happens when γ=0 or γ=1?

Question 3:

Why is a baseline introduced in the REINFORCE algorithm, and how does it contribute to training stability?

Question 4:

What are the primary challenges associated with policy gradient methods like REINFORCE?

Question 5:

Based on the results, how does REINFORCE with a baseline compare to REINFORCE without a baseline in terms of performance?

Question 6:

Explain how variance affects policy gradient methods, particularly in the context of estimating gradients from sampled trajectories.

2.3 Task 3: REINFORCE in a continuous action space

In this homework, you will implement the REINFORCE algorithm in an environment with a continuous action space, such as MountainCarContinuous-v0 from OpenAI Gym, and compare it to its application in discrete action space environments like CartPole. While most of the core concepts remain the same, handling continuous actions introduces key differences that affect learning and performance.

2.3.1 Task Overview

The MountainCarContinuous environment features a car positioned between two hills. The agent must apply continuous forces to push the car up the right hill to reach the goal. Unlike its discrete counterpart (MountainCar-v0), where the agent chooses from a few predefined actions (e.g., push left, push right, or do nothing), in the continuous version, the agent must select an exact force value within a given range.

Your task is to implement the REINFORCE algorithm in this environment and analyze how learning in a continuous action space differs from a discrete one.

2.3.2 Instructions

All the necessary explanations and guidance for completing this task are provided in the corresponding notebook. Additionally, the notebook includes a set of conceptual questions that you will find within the file. You are required to provide appropriate answers to these questions.

However, due to space limitations in the notebook, you do not need to write your answers directly in the notebook. Instead, please submit them separately.

2.3.3 Questions

Question 1:

How are the observation and action spaces defined in the MountainCarContinuous environment?

Question 2:

How could an agent reach the goal in the MountainCarContinuous environment while using the least amount of energy? Explain a scenario describing the agent's behavior during an episode with most optimal policy.

Question 3:

What strategies can be employed to reduce catastrophic forgetting in continuous action space environments like MountainCarContinuous?

(Hint: experience replay or target networks)

2.4 Task 4: Policy Gradient Drawbacks

Two major approaches in RL are value-based methods, which estimate the value of actions, and policy-based methods, which learn a direct mapping from states to actions. In this notebook, we will compare Deep Q-Network (DQN), a popular value-based method, with Policy Gradient (REINFORCE), a policy-based method, on the Frozen Lake environment.

The Frozen Lake environment is a grid-based world where an agent must navigate from a starting position to a goal while avoiding icy holes that result in failure. Since this environment involves stochastic transitions, it provides an interesting challenge for comparing these two learning approaches.

2.4.1 Task Overview

Task Overview

Your objective in this notebook is to experiment with and compare the performance of DQN and Policy Gradient (REINFORCE) in the Frozen Lake environment. The core implementations of both algorithms are provided. Your role is to set the hyperparameters, run the training process, and analyze the results.

Both methods have different learning behaviors:

- DQN approximates the Q-values using a deep neural network and updates its estimates using the Bellman equation.
- Policy Gradient (REINFORCE) directly optimizes the policy using rewards, learning a probabilistic policy without estimating Q-values.

You will tune key hyperparameters such as the learning rate, discount factor, and exploration strategy, then observe their effects on training stability, convergence, and final performance. After training, you will analyze the results using performance metrics and visualizations.

2.4.2 Instructions

This notebook is designed to let you explore the differences between Deep Q-Network (DQN) and Policy Gradient (REINFORCE) by tuning hyperparameters and observing their impact on learning performance. The core implementations of both algorithms are already provided, so your focus will be on adjusting parameters and interpreting the results. To complete this task, first navigate to the Hyperparameter Settings section in the notebook. Modify the provided hyperparameters for both DQN and Policy Gradient according to your preference. You can experiment with different values for parameters such as the learning rate, discount factor (gamma), exploration settings (for DQN), and the number of training episodes. Once you have set the hyperparameters, proceed by running the training cells. This will initiate the learning process for both algorithms, allowing the agents to interact with the Frozen Lake environment and improve their policies over time.

After the training is completed, examine the results using the provided visualizations and performance metrics. The notebook includes plots showing episode rewards, training stability, and convergence speed for both methods. Compare their learning efficiency and final success rates to understand how policy-based and value-based approaches differ in handling this environment.

You are not required to modify the core algorithmic code. Instead, focus on tuning the hyperparameters, observing their effects, and drawing conclusions about the strengths and weaknesses of each learning method in the Frozen Lake environment.

Note that Maybe Policy Gradient is not good for this environment.

2.4.3 Questions

1. **Which algorithm performs better in the Frozen Lake environment? Why?**
   Compare the performance of Deep Q-Network (DQN) and Policy Gradient (REINFORCE) in terms of training stability, convergence speed, and overall success rate. Based on your observations, which algorithm achieves better results in this environment?
2. **What challenges does the Frozen Lake environment introduce for reinforcement learning?**
   Explain the specific difficulties that arise in this environment. How do these challenges affect the

learning process for both DQN and Policy Gradient methods?

3. **For environments with unlimited interactions and low-cost sampling, which algorithm is more suitable?**
   In scenarios where the agent can sample an unlimited number of interactions without computational constraints, which approach—DQN or Policy Gradient—is more advantageous? Consider factors such as sample efficiency, function approximation, and stability of learning.

3 References

[1] Cover image designed by freepik

[2] Policy Search

[3] CartPole environment from OpenAI Gym

[4] Mountain Car Continuous environment from OpenAI Gym

[5] FrozenLake environment from OpenAI Gym