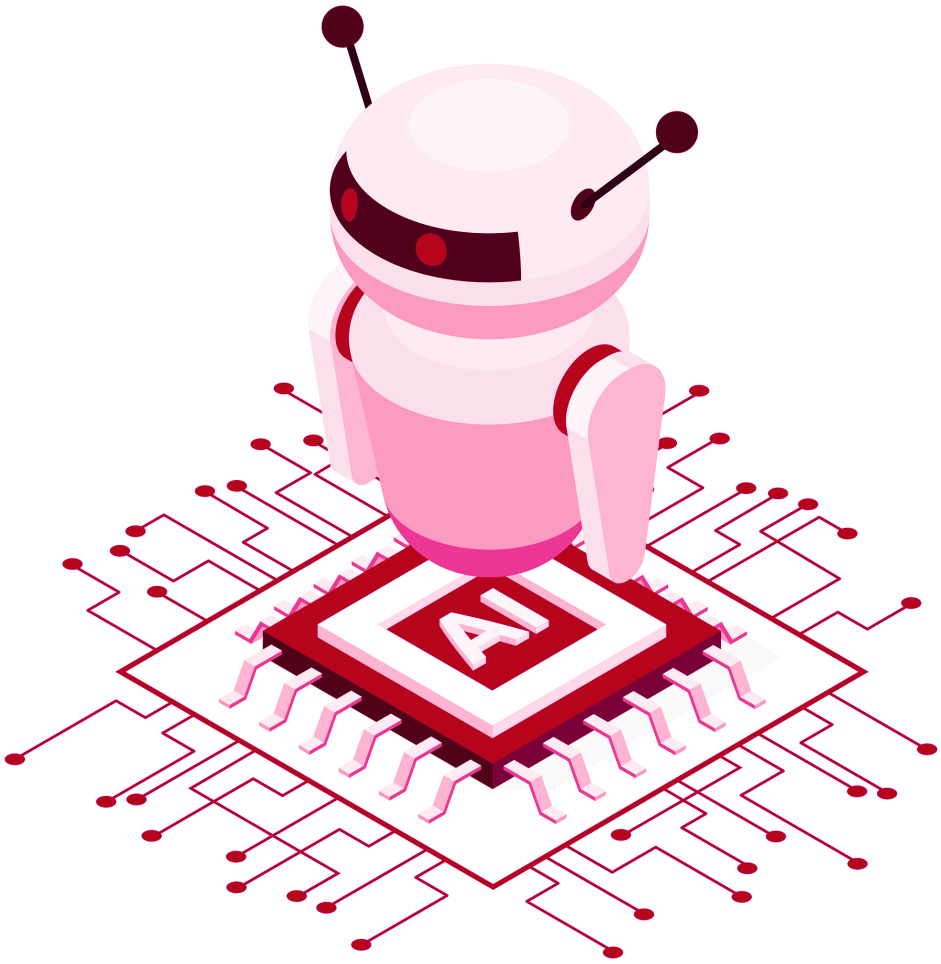

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 3:

Policy-Based Methods

By:

[Full Name]

[Student Number]

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: 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

1 Task 1: Policy Search: REINFORCE vs. GA [20]

1.1 Question 1:

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

Reinforce uses a neural network to represent policy, and then train it using gradient. Its better when we have a continues observation space. While Genetic Algorithm generates a population of possible policies, and evolves the best ones. It is more useful for complex, non smooth or discrete tasks. However GA has higher computation cost (both memory and convergence time)

1.2 Question 2:

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

Reinforce performs better when the policy space is differentiable and smooth (like CartPole), but it might face high variance in gradient estimates, leading to slow convergence. Genetic Algorithms usually have even slower convergence due to their nature and high computation cost, but they perform better in non smooth environments. Reinforce might get stuck in local optima, where GA escapes them. Reinforce is less stable due to variance issues, while GA is more stable because it evolves multiple policies at each step, rather than a single policy. Although in continuous spaces Genetic Algorithm might not even work at all.

1.3 Question 3:

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

In Reinforce, we use choose action using epsilon greedy technique, and adjust the epsilon to balance between exploration and exploitation. In Genetic, the crossover and mutation steps make the solutions more diverse and lead to exploration. As you can see in figures 1&2, Genetic Algorithm finds more diverse policies comparing to Reinforce which finds a simple straight path. Reinforce handles exploitation by simply decreasing epsilon and picking best action. Genetic handles exploitation using the fitness functions, and better policies are more likely to evolve. Reinforce might be preferred in environments where the policy space is differentiable, and quick convergence is desired. Genetic algorithms are better for environments with complex, non differentiable policies or when a more thorough exploration of the policy space is needed.

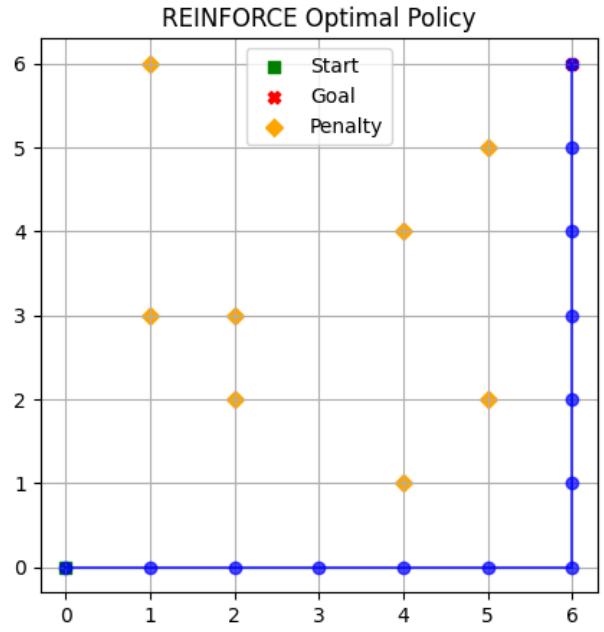

Figure 1: Reinforce optimal policy

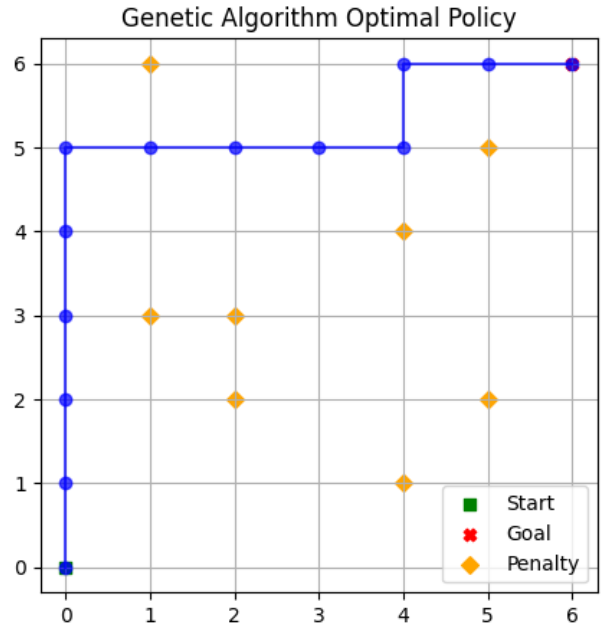

Figure 2: Genetic optimal policy

2 Task 2: REINFORCE: Baseline vs. No Baseline [25]

2.1 Question 1:

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

The observation space in the CartPole is a continuous space with 4 dimensions: cart position, cart velocity, pole angle, and pole angular velocity. The action space is discrete with 2 possible actions: moving the cart left (0) or right (1). So the observation space is a Box(4) and the action space is a Discrete(2).

2.2 Question 2:

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

The discount factor determines weight of future rewards. If γ=0, only current rewards are considered and fututre rewards are ignored. If γ=1, all future rewards have equal importance, and they are not discounted. The higher the γ, the more we care about long term rewards but the variance also increases.

2.3 Question 3:

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

A baseline (often the value function) is introduced to reduce the variance of the gradient estimates.

Subtracting the baseline from the return does not change the expected value of the gradient but reduces variance, and hence stabilizes training due to less noise in updates.

2.4 Question 4:

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

The main challenge is high variance in gradient estimates (which we aim to reduce with some techniques) which makes the convergence slow and unstable. Additionally, they struggle in environments with sparse rewards.

2.5 Question 5:

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

Reinforce with a baseline reaches the optimal policy faster, because we reduced the variance of updates. You can see in figure 3, how using a baseline results to faster convergence.

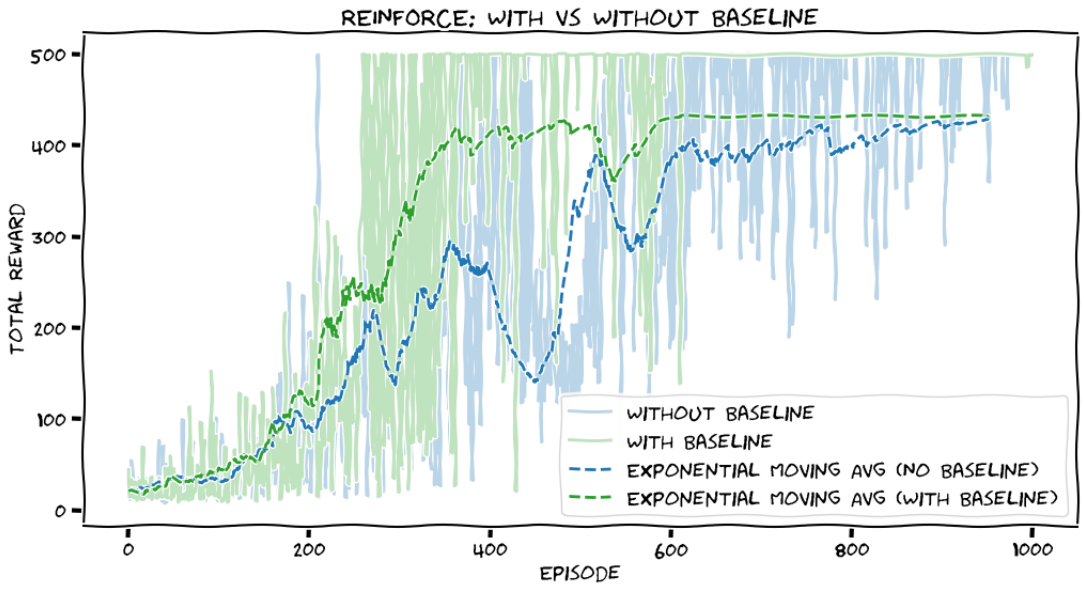

Figure 3: Reinforce total rewards in the Cartpole environment, with vs. without a baseline

2.6 Question 6:

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

Policy gradient methods rely on sampled trajectories to estimate the gradient. Due to the stochastic nature of these trajectories, the gradient estimates can have high variance, leading to unstable and noisy updates and slow convergence.
We use causality trick and baseline network to reduce the variance of gradient estimates, without introducing bias.

3 Task 3: REINFORCE in a continuous action space [20]

3.1 Question 1:

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

The observation space is a continuous space with 2 dimensions: the car's position and velocity. The action space is also continuous, a single value between [1, 1], which determines the force applied to push the car left or right.

3.2 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.

To reach the goal with minimal energy, the agent should leverage momentum by oscillating between the valley walls before making the final push. Instead of applying maximum force immediately, an optimal policy allows the car to build up speed gradually by moving back and forth, using gravity to assist in reaching the peak with the least energy expenditure [6] (you can see this in the generated video in the notebook).

3.3 Question 3:

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

(Hint: experience replay or target networks)

To reduce catastrophic forgetting, we can use techniques like storing past experiences and replaying them during training (experience replay). Another way is to use a separate network to guide learning, making updates more stable (target networks). Also using other learning structures, like actor critic models, improves stability in continuous action environments.

4 Task 4: Policy Gradient Drawbacks [25]

4.1 Question 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?

DQN performs better in the Frozen Lake environment. This is because DQN, learns the optimal Q-values even in sparse reward environments. Policy based methods such as Reinforce, struggles due to high variance and the lack of frequent rewards. DQN converges faster and more reliably (after carefully tuning hyperparameters), while REINFORCE often fails to learn a successful policy, because even when it reaches the goal once or twice, it still does not learn what middle steps are better to take, in other states.

4.2 Question 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?

Frozen Lake introduces sparse rewards, making it difficult for algorithms to learn actions with long-term rewards. DQN is more sample efficient (replay buffer helps with this!) and after reaching the goal a couple of times, it learns what state action pairs are more valuable. Reinforce relies on a trajectory to update the policy just a little bit, and even those updates have high variance. So it cant learn effectively at all.

4.3 Question 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.

For environments with unlimited interactions, policy gradient methods can be useful, especially in continuous action spaces or high-dimensional environments where Q-learning struggles. However, in environments with sparse rewards (like Frozen Lake) DQN is more sample-efficient and stable, even with unlimited interactions. Policy gradient methods require variance reduction techniques and reward engineering to become competitive in such cases.

References

[1] Cover image designed by freepik. Available: https://www.freepik.com/free-vector/cute-artificial-intelligence-robot-isometric-icon_16717130.htm

[2] Policy Search. Available: https://amfarahmand.github.io/IntroRL/lectures/lec06.pdf

[3] CartPole environment from OpenAI Gym. Available: https://www.gymlibrary.dev/environments/classic_control/cart_pole/

[4] Mountain Car Continuous environment from OpenAI Gym. Available: https://www.gymlibrary.dev/environments/classic_control/cart_pole/

[5] FrozenLake environment from OpenAI Gym. Available: https://www.gymlibrary.dev/environments/toy_text/frozen_lake/

[6] This paragraph is generated by ChatGPT.