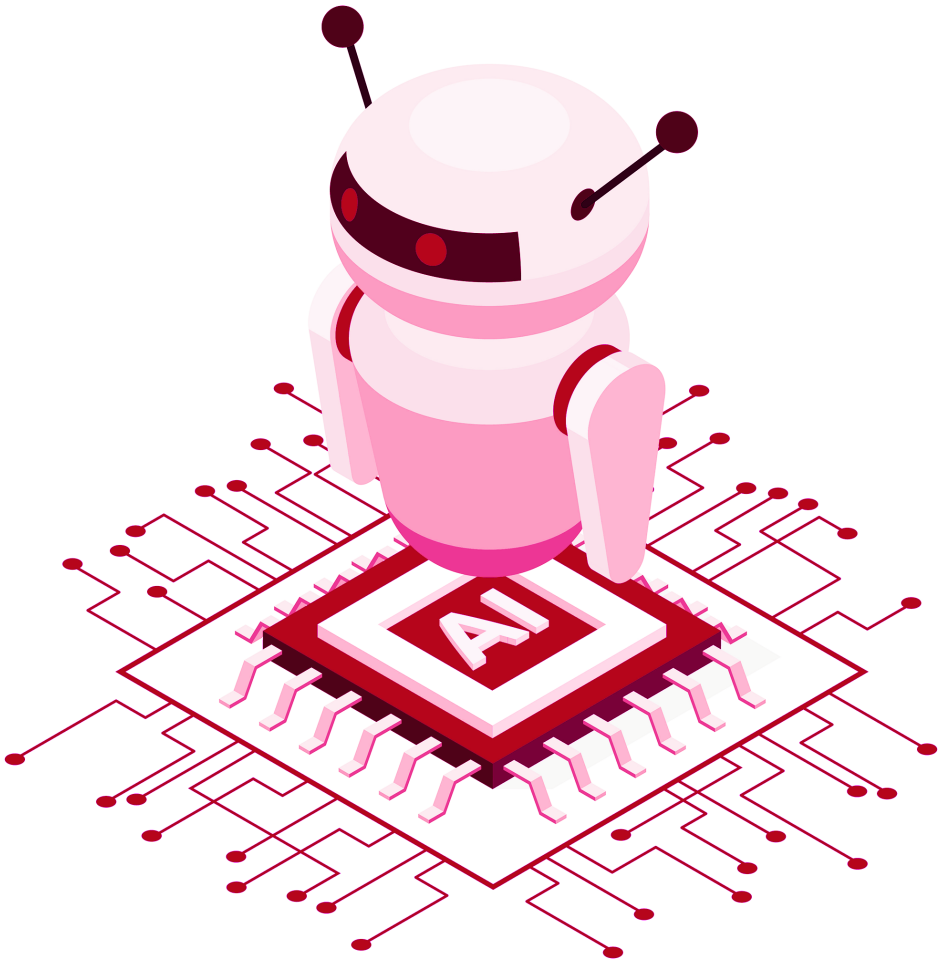


Deep Reinforcement Learning

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?

I ran the REINFORCE algorithm with different hyperparameters (0.008 and 0.002 LR, 0.999 ep decay, and 0.9995 ep decay) and regardless of the faster convergence with better hyperparameters, the final performance was still lower than GA. Also, the paths taken by the REINFORCE agents still cross the points with negative penalty and it wasn't able to find the best policy even after all the 6000 episodes and enough exploration. In a simple discrete environment like this grid world, the REINFORCE algorithms seems like an overkill, and the GA found the optimal policy in a more efficient way and had a better final reward.

1.2 Question 2:

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

In this specific task, the effectiveness of GA was better as it converged way faster (in 50 generations) while REINFORCE took 1800 up to 5000 episodes depending on the hyperparameters until convergence. Also since the REINFORCE algorithm does backpropagation on a deep network, it has a higher computation cost. The performance of GA was better since it didn't do backpropagation and used simple parameter swap operations for convergence, unlike REINFORCE. Also it converged after 50 generations which is faster than the many episodes taken for the REINFORCE convergence. The REINFORCE algorithm depends on the parameters for the convergence and a lower LR and higher epsilon caused the convergence to happen way later. Also the REINFORCE algorithm seems to have less stable rewards and a small misdirection of the trajectory due to random actions causes the algorithms to do really bad (-3000 reward).

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, there is no implicit exploration so we use epsilon-greedy with epsilon decay to act randomly specially in the beginning of the training. In the GA, the algorithm itself has an implicit exploration by doing mutation on the created children in the new generations. It explores the model space by setting solely random parameters to a layer of the neural network and it works well in many of the situations. The REINFORCE algorithm works better when a good epsilon is set for it in continuous environments and it can fine-tune the policy by small gradient changes and have good general results for infinite states of continuous spaces. The GA is better for the discrete environments because of it's simple non-differentiable nature.

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? In the CartPole environment, the observation space is made up of four continuous values: the cart's position, its velocity, the pole's angle, and the pole's angular velocity. On the other hand, the action space is discrete, just two possible moves of pushing the cart either left or right.

`Observation Space: Box([-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38], [4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38], (4,), float32) Action Space: 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?

In reinforcement learning, the discount factor γ determines how much future rewards matter compared to immediate ones. A γ of 0 makes the agent focus only on immediate rewards, while a γ of 1 makes it value future rewards just as much. In Cartpole-v1, a higher γ encourages the agent to plan for keeping the pole balanced over the long run rather than just looking at the next reward.

2.3 Question 3:

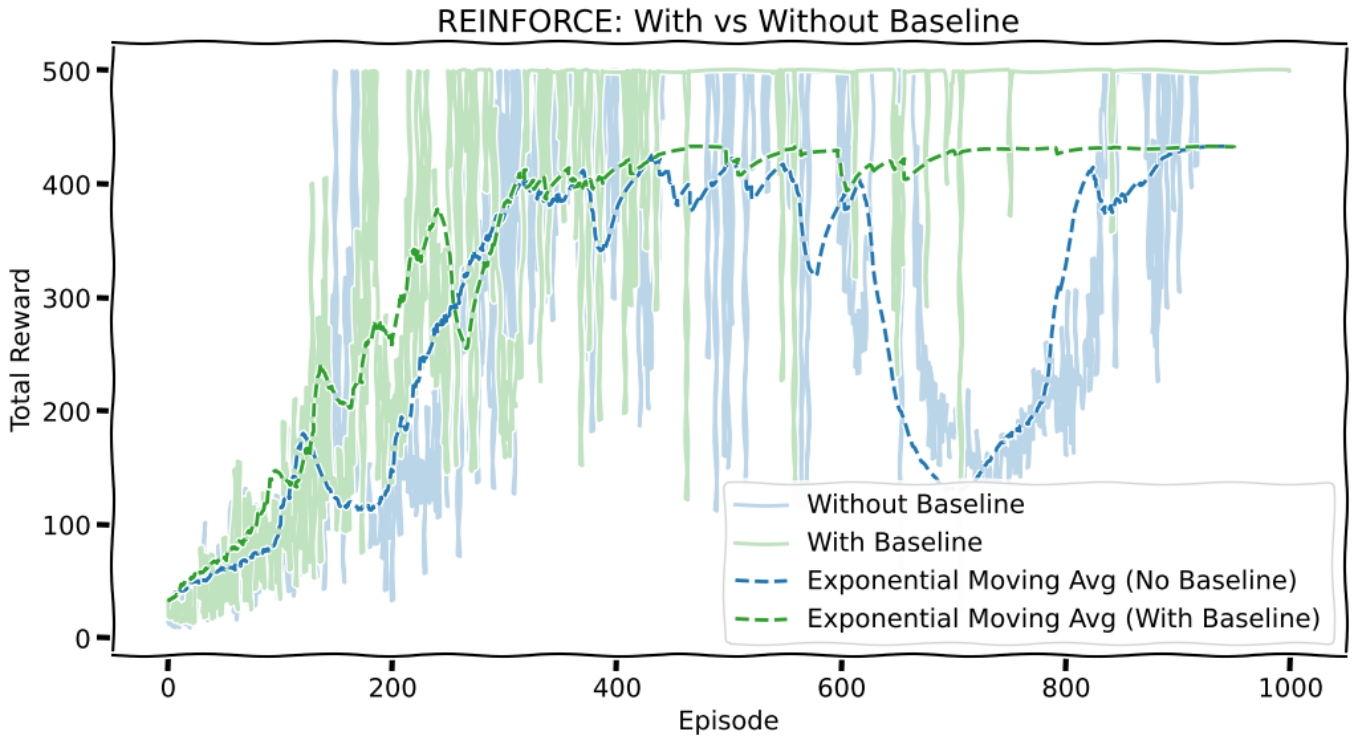
Why is a baseline introduced in the REINFORCE algorithm, and how does it contribute to training stability?

The reason was that without a baseline, all the rewards would be positive, and therefore, we were increasing the chance of all the actions taken in the trajectory, just with different weights. This would increase the variance of the training phase because any trajectories taken would increase that trajectory's probability. The baseline reduces the variance and also fixes the issue because some of the action probabilities with lower returns than the baseline get a reduced probability.

2.4 Question 4:

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

Policy gradient methods like REINFORCE face a lot of challenges due to their reliance on Monte Carlo sampling rather than bootstrapping with a value function. Without a value function, these methods must compute returns over complete episodes, which gives high variance into the gradient estimates. This high variance makes learning unstable and can slow down convergence significantly. Additionally, the sample inefficiency of relying solely on Monte Carlo methods means that a large number of episodes are required to produce reliable updates. Careful tuning of hyperparameters, such as the learning rate, is critical; even small bad step directions can lead to divergent or suboptimal policies.



2.5 Question 5:

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

The REINFORCE algorithm with baseline had a faster rise in reward than the one without baseline as we can see. As expected, the REINFORCE agent with baseline is more stable and didn't suffer from a sudden decrease in rewards; because the value network learned a good approximation of the return so if a bad trajectory or bad weight update happen, the baseline would stop the variance and the miss-step towards bad weights/trajectories again.

2.6 Question 6:

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

The high variance of samples caused by MC sampling causes a lot of model-fitting problems. When estimating gradients from sampled trajectories, each trajectory provides a noisy estimate of the return. Since these returns are based on complete episodes and subject to the inherent randomness of both the environment and the policy's action selections, the resulting gradient estimates can vary significantly from one sample to another. This high variance means that the updates to the policy parameters may not consistently point in the direction of true improvement, making the learning process unstable and often requiring smaller learning rates or more samples to achieve convergence.

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?

They are defined as:

`Observation Space: Box([-1.2 -0.07], [0.6 0.07], (2,), float32) Action Space: Box(-1.0, 1.0, (1,), float32)`

The state space is a 2D vector [position, velocity], where position ranges from -1.2 to 0.6 and velocity from -0.07 to 0.07. The action space is a single continuous value from -1.0 to 1.0, meaning the agent can push left (negative) or right (positive) with varying force.

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.

We need to build the most energy by never forcing in the opposite direction of the current moving velocity. When on the right hill and going left, we should force left to get maximum summed energy and when on the left and going right, go right to get the max velocity and get to the goal.

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)

Experience replay ensures that the learning process isn't overly biased toward recent data and helps preserve older, valuable experiences by reusing older data. Also, target networks are used to stabilize training by maintaining a separate network that is updated more slowly, preventing sudden changes that might overwrite previously learned policies. Together, these approaches help the agent balance learning new information while retaining past knowledge.

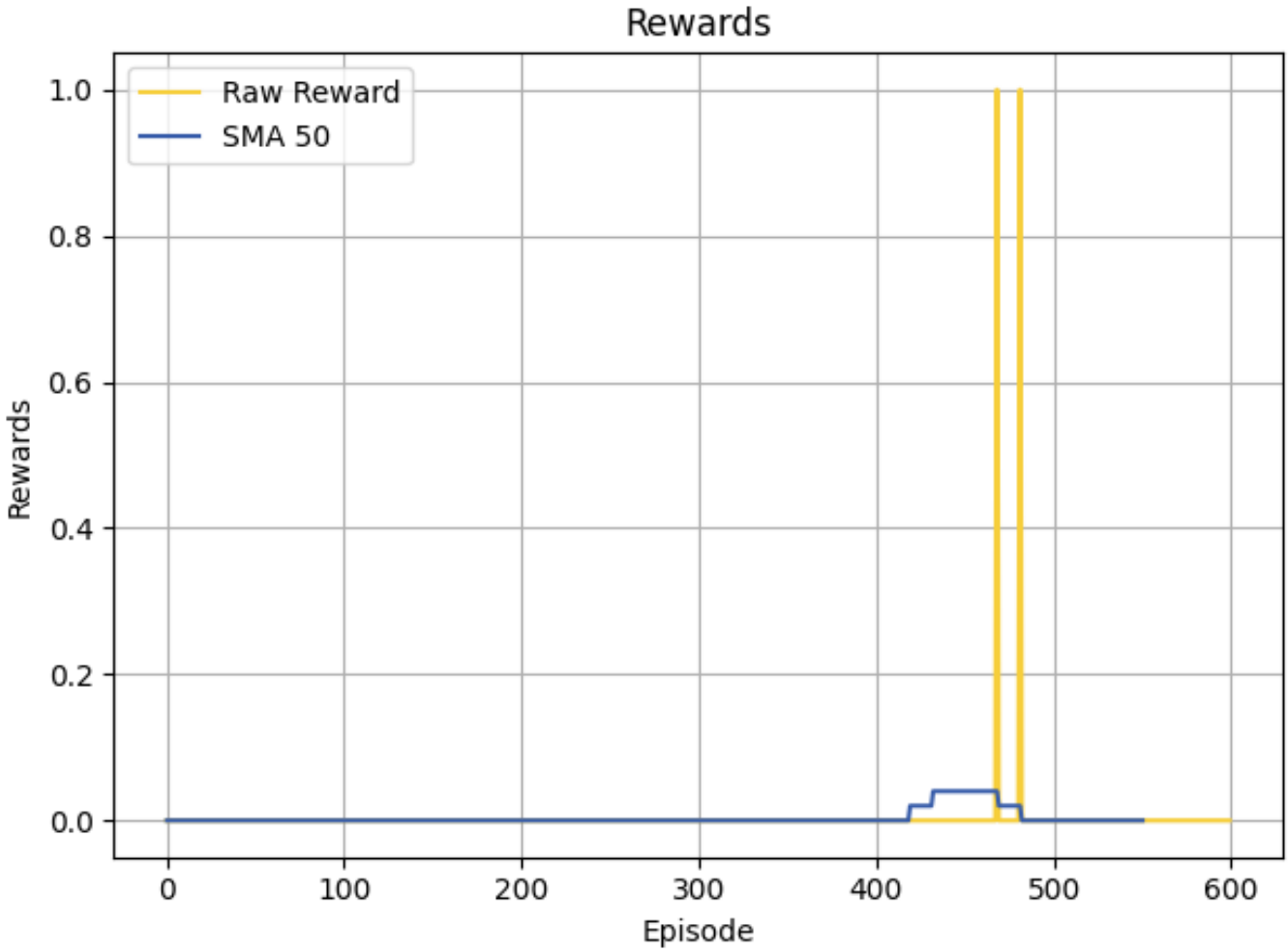


Figure 1: DQN

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?

From the reward plots, DQN clearly does better than REINFORCE in the Frozen Lake environment. DQN takes a long time to learn (around 480–500 episodes). The learning isn't very stable since the reward stays near zero for a long time before suddenly spiking, but at least it shows real progress. On the other hand, REINFORCE stays stuck at zero the entire time, with no meaningful improvement. It just doesn't create any non-zero reward trajectory because of the reward sparsity.

The main reason DQN works better here is that Frozen Lake is a small, discrete problem where value-based methods like DQN usually perform well. REINFORCE, as a policy gradient method, tends to struggle when rewards are sparse, and it also has high variance in learning, which makes it hard to train properly. It might need better tuning, but as it is, it fails to learn a good policy. So overall, DQN is the better choice

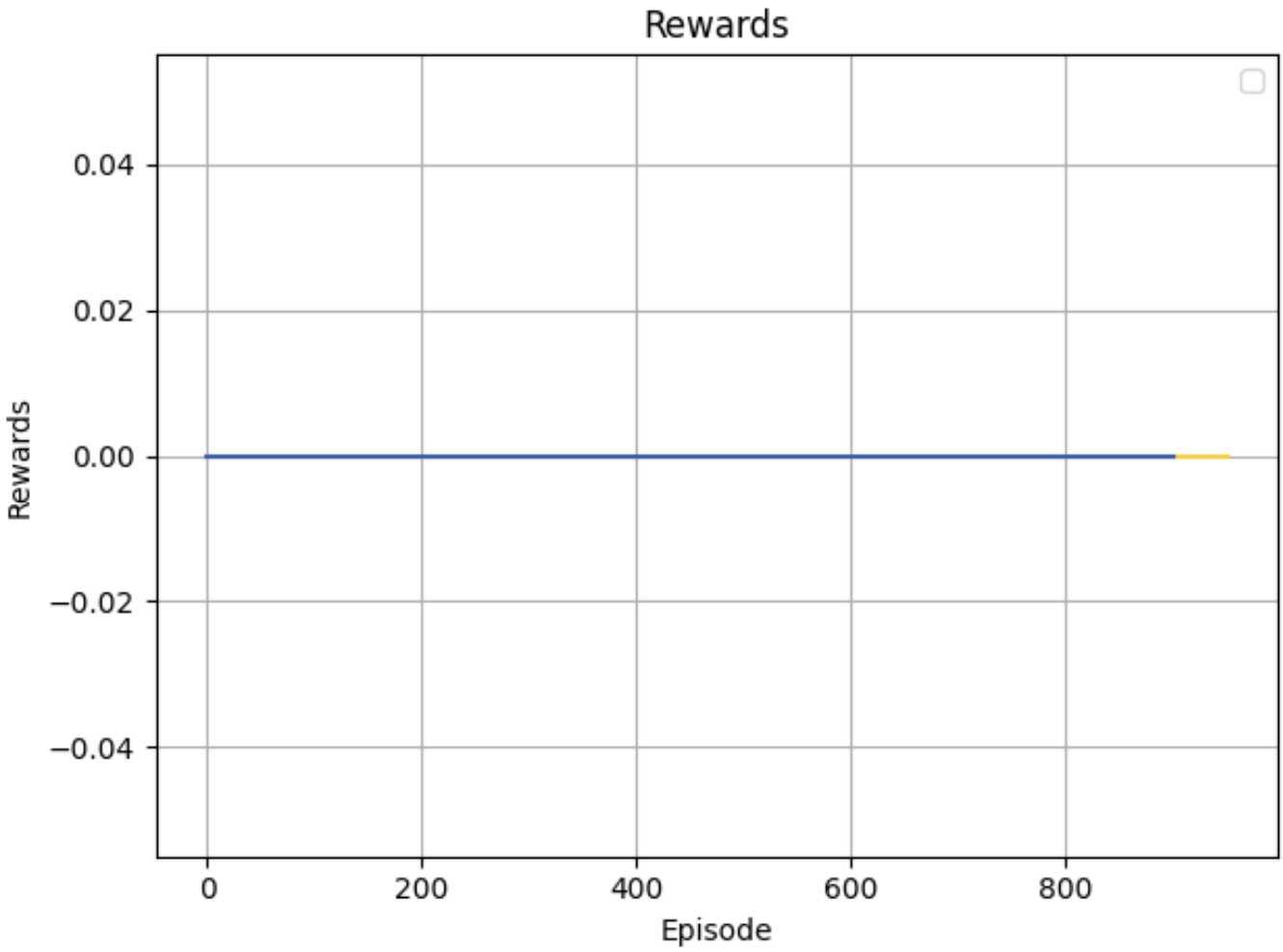


Figure 2: Policy Gradient

for this environment because it at least finds a way to succeed, while REINFORCE doesn't get anywhere.

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?

The challenge of this environment comes from the sparsity of the rewards of the environment. When the reward is hard to obtain, we can't get it in our MC sampling so we just keep getting zero rewards in all the trajectories and have no model updates.

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.

The DQN uses bootstapping to have better use of samples and reduce the variance of the learning, while Policy gradient suffers from high variance of the MC sampling. Now in this situation the downsides of Policy Gradient (on-policyness and MC sampling) fade away because we can get unlimited samples after each policy update and feed the exponential need of data for MC sampling.

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/