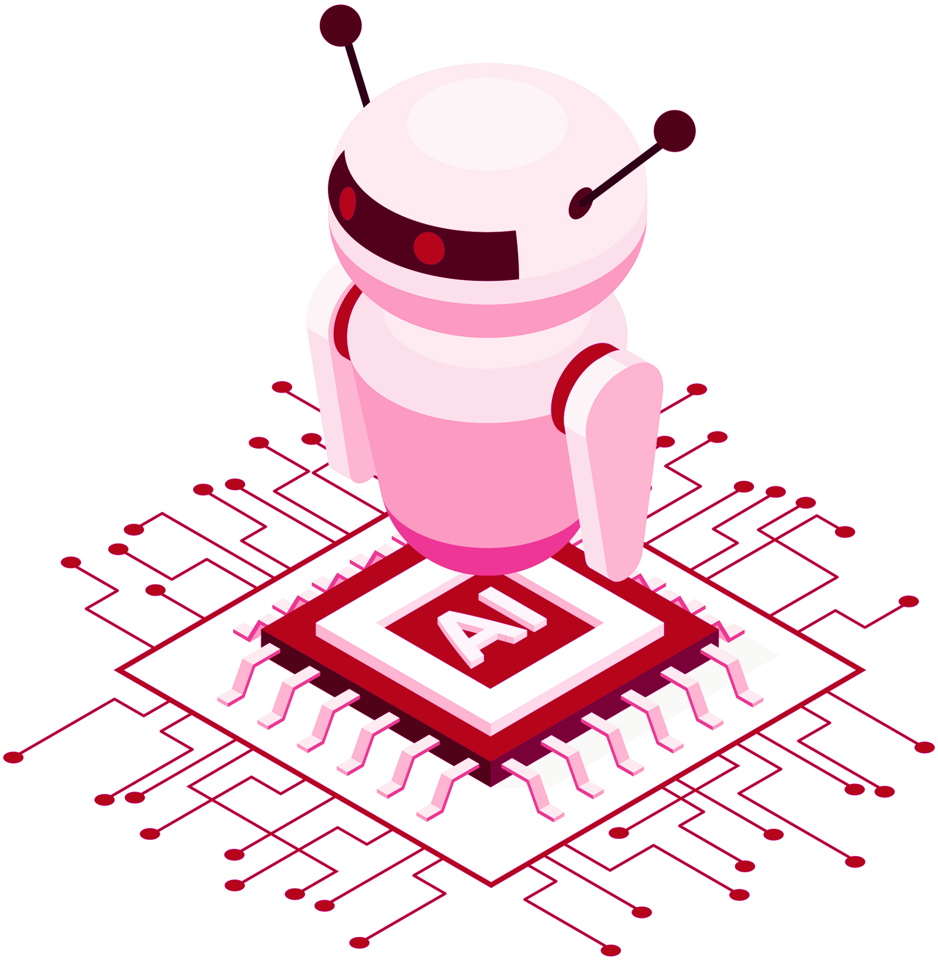

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 3:

Introduction to RL

By:

Parsa Ghezelbash
401110437

Spring 2025

Contents

1 Task 1: Policy Search: REINFORCE vs. GA

Question 1:

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

Answer:

The REINFORCE algorithm is a policy gradient method that directly optimizes the policy by maximizing the expected reward. Genetic Algorithm is optimization techniques inspired by natural selection. They are effective in solving complex optimization problems, including reinforcement learning tasks, especially when the search space is large and complex.

REINFORCE algorithm is sample-efficient but has high variance, on the other hand Genetic algorithm is less sample-efficient due to population evaluations.

Question 2:

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

Answer:

REINFORCE achieves faster initial progress in dense-reward settings but may plateau if gradients become noisy or uninformative. GA progresses more slowly but can discover globally optimal policies in complex, non-convex environments.

REINFORCE converges faster when gradients. GA converges more slowly due to population-based stochastic search but avoids gradient pitfalls like vanishing/exploding updates.

REINFORCE is less stable due to high variance in Monte Carlo returns, even with baselines. GA is more stable through population diversity, but mutations can introduce noise.

Question 3:

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

Answer:

REINFORCE uses epsilon-greedy exploration and exploits via policy gradients. Over time, exploration diminishes as decays. GA explores via mutation and crossover, while exploiting by selecting high-fitness parents.

REINFORCE is better for differentiable policies, dense rewards, and limited compute (e.g., grid worlds with clear paths). GA suits non-differentiable systems, sparse/deceptive rewards (e.g., maze navigation with few rewards).

2 Task 2: REINFORCE: Baseline vs. No Baseline

Question 1:

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

Answer:

Observation consists of four scalars horizontal position, velocity, angle and angular velocity of the cart.

Action space is only a binary input in which 0 means move left and 1 means move right.

Question 2:

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

Answer:

The discount factor determines the importance of future rewards. It balances immediate versus long-term rewards.

$\gamma = 0$: The agent optimizes only immediate rewards and ignores future consequences.

$\gamma = 1$: The agent values all future rewards equally, which can lead to infinite expected returns in non-episodic environments.

Question 3:

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

Answer:

By centering returns around the baseline, updates become less noisy, enabling faster and more stable convergence, and baseline does not bias the gradient.

Question 4:

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

Answer:

High Variance, Sample Inefficiency and Hyperparameter Sensitivity are key challenges in these mothods.

Question 5:

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

Answer:

REINFORCE with baseline Achieves lower variance in gradient and convergence faster with higher average rewards but REINFORCE without baseline leads to less stable performance.

In my observation both methods converged but the solution captured by REINFORCE with baseline is more stable. the solution of REINFORCE without baselie is more likely to diverge if the episodes were longer.

Question 6:

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

Answer:

G_t can vary significantly across trajectories due to the compounding effect of rewards over time, and Since the expectation of gradient is estimated using sampled trajectories, high variance can lead to unstable updates and slow convergence.

3 Task 3: REINFORCE in a continuous action space

Question 1:

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

Answer:

Observation space contains position and velocity of the Car. And action space the force applied to the car and ranges in [-1, 1].

$S = \{(x, v)x[1.2, -0.07], v[0.06, 0.07]\}$

$A = \{aa[1.0, 1.0]\}$

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 agents behavior during an episode with most optimal policy.

Answer:

The optimal policy for using least amount of energy is such that the Car pushes to left and then uses gravity to gain momentum and then use a little energy on when it reaches the rightmost point of the valley to reach the goal.

Question 3:

What strategies can be employed to reduce catastrophic forgetting in continuous action space environments like MountainCarContinuous? (Hint: experience replay or target networks)

Answer:

Instead of training on the most recent experiences, an experience replay buffer stores past transitions and samples them randomly for training. This allows the agent to learn from a more diverse set of past experiences.

Using a separate target network prevents rapid updates to the target estimates, which can destabilize learning.

4 Task 4: Policy Gradient Drawbacks

Question 1:

Which algorithm performs better in the Frozen Lake environment? Why?

Answer:

In my experience DQN performed better and REINFORCE didn't even converge with any set of hyperparameters I tesetd. I ran a code and tested a wide range of hyperparams and none of them worked.

Question 2:

What challenges does the Frozen Lake environment introduce for reinforcement learning?

Answer:

The most difficult challenge of this environment was thae sparsity of it's reward function. If the agnet doesn't reach the goal it gets zero reward and in the begining of the learning when the agent know nothing and act randomly, it is not likely for it to reach the goal oftenly and learns nothing.

Question 3:

For environments with unlimited interactions and low-cost sampling, which algorithm is more suitable?

Answer:

DQN is typically more sample efficient, REINFORCE directly parameterizes and optimizes the policy, which can be advantageous especially in continuous or high-dimensional action spaces and DQN is often easier to stabilize with standard techniques like target networks, replay buffers.

In this senario since sample efficiency is not important REINFORCE will be a better option.