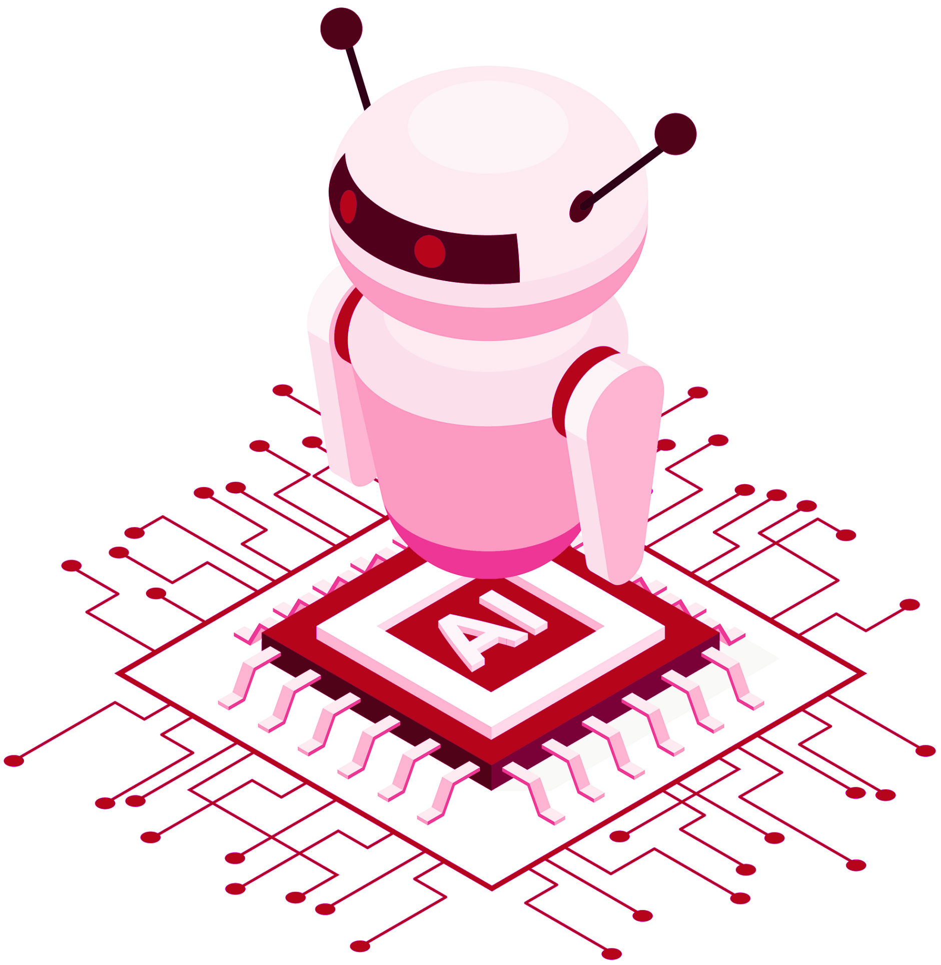

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 3:

Policy-Based Methods

By:

Amir Kooshan Fattah Hesari

401102191

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 vs Genetic Algorithms for RL Tasks

These two algorithms approach reinforcement learning from fundamentally different perspectives, which affects their effectiveness in various scenarios:

REINFORCE Algorithm

REINFORCE is a policy gradient method that directly optimizes a policy through gradient ascent:

- Learning Approach: Uses gradient-based optimization, directly learning from experience through interaction with the environment
- Sample Efficiency: Generally more sample-efficient than genetic algorithms as it leverages the reward signal to precisely adjust the policy parameters
- Credit Assignment: Uses the entire episode return to update policy parameters, though it can suffer from high variance
- Continuous State/Action Spaces: Naturally handles continuous spaces when implemented with neural networks

Genetic algorithms use evolutionary principles to evolve policies:

- Learning Approach: Population-based, evolving multiple policies through selection, crossover, and mutation without explicit gradient information
- Sample Efficiency: Typically less sample-efficient, requiring many evaluations of many policies
- Credit Assignment: Only uses the total return for the entire episode/task, with no intermediate credit assignment
- - Exploration: Often better at global exploration since they maintain a diverse population of policies
-

For complex RL tasks with large state spaces (like modern video games or robotics), REINFORCE and other policy gradient methods typically outperform pure genetic approaches, though hybrid approaches can sometimes offer advantages of both paradigms.

1.2 Question 2:

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

1.2.1 Convergence rate

- Reinforce Algorithm

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t)\ \nabla_\theta J(\theta_t) \quad \mathbb{E}\tau \sim \pi\theta \left[\sum_{t=0}^{T} G_t \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{1}$$

  - Faster initial convergence in well-behaved environments due to directed gradient-based updates
  - Slower convergence in environments with sparse or deceptive rewards
  - Convergence rate of $O(1/\sqrt{T})$ for convex objectives, where T is the number of iterations
- Genetic Algorithm
  - Slower initial convergence compared to gradient methods
  - No theoretical convergence rate guarantees in the general case

1.2.2 Stability

- Reinforce Algorithm

$$\text{Var}[\nabla_\theta J(\theta_t)] = \mathbb{E}[(\nabla_\theta J(\theta_t) - \mathbb{E}[\nabla_\theta J(\theta_t)])^2] \tag{2}$$

  - High variance in gradient estimates, often requiring variance reduction techniques:

$$\nabla_\theta J(\theta_t) = \mathbb{E}\tau \sim \pi\theta \left[\sum_{t=0}^{T} (G_t - b(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{3}$$

    where $b(s_t)$ is a baseline function
  - Sensitive to hyperparameter selection, particularly learning rate α
  - Performance can collapse dramatically during training
- Genetic Algorithm

$$\sigma_P^2 = \frac{1}{n} \sum_{i=1}^{n} (J(\theta_i) - \bar{J})^2 \text{ where } \bar{J} \qquad = \frac{1}{n} \sum_{i=1}^{n} J(\theta_i) \text{ is the mean fitness} \tag{4}$$

  - Population diversity provides inherent robustness against performance collapse
  - Maintains multiple solutions simultaneously, providing robustness
  - Performance tends to plateau rather than collapse

1.2.3 Performance

- Sample inefficiency

$$\text{Sample Efficiency} = \frac{\text{Performance Improvement}}{\text{Number of Environment Interactions}} \tag{5}$$

  - REINFORCE: $O(1/\epsilon^2)$ samples to reach ϵ-optimal policy in well-behaved environments
  - Genetic Algorithms: $O(n/\epsilon^2)$ samples, where n is population size

1.3 Question 3:

Additionally, explore how each method handles exploration and exploitation, and suggest situations where one might be preferred over the other.
Both **REINFORCE** and **genetic algorithms** approach the exploration-exploitation dilemma differently: REINFORCE typically controls exploration through policy entropy or stochastic action selection, with exploration gradually decreasing as the policy converges, whereas genetic algorithms naturally maintain exploration through mutation operators and population diversity, allowing simultaneous exploration of different policy regions. REINFORCE is generally preferred when the reward landscape is relatively smooth, sample efficiency matters, and the policy space is continuous; it excels in complex control tasks with well-defined reward signals and benefits from efficient gradient-based optimization. Genetic algorithms shine in scenarios with deceptive rewards, discrete action spaces, or when the reward function is non-differentiable; they're particularly valuable for problems with multiple viable solutions, sparse rewards, or when the optimal policy has a discontinuous naturesuch as in certain game AI, combinatorial optimization problems, or when the environment dynamics are highly stochastic and unpredictable.

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 is a 2d continous space, whereas the action space is a discrete space with 2 actions.

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?

It is used to give weight to future rewards or as it is said "discount" them , just like the concept immediate return and long time return in economics. The higher the gamma , the farsightedness of the gained reward is more and the less it is the immediate reward has more importance.

2.3 Question 3:

Why is a baseline introduced in the REINFORCE algorithm, and how does it contribute to training stability? A baseline is introduced in REINFORCE to reduce the variance of gradient estimates without changing

their expectation. By subtracting a state-dependent value (typically the estimated state value function) from the returns, the algorithm maintains the same gradient direction in expectation but with significantly reduced variance. This approach keeps the policy updates unbiased while making them more stable and consistent, which accelerates learning and prevents performance collapses that can occur with high-variance updates.

2.4 Question 4:

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

Policy gradient methods like REINFORCE face several key challenges. **First**, they suffer from high variance in gradient estimates, making learning unstable without variance reduction techniques. **Second**, they struggle with credit assignment over long time horizons, where determining which actions contributed to delayed rewards becomes difficult. **Third**, they're prone to converging to local optima due to their gradient-based nature. **Fourth**, they typically require careful hyperparameter tuning, particularly learning rates, to achieve good performance. **Finally**, they can be sample-inefficient compared to value-based methods, often requiring many environment interactions to learn effectively

2.5 Question 5:

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

As it is evident in the image, the baseline method has lower variance

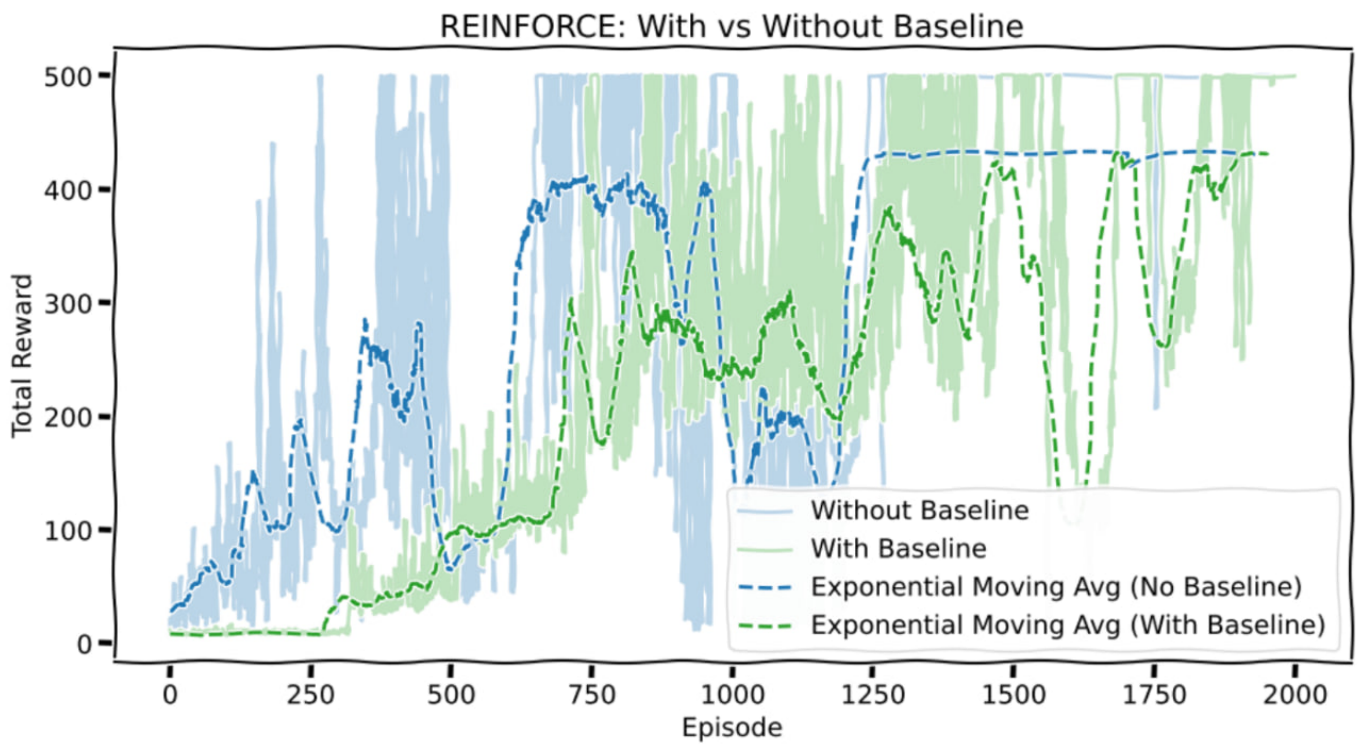

Figure 1: comparison of REINFORCE with baseline and REINFOCE without baseline

2.6 Question 6:

Explain how variance affects policy gradient methods, particularly in the context of estimating gradients from sampled trajectories.
High variance in policy gradient methods occurs because they estimate gradients from a limited number of sampled trajectories, which may not fully represent the true distribution of possible outcomes. This variance is amplified when rewards are sparse, delayed, or have high magnitude differences. As trajectory length increases, variance grows substantially because small differences in early actions can lead to vastly different returns. This excessive variance causes unstable updates where the policy might improve in one update but degrade in the next, resulting in erratic learning, slower convergence, or even failure to learn. Techniques like baselines, advantage functions, and multiple trajectory sampling help mitigate this issue by providing more reliable gradient estimates.

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?

Both are continous spaces.

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 in the MountainCarContinuous environment while using the least amount of energy, an agent should learn to strategically oscillate back and forth between the slopes by applying small bursts of acceleration in both directions, effectively building up momentum to reach the top of the hill with minimal energy expenditure, essentially "swinging" the car to the goal rather than directly powering up the slope.

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)

Catastrophic forgetting is a challenge in continuous learning, especially in environments like Mountain-CarContinuous. Here are key strategies:

- **Experience Replay:** Store and sample past experiences to break correlation and learn from diverse data.
- **Target Networks:** Use separate, less frequently updated networks for stable target value estimation.
- **Elastic Weight Consolidation (EWC):** Protect important weights by penalizing changes crucial for past tasks.
- **Regularization:** Use L1/L2 to prevent overfitting to recent experiences.
- **Reward Normalization:** Stabilize training with consistent reward scales.

These techniques, particularly experience replay and target networks, are crucial for mitigating forgetting in MountainCarContinuous.

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:

- Stable in discrete spaces, less so in continuous.
- Variable convergence speed.
- Complex adaptation for continuous actions.

REINFORCE:

- High variance, unstable training.
- Potentially slow convergence.
- Well-suited for continuous actions.

Conclusion: REINFORCE, and policy gradient methods in general, typically perform better in continuous action spaces due to direct policy optimization.

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 presents:

- **Stochasticity:** Slippery ice introduces randomness, making transitions unpredictable.
- **Sparse Rewards:** Only the goal state yields a positive reward, hindering learning.
- **Long-Term Dependencies:** Optimal paths require multiple steps, demanding credit assignment over extended sequences.

Effects:

- **DQN:** Struggles with stochasticity and sparse rewards, leading to unstable Q-value estimates and slow learning.
- **Policy Gradient:** High variance due to stochasticity and difficulty in credit assignment due to sparse rewards and long term dependencies.

These challenges slow convergence and reduce learning efficiency for both methods.

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.
In environments with unlimited, low-cost interactions:

- **Policy Gradient** methods become highly advantageous.

Reasons:

- **Sample Efficiency Irrelevant:** With unlimited samples, low sample efficiency of policy gradients is no longer a major concern.
- **Direct Policy Optimization:** Policy gradients directly optimize the policy, potentially leading to better convergence in complex environments, given enough samples.
- **Function Approximation Flexibility:** While both methods use function approximation, policy gradients often demonstrate more stable behavior with complex function approximators when provided with abundant data.

Conclusion: When sampling is effectively free, the direct policy optimization of Policy Gradient methods offers a significant advantage.

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/