COMP-579: Reinforcement Learning - Assignment 3

Posted: Friday, February 13, 2026

Due: Friday, February 27, 2026

The assignment can be undertaken individually or in teams of two. Alongside this document, you have been provided with a PLOTTING.PY file. Your task is to create a .IPYNB file, displaying results, and providing explanations for questions. **For the theory part, all written answers must be typed (computer-written); handwritten submissions will not be accepted.**

Following the notation and interfaces of the predefined classes and functions is mandatory; however, you may adapt variable names and intermediate notations in your own code to better align with your thought process and coding style. For coding questions, ensure that you only submit the completed .IPYNB file for evaluation.

You must submit two separate files: one PDF containing all theory answers, and one completed .ipynb file containing all code, results, and code-related explanations. **You must also include an author contribution statement that explains the contributions of the various team members in detail. Additionally, you are required to include a detailed statement on LLM usage for every aspect of usage in the assignment.**

If you have any questions or require clarification, feel free to reach out for assistance on the Ed. We will try to answer your questions within 24 hours. Good luck with your assignment!

Theoretical Questions [40 points]

1. Maximization Bias in Q-Learning. Given an MDP with a state space $\mathcal{S}$ and an action space $\mathcal{A}$. Assume the learned action-value estimate follows a simple additive noise model:

$$Q(s,a) = Q^*(s,a) + \epsilon_{s,a}, \quad \text{where } \epsilon_{s,a} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1).$$

Here, $Q^*(s,a)$ denotes the true optimal action-value function, and the noise terms $\{\epsilon_{s,a}\}$ represent independent estimation errors added to each state-action value.

(a) **[6 points]** Show that even if each $Q(s,a)$ is an unbiased estimator of $Q^*(s,a)$, the maximized estimate is biased upward, i.e.,

$$\forall s, \quad \mathbb{E}\left[\max_{a\in\mathcal{A}} Q(s,a)\right] \geq \max_{a\in\mathcal{A}} Q^*(s,a).$$

(b) **[5 points]** Recall that Q-learning uses the bootstrapped target

$$Y^{QL} = r + \gamma \max_{a'} Q(s',a').$$

Using part (a), explain why Y^{QL} introduces overestimation bias. Consider repeated updates where each new target uses the previously learned Q-values. Explain how the bias evolves through bootstrapping over time.

(c) **[5 points]** The Expected SARSA update uses the target

$$Y^{ES} = r + \gamma \sum_{a'} \pi(a'|s') Q(s', a').$$

Show that if each $Q(s', a')$ is unbiased, then Y^{ES} is also unbiased.

(d) **[8 points]** *Double learning.* Now, consider another estimator,

$$Q'(s, a) = Q^*(s, a) + \epsilon'_{s,a}, \quad \text{where } \epsilon'_{s,a} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1).$$

Double Q-learning uses the target

$$Y^{DQ} = r + \gamma Q'\left(s', \arg\max_{a'} Q(s', a')\right).$$

Assume the independence between ϵ and ϵ'. Compare

$$\mathbb{E}\left[Q'\left(s', \arg\max_{a'} Q(s', a')\right)\right] \quad \text{and} \quad \mathbb{E}\left[\max_{a'} Q(s', a')\right],$$

and explain why double learning alleviates the maximization bias in standard Q-learning.

Hint: You can let $a^* \in \arg\max_{a'} Q(s', a')$. Note that a^* is also a random variable!

2. Policy Improvement Theorem and Policy Gradient.

(a) **[8 points]** Let π be a policy with action-value function $q_\pi(s, a)$ and state-value function $v_\pi(s)$. Define the greedy (argmax) policy with respect to q_π as:

$$\pi_{\text{greedy}}(a|s) = \begin{cases} 1, & a \in \arg\max_{a'} q_\pi(s, a'), \\ 0, & \text{otherwise.} \end{cases}$$

Consider the stochastic mixture policy

$$\pi'(a|s) = (1 - \alpha)\, \pi(a|s) + \alpha\, \pi_{\text{greedy}}(a|s),$$

where $\alpha \in (0, 1]$ is a scalar constant. Prove that the mixture policy satisfies the policy improvement condition

$$\sum_a \pi'(a|s) q_\pi(s, a) \geq v_\pi(s).$$

Conclude that π' is an improved policy.

(b) **[8 points]** The policy gradient theorem states

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s, a)].$$

Consider learning the critic by minimizing the objective

$$L(\phi) = \mathbb{E}_{\pi_\theta}\left[\left(Q_\phi(s, a) - Q^{\pi_\theta}(s, a)\right)^2\right],$$

where $Q_\phi(s, a) \approx Q^{\pi_\theta}(s, a)$. This means

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q_\phi(s, a)].$$

(i) **[2 points]** Derive the gradient $\nabla_\phi L(\phi)$.

(ii) **[6 points]** Then, let $\phi_* = \arg\min_\phi L(\phi)$. Assume

$$\nabla_\phi Q_\phi(s, a) = \nabla_\theta \log \pi_\theta(a \mid s).$$

Show that under the given conditions, although Q_{ϕ_*} need not equal Q^{π_θ} pointwise, the policy gradient using Q_{ϕ_*} is exact,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q_{\phi_*}(s, a)].$$

Coding Questions [60 points]

3. [30 points] Deep Value-based Methods for Control. You are given the MinAtar domain with the task Breakout-v1. Read the document for a detailed interface. Note that:

- For the questions below, you are free to use automatic differentiation packages, such as PyTorch, TensorFlow, and JAX.
- For neural network function approximators, use an MLP with 2 hidden layers and 128 hidden units, and apply the ReLU activation function. Initialize network parameters uniformly between -0.001 and 0.001. Use Adam optimizer.
- Average 5 seeds for each experiment. To check the correctness of your implementation, consider running your code on few seeds first.
- When comparing algorithms, use identical hyperparameters and exploration strategies across algorithms for fair comparison, where discount factor $\gamma = 0.99$, exploration rate $\epsilon = 0.05$. Apply 1 gradient step per environment step. Use 1000 warm-up steps to collect experience and explore before gradient update.
- Use the plot function we provide. Train 3000 episodes per seed, and report the aggregated (mean over 5 seeds) *discounted* return for each training episode.
- When using a replay buffer, make sure that it is cleared before you run a new seed.

(a) **[6 points]** In reinforcement learning (RL), environments are not merely "scorer" of an algorithm; they provide essential structure and insight for interpreting agent behavior.

Before conducting algorithmic experiments, it is advisable to analyze and visualize the environment itself, as this helps understand why different algorithms exhibit varying performance. Show visualizations (a few images is enough) of the Breakout-v1 environment by executing a small number of random actions, and provide a concise description of what is shown in the resulting outputs. Additionally, print the specifications of the observation space and action space of the environment, which define its input and output interfaces.

(a1) **[Optional, 3 bonus points, you can't have more than 100 points]** Carefully inspect the dynamics of the Breakout-v1 environment and its visualizations. Based on empirical investigation and/or reading the environment implementation, discuss whether there exists a small design issue that could make the observation process slightly non-Markovian when each episode initiates. (Don't worry, the experiment result won't change significantly because of this!)

(b) **[24 points]** *Control with experience replay.* Implement Deep Q-learning under:

(i) **Sequential updates (no replay buffer, use the most recent experience to update).**

(ii) **Experience replay with uniform sampling.**

(iii) **Experience replay with prioritized sampling.**

Use a target network that is updated via hard parameter copying every 1000 environment steps. For the replay buffer, use capacity 50,000 and perform mini-batch TD updates with batch size 64. For prioritized replay, specify the sampling rule based on *TD error* and include *importance-sampling corrections*. For each setting, test under different learning rates $\alpha = \{1e^{-4}, 5e^{-4}, 1e^{-3}\}$, and *plot those three curves within one figure*. Compare performance across the three settings and provide plausible explanations for the observed results. The analysis should consider both environmental characteristics and algorithmic properties.

4. [30 points] Deep Policy-based Methods for Control For these questions, you will use the CartPole-v1 environment from Gymnasium. Note that:

- For the questions below, you are free to use automatic differentiation packages, such as PyTorch, TensorFlow, and JAX.
- For neural network function approximators, use an MLP with 2 hidden layers and 128 hidden units, and use the ReLU activation function. Initialize network parameters uniformly between -0.001 and 0.001. Use Adam optimizer (with Pytorch's default hyperparameters).
- Average 5 seeds for each experiment. To check the correctness of your implementation, consider running on your code on a few seeds first.
- When comparing algorithms, use identical hyperparameters and exploration strategies across algorithms for fair comparison, where discount factor $\gamma = 0.99$. Apply 1 gradient step per environment step.

- Use the plot function we provide. Train 1000 episodes per seed, and report the aggregated (mean over 5 seeds) *discounted* return for each episode.

(a) **[10 points]** Implement the base REINFORCE algorithm, as well as REINFORCE with baseline using neural network function approximators. Use a learned baseline, i.e., a neural network that approximates the state value function $V(s)$. Use a Boltzmann policy with temperature T,

$$\pi(a_i \mid s) = \frac{\exp(z(s, a_i)/T)}{\sum_j \exp(z(s, a_j)/T)},$$

where $z(s, a_i)$ is the output of a neural network (logit).

Initialize the temperature at $T_0 = 10.0$ and decay it exponentially with decay factor 0.99. Enforce a lower bound $T \geq 0.5$ to maintain sufficient exploration. Mathematically,

$$T_{n+1} = \max(0.5, 0.99\, T_n),$$

where n denotes the current episode number. For the value function network, use $\alpha_V = 1e^{-3}$ and $\alpha_\pi = 1e^{-4}$ for the policy network. Compare the two algorithm, and *plot the two curves within one figure*.

(b) **[5 points]** For the base REINFORCE algorithm, test different $T \in \{0.1, 1.0, 10.0\}$ alongside the version with annealing from (a). *Plot those four curves within one figure.* Analyze the result briefly.

(c) **[10 points]** Implement the one-step Actor-Critic algorithm. For the learning rates, use $\alpha_\pi = 1e^{-4}$ for the actor and $\alpha_V = 1e^{-3}$ for the critic. Use the same Boltzmann policy (with annealing) as for the REINFORCE algorithm.

(d) **[5 points]** For the Actor-Critic algorithm, run the same experiment for $\alpha_V \in \{1e^{-4}, 5e^{-4}, 2e^{-3}\}$. *Plot those four curves (including $\alpha_V = 1e^{-3}$) within one figure.* How does the learning rate of the critic affect this algorithm, particularly in relation to the learning rate of the actor? Explain briefly why this might be.