COMP-579: Reinforcement Learning - Assignment 3

Posted Tuesday, February 25, 2025
Due Wednesday, March 19, 2025

1. **Value-based methods with deep neural network [50 points]**

   Implement Q-learning and Exptected SARSA for both Acrobot-v1 [1] and ALE/Assault-ram-v5[2] environments from the Gym suite using the following guidelines:

   - Use a Neural Network approximation for Q, that is, if $\mathbf{x}$ is a vector representing the state and $\mathbf{a}$ is the action vector, use $Q\_value(\mathbf{x}) = MLP(\mathbf{x}; \theta)$, where θ are the parameters of the Value function you need to learn, $Q \in \mathbf{R^m}$ where m denotes the number of discrete actions.
   - Model configuration: Initialize the parameters for the value function uniformly between -0.001 and 0.001, we recommend using either a 2 or 3-layer Neural Network for the Value function, with a hidden dimension of 256.
   - Use an ϵ- greedy policy with three choices of ϵ and step-size parameters $1/4, 1/8, 1/16$. and run 50 learning trials with different initializations for the Value function, each having 1000 episodes, for each configuration. That means 3(ϵ's) * 3 (step-sizes) * 50 runs * 1000 episodes.
   - Repeat the Previous step using a replay buffer (with transitions randomly sampled) and do gradient updates using a mini-batch of transitions. The capacity of the replay buffer is 1M.
   - Plot training curves with the mean across seeds as lines and the standard deviation as a shaded region. (Performance on the Y-axis, and the episode on the X-axis). Generate 18 graphs covering all configurations per environment. Present separate plots for each environment, with distinct graphs for settings with and without a replay buffer. Use green for Q-Learning and red for Expected SARSA, differentiating hyperparameters with different line styles (e.g., solid, dashed).
   - **Implement all the methods using any automatic differentiation package, such as Pytorch**.

2. **Policy Gradient Theorem [20 points]**

   Given an MDP with a state space $\mathcal{S}$, Discrete action space $\mathcal{A} = [a_1, a_2, a_3]$, Reward function $\mathcal{R}$, discount factor γ, and a policy with the follwing functional representation:

$$\pi(a_1|s) = \frac{\exp(z(s, a_1))}{\sum_{a \in \mathcal{A}} \exp(z(s, a))}. \tag{1}$$

[1] `https://gymnasium.farama.org/environments/classic_control/acrobot/`
[2] `https://ale.farama.org/environments/assault/`

Use the policy gradient theorem to show the following:

$$\nabla_z J(\pi) = \frac{\partial J(\pi)}{\partial z(s,a)} = d^\pi(s)\pi(a|s)A^\pi(s,a), \tag{2}$$

where d^π is the steady state distribution of the Markov chain induced by π and $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

3. **Policy-based methods with deep neural network [30 points]**

   Implement REINFORCE and Actor-Critic method for both the Acrobot-v1 and ALE/Assault-ram-v5 environments.

$$\pi(a_i|s) = \frac{\exp(z(s,a_i)/T)}{\sum_{a\in\mathcal{A}} \exp(z(s,a)/T)}. \tag{3}$$

   - Implement a Boltzman's Policy as in eq. (3) and Neural Network approximation for z. That is $z(\mathbf{s}) = MLP(\mathbf{s}, \theta)$, where θ are the parameters of z you need to learn, and $a \in \{1, \ldots, m\}$ is a discrete action. In the case of the Actor-Critic algorithm use a Neural Network approximation for the State-Value function $\hat{V}(s,w)$, where w are the parameters of the State-Value function.
   - Similar to Question-1, use appropriate initialization & model configuration for the policy parameters and state-value parameters.
   - Implement a Boltzman's Policy and run 50 learning trials with different initializations for the model, each having 1000 episodes for the following two configurations. 1. A fixed temperature $T > 0$ (of your choice) and 2. A decreasing temperature T. (50 runs * 1000 episodes * 2 configuration) You are free to choose your step sizes for these implementations.
   - Plot training curves with the mean across seeds as lines and the standard deviation as a shaded region. (Performance on the Y-axis, and the episode on the X-axis). Generate 2 graphs covering all configurations per environment. Use green for REINFORCE and red for Actor-Critic, differentiating hyperparameters with different line styles (e.g., solid, dashed).
   - Similar to value-based methods, you can implement all the methods using any automatic differentiation package, such as Pytorch.

Tips:

- In the *Ale/Assault-ram-v5* the observation space is *Box(0, 255, (128,), uint8)*, it is recommended to normalize observation by dividing it by 255 for better convergence.
- In Question 3, you can use a Linear scheduler for decreasing temperature.
- For plotting the training curve with the mean across seeds as lines and the standard deviation as a shaded region, you can use *matplotlib.pyplot.fill_between*[3]

[3] `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill_between.html`