DS 543 HW2

Due Date: 12:30PM Feb 20th

Instructions

- Write all your code and analysis in a **Google Colab** or **Jupyter Notebook**.
- Save the notebook as a **PDF** and submit the PDF on Gradescope.
- Ensure your notebook is well-organized, with clear headings and markdown cells explaining your work.
- For Problems 2 and 3, plot the **reward vs. episode** graph for all 10,000 episodes during training. Perform analysis and comparison based on the **Area Under the Curve (AUC)**, i.e., the total rewards accumulated over all episodes.

Problem 1: Implementing Naive BlackJack as a Gym Environment (25 points)

Implement the "Naive BlackJack" game in HW1 as a custom Gym environment. Follow the game rules described in HW1. Test your environment by running random actions for 1000 episodes and report the average reward.

Problem 2: Implementing DQN Variants (25 points)

In this problem, you will implement and compare three DQN variants:

1. **Naive DQN** (provided in the last code block in Lecture 6 Colab).
2. **Double DQN**.
3. **Multi-step Return DQN**.

Tasks

1. Implement each algorithm and evalute them in the Naive BlackJack environment from Problem 1.
2. Perform hyperparameter tuning for each variant. Consider the following hyperparameters:
   - network structure (e.g. number of layers and number of neurons per layer)[1].
   - batch size (e.g. 128, 256, 512)
   - Learning rate (e.g., 1e-3, 1e-4, 1e-5).
   - Discount factor (e.g., 0.9, 0.95, 0.99).
   - Number of steps for multi-step returns (e.g., 3, 5, 10).
   - Exploration strategy (e.g., ϵ decay rate).

   Performing hyper-parameter tuning by computing the total rewards across 10,000 episodes and choosing the hyper-parameter spec that maximize the total rewards.
3. Plot the **reward vs. episode** graph for each variants with the best found hyper-parameter setup.
4. Compute the **AUC** (total reward over all episodes) for each variant and compare their performance.

[1] The range provided above only serves as an example. The optimal hyper-parameter could be outside of this range.

Problem 3: Implementing Policy Gradient Methods (25 points)

In this problem, you will implement and compare three policy gradient methods:

1. **REINFORCE**.
2. **Trust Region Policy Optimization (TRPO)**.
3. **Proximal Policy Optimization (PPO)**.

Tasks

1. Implement each algorithm and evalute them in the Naive BlackJack environment from Problem 1.
2. Perform hyperparameter tuning for each algorithm. Consider the following hyperparameters:
   - Learning rate (e.g., 1e-3, 1e-4, 1e-5).
   - batch size (e.g. 128, 256, 512)
   - Discount factor (e.g., 0.9, 0.95, 0.99).
   - Clip parameter (for PPO).
3. Plot the **reward vs. episode** graph for each variants with the best found hyper-parameter setup.
4. Compute the **AUC** (total reward over all episodes) for each variant and compare their performance.

Problem 4: Algorithm Comparison and Analysis (25 points)

In this problem, you will compare the performance of all algorithms implemented in Problems 2 and 3.

Tasks

1. Compare the performance of the DQN variants (Naive DQN, Double DQN, Multi-step Return DQN) and the policy gradient methods (REINFORCE, TRPO, PPO) based on:
   - **AUC** (total reward over all episodes).
   - Stability of training (variance in rewards during training).
2. For each algorithm, discuss their hyper-parameter sensitivity with respect to each applicable hyper-parameter. For example, is DQN particularly sensible to the discounting factor? Is REINFORCE particularly sensitive to the learning rate?
3. Plot the **reward vs. episode** graph for all algorithms on the same plot for easy comparison.
4. Discuss the strengths and weaknesses of each algorithm in the context of the Naive BlackJack environment.