AI3601 Reinforcement Learning – 2025 Fall

Final Project

Due: January 18th, 2026, at 8:00 pm
Assigned: December 12th, 2025

General Policy:

- Please submit on **CANVAS** a **.zip** version of your solutions to this project assignment, which should contain at least the following two items:
  1) A **report** in **.pdf** format (naming: Report+studentID+firstName lastName);
  2) your **source code** to implement the experiment (naming: Code+studentID+firstName lastName);
  3) and a **README file** to show the instructions to run your code (naming: README+studentID+firstName lastName).
- **NO LATE** submission is allowed, otherwise the grade will be discounted.

1 Introduction

The goal of the final project is to implement two kinds of model-free RL methods: value-based RL and policy-based RL. In this project, you are free to choose RL methods to solve two benchmark environments.

2 Review

2.1 Value-Based Reinforcement Learning

Value-based methods strive to fit action value function or state value function, e.g. Monte-Carlo, TD learning for model-free policy evaluation and SARSA, Q-learning for model-free control. Off-policy training mode is easy to implement in value-based method. DQN achieves remarkable performance under off-policy. In DQN (Silver, 2015), past experiences that stored in experience buffer can be used to train the deep Q network. In many transfer algorithms for DQN, expert's experiences are often used to fit the current value function. Hence value-based methods are often more sample efficient.
Although value-based RL like DQN and its variants achieve remarkable performance in some task, e.g. atari games, the inherent drawbacks hinder its development.

- First, action selection in value-based methods is according to the action values, which is inherently unsuited to continuous action space.
- Second, non-linear value function approximation like neural network is unstable and brittle with respect to their hyperparameters.

2.2 Policy-Based Reinforcement Learning

In original policy gradient $\nabla_\theta \log \pi_\theta(s_t, a_t) v_t$, return v_t is the unbiased estimation of expected long-term value $Q^{\pi_\theta}(s, a)$ following a policy $\pi_\theta(s)$ (Actor). However, original policy gradient suffers from high variance. Actor-Critic algorithm uses Q value function $Q_w(s, a)$, named Critic, to estimate $Q^{\pi_\theta}(s, a)$. Though Critic may introduce bias, it can dramatically reduce variance and proper chose of function approximation may avoid it.
The biggest drawback for policy gradient methods is sample inefficiency: since policy gradients are estimated from rollouts. Although actor-critic methods use value approximators (Critic) instead of rollouts, its on-policy style remains sample inefficient. Prior works, such as DDPG (Lillicrap, 2015), Soft Actor-Critic (Haarnoja, 2018) strive to introduce off-policy mode to Actor-Critic.

3 Experiment Environments and Requirements

OpenAI provides benchmark environments toolkit 'gym' to facilitate the development of reinforcement learning. 6 types of experiment environments are

available (access https://www.gymlibrary.dev/ for more). In our project, you are required to train agents over Atari and MuJuCo. You should choose appropriate and effective RL methods to achieve high scores in the environments as possible as you can. To get started with gym, refer to https://github.com/openai/gym.

3.1 Atari Games Environment Description

The Atari 2600 is a home video game console developed in 1977. Dozens of games are provided by 'gym'. In our project, we limit the choice of environment to the following:

- VideoPinball-v5
- Breakout-v5
- Pong-v5
- Boxing-v5

You should at least choose one environment to test your value-based method.

3.2 MuJuCo Continuous Control Environment Description

MuJuCo stands for Multi-Joint dynamics with Contact, which is originally designed for model-based control methods test. Now, MuJoCo simulator is a commonly adopted benchmark for continuous control. We narrow down the choice of environments to the following:

- Hopper-v4
- Humanoid-v4
- HalfCheetah-v4
- Ant-v4

You should at least choose one environment to test your policy-based method.

3.3 Requirements

Here is the experiment content:

- You are required to choose and implement value-based RL algorithm and test it on at least one of the Atari game listed above.
- You are required to choose and implement policy-based RL algorithm (except Soft Actor-Critic) and test it on at least one of the MuJuCo simulator listed above.

The algorithms you choose in the scope of value-based and policy-based are non-limited. For the ease of running your submitted codes and grading, we have some limitations in this project.

- Programming language: python3
- The final results should use the experiment Name like following:

```
python run.py --env_name BreakoutNoFrameskip-v4
```

4 Report and Submission

4.1 About Submission

- You are required to accomplish this project individually.

4.2 About Report

The report should cover but not be limited to the following sections:

- The description of the algorithms you use.
- The performance of the algorithms you achieve in selected environments
- The analysis about the algorithms.

4.3 Bonus

- Modification of the algorithms that achieves better performance.
- Test your algorithms on more than one environment.
- Excellent analysis about the algorithms.
- **It is also highly encouraged (with probably extra bonus points!) to implement these value-based and policy-based RL algorithms that we have learned in class to solve those practical engineering or research problems that you are currently working on. If you decide to do so, please contact with me first to see whether such a proposal can be granted to replace the original project.**