AIL 722: Assignment 3
Semester I, 2024-25

Due 25th November 2024 11:55 PM

Instructions:

- This assignment has three parts.
- You should submit all your code (including any pre-processing scripts you wrote) and any graphs you might plot.
- Include **a write-up (pdf) file**, which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- Due to limited support for Windows/MacOS in some Gymnasium environments, we **recommend using Linux-based machines provided by HPC IIT Delhi** (see App. A) for this assignment.
- Please use Python for implementation using only standard python libraries. Do not use any third-party libraries/implementations for algorithms.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, a penalty of -10 points and possibly much stricter penalties (including a **fail grade** and/or a **DISCO**).
- Starter code is available at this link and a short description is in Section A.8.
- The hyperparameter tuning and experiments may take a significant amount of time. Please start the assignment early.

1 Monte Carlo Sampling Methods [25 points]

In this part of the assignment, you will implement an off-policy Monte Carlo control algorithm with importance sampling as given in Fig 1 to solve environments with unknown transitions, specifically *Taxi-v3* and *FrozenLake-v1* environment. Note that the Monte Carlo method may require a significantly larger number of episodes before convergence. Use the conda environment from previous assignment for this part.

1. **Implementation**: Implement the off-policy Monte Carlo algorithm and apply it to the *Taxi-v3* and *FrozenLake-v1* environment from the gymnasium. Plot the Mean Reward vs Episode curve.
2. **Visualization**: Evaluate the learned policy by visualizing the trajectories and report the mean reward over 100 trajectories.

Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \in \mathbb{R}$ (arbitrarily)
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
$b \leftarrow$ any soft policy
Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
$W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Figure 1: Off Policy Monte Carlo Control

2 Policy Gradient Method [60 points]

In environments with continuous action spaces, learning the Q-function may be difficult. The policy gradient method directly learns the policy without the need to learn the Q-function. In this section, we will implement the Policy Gradients algorithm and assess its performance in the provided domains (Table 1).

Table 1 contains the descriptions of the environments that we will work within this part of the assignment. These environments(except TreasureHunt-v2) belong to MuJoCo (Multi-Joint Dynamics with Contact) RL environments. These environments model robotic control and locomotion, offering high-fidelity simulations for tasks like walking, running, and manipulation, focusing on continuous action spaces

Refer to the A.8 for a description of the starter code. Note that all hyperparameters are to be stored in 'config.py' file for MuJuCO environments. Instructions for setting up the conda environments are available in Section A.5

Environment	**State Space**	**Action Space**	**Description**
TreasureHunt-v2(Large & Varying)	Discrete of the order 10^{10}	4 discrete actions (Move left, right, top and bottom)	The agent navigates the boat through grid world sea, avoiding pirates and collecting treasures to finally reach the fort. The map gets reconfigured at the start of each episode.
InvertedPendulum-v4	Box(-inf, inf, (4,))	Box(-3.0, 3.0, (1,))	A classic control problem where the agent must balance a pole attached to a moving cart by applying forces to the cart. The goal is to prevent the pole from falling over.
Hopper-v4	Box(-inf, inf, (11,))	Box(-1.0, 1.0, (3,))	A biomechanical environment where the agent controls a 2D, one-legged robot, aiming to make it hop forward while maintaining balance. The agent must maximize forward velocity while avoiding falls.
HalfCheetah-v4	Box(-inf, inf, (17,))	Box(-1.0, 1.0, (6,))	A physics-based environment where the agent controls a 2D, 6-legged cheetah robot. The goal is to maximize forward velocity by efficiently coordinating the movements of the legs while maintaining balance.

Table 1: Environments used in Policy Gradient Methods Section

1. **Implementation:** Implement the vanilla REINFORCE algorithm in the *RLAgent* Class of 'MujucoEn-

vs/agents/mujuco_agent.py' file.

2. **InvertedPendulum-v4** Train the policy using the vanilla REINFORCE algorithm. Plot the reward vs episodes.

3. **Hopper-v4** Train the policy using the vanilla REINFORCE algorithm. Plot Reward vs episodes curve. In case of suboptimal performance improve the algorithm in RLAgent class by adding a **average reward baseline** to reduce the variance, and again train the model and compare the reward curves.

4. **TreasureHunt-v2** Implement and train two models: one using the vanilla REINFORCE algorithm and the other using REINFORCE with the average reward baseline. Compare the performance of both models by plotting the total reward per episode versus the number of training steps. Use the following architecture for the policy network:

Layer Type	**Input Shape**	**Output Shape**	**Kernel Size**	**Stride**	**Padding**	**Activation**
Conv2D	(4, H, W)	(64, H, W)	(3, 3)	1	1	ReLU
Conv2D	(64, H, W)	(64, H/2, W/2)	(3, 3)	2	1	ReLU
Conv2D	(64, H/2, W/2)	(64, H/4, W/4)	(3, 3)	2	1	ReLU
Flatten	(64, H/4, W/4)	(64 * 9)	-	-	-	-
Fully Connected	(64 * 9)	(64)	-	-	-	ReLU
Fully Connected	(64)	(4)	-	-	-	Softmax

Table 2: TreasureHunt-v2 Policy Network

5. **Half-Cheetah-v4** : Implement a **critic-network** and train the agent on Half-Cheetah-v4 environment. Use *ActorCriticAgent* class in 'MujucoEnvs/agents/mujuco_agents.py"

6. **Analysis and Visualisation:** Calculate the maximum reward (averaged over last 100 episodes) reached for all four environments. Save a video of 10 rollouts on all four environments with the best-trained models.

3 Stable Baselines3[15 Points]

In this part of the assignment, you will be using StableBaselines3.It provides reliable implementation of state-of-the-art RL methods like PPO, DDPG, SAC, TD3, etc to train a policy. Follow the instructions in Section A.6 to set up the conda environment.

Use python scripts/train_sb.py –env_name <ENV> command to start the training.

1. **SBAgent Class**: Use the SBAgent class in 'scripts/mujuco_agent.py' file.

2. **Soft Actor Critic** : Train three models using soft actor-critic (SAC) Algorithm on InvertedPendulum-v4, Hopper-v4, and HalfCheetah-v4 environments and plot the Reward vs episodes curve.

A Instructions to access HPC Cluster of IIT Delhi

The main website for HPC can be accessed at Supercomputing IITD . A basic overview of hpc is available at Overview PPT. The cluster consists of linux machines, and the basic commands for linux can be found at Linux Commands

A.1 Creating Account

You can apply for HPC account using the following link Apply for account

- Login using your Kerberos ID and Password
- Fill **raunakbh** as 'Faculty supervisor (uid)'
- Choose 15th December as Expiry Date

A.2 Accessing HPC

See Accessing HPC
For Linux and Mac systems HPC can be accessed using Terminal, For windows an SSH Client is required. MobaXterm (Download) is the preffered client.

- From the terminal log in using the command: **ssh kerberos_id@hpc.iitd.ac.in**
- At first login perform one-time setup using the following command:
  source /home/apps/skeleton/oneTimeHPCAccountEnvSetup.sh

The access to the cluster is through a few designated login nodes, which allow users to log in, and not run jobs. For running jobs resources (CPU/GPU) have to be allocated using request commands as mentioned in the section below

A.3 Requesting resources and associated charges

Storage is allocated by default in two portions limited to 100GB of space on /home and 25TB on /scratch directory. /scratch data is not backed up.

The resources are allocated through a Portable Batch System(PBS) that performs job scheduling. See the following webpage for PBS Commands. As an example a script (*'HPC_cpu.pbs'*) is provided in starter code. The usage charges are given at Usage Charges.

For this course, each registered student has been allocated **Rs. 165.0** HPC Credits which is equivalent to around 150 GPU Hours or 1500 CPU-Core Hours through a project named **ail722.{kerberos}.course** which is to be utilized for Assignment 2 & 3.

- Use **-P ail722.{kerberos}.course** as project name to use the allotted credits
- Use **amgr login** to login to projects manager
- Use **amgr checkbalance project -n ail722.{kerberos}.course** to check remaining credits
- Use **qstat -nu $USER** command to check running jobs and **qstat -nTu $USER** command to check estimated start time
- Use **qdel JobID** command to terminate job.

A.4 Accessing Internet and Installing Anaconda

For setting up internet access follow the commands at HPC Internet On the terminal with internet access.

- Use command **wget https://repo.anaconda.com/archive/Anaconda3-2024.06-1-Linux-x86_64.sh** to download Anaconda Installer
- Use **sh Anaconda3-2024.06-1-Linux-x86_64.sh** to install Anaconda

A.5 AIL722 Policy Gradient Environment

Run the following commands to setup the environment

- cd ./MujocoEnvs
- conda env create -f rl_env_lin.yml
- conda activate rl_env
- pip install -e .

A.6 AIL722 Stable Baselines Environment

1) Create a conda environment

- cd ./MujocoEnvs
- conda env create -f sb3_env_lin.yml
- conda activate sb3
- pip install -e .

2) Install MujoCo

- wget https://mujoco.org/download/mujoco210-linux-x86_64.tar.gz
- tar -xzf mujoco210-linux-x86_64.tar.gz

3) Move the mujoco210 into /.mujoco/

- mkdir /.mujoco
- mv ./mujoco210 /.mujoco/

4) In case of error: Export the paths to "LD_LIBRARY_PATH" only if you face errors

- export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path-to-mujoco210>/bin
- export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/nvidia

A.7 Tensorboard Logging for Tracking Experiments

In this assignment we will ise Tensorboard for visualising and tracking experiments. On a local machine, dowload the logdire and then you can run tensor board as follows:

- tensorboard –logdir ../data

If you are using a remote server for training, use port forwarding to view the tensor board in your local browser. Read more here

- tensorboard –logdir >path> –port 6006
- ssh -N -f -L localhost:16006:localhost:6006 >user@remote>
- Now open http://localhost:16006 on your browser

A.8 Description of Code Structure

The file structure of the provided code is as follows:

```
TreasureHunt-v2
   images
   agents.py #Your Implementations
   grid.py
   env.py
Template_MonteCarlo.py #Your Implementations
MujucoEnvs
   agents
      base_agent.py
      mujoco_agents.py # Your implementations
   utils
   scripts
      train_agent.py
      train_sb.py
   rl_env_lin.yml # Conda environment file for Linux.
   sb3_env_lin.yml
   config.py # Config File
   setup.py
```

You need to modify only 'mujoco_agents.py' and 'config.py'. For training use the following commands with description in Table 3 below

```
python scripts/train_agent.py --env_name <ENV> --exp_name <ALGO> [optional tags]
```

Tag	Description	Possible values
env_name	The name of the MuJoCo Environment	Hopper-v4, InvertedPendulum-v4, HalfCheetah-v4
exp_name	The algorithm you want to use to train the policy	AC, RL
no_gpu	Include this tag if you want to train on CPU.	
scalar_log_freq	The frequency with which you want to log your metrics like loss and rewards	Positive integer
video_log_freq	The frequency with which you want to log the visual performance of your model	Positive integer
load_checkpoint	You can load a checkpoint model to resume training.	<path to the checkpoint>

Table 3: Training Configuration Tags