CMP9137M Advanced Machine Learning – Workshop Week 6

Summary: This workshop aims to give you practical experience with deep reinforcement learning using value-based and policy-based methods. First, you will test and train agents using a well-known benchmark called "LunarLander". Then you will test and train more complex agents to play the game of "SuperMarioBros2". The materials of this workshop make use of the widely-used and open-source toolkit Stable-Baselines3. The code of this particular workshop has been tested on Windows and with Linux Ubuntu. But some lab PCs running Windows may not have OpenGL-related dependencies. If you want to see visualisations, booth your PC with Ubuntu. Windows otherwise should be okay.

To run the programs of this workshop you need to install the following:

- pip install gym==0.21.0
- pip install swig
- pip install box2d-py
- pip install gymnasium
- pip install pyglet==1.5.11
- pip install stable-baselines3[extra]==1.8.0
- pip install gym-super-mario-bros==7.3.0

Task 1: **Train and evaluate LunarLander agents**

To familiarise yourself with the states, actions, rewards and task of these agents, you should read the following, which will give you an appreciation for the non-trivial decision-making that needs to be made by trial and error.: Lunar Lander - Gym Documentation (gymlibrary.dev)

Work in groups of three students so each of you trains an agent to fill the table below. You should be able to train your agent(s) as follows:

- python sb-LunarLander.py train DQN
- python sb-LunarLander.py train A2C
- python sb-LunarLander.py train PPO

You can evaluate the performance of those agents as follows (see your files to identify seed values):

- python sb-LunarLander.py test DQN SEED_NUMBER
- python sb-LunarLander.py test A2C SEED_NUMBER
- python sb-LunarLander.py test PPO SEED_NUMBER

Algorithm	Avg. Reward w/500k million steps
DQN	
A2C	
PPO	

Compare the learnt behaviour of the agents above against random behaviour by editing your program, setting num_training_steps=100, and pretending to train an agent as follows:

- python sb-LunarLander.py train DQN

Which is the best agent, and which is the worst, and why? Discuss with your peers.

Task 2: **Train and evaluate SuperMarioBros agents**

Open a new terminal tab (ctrl+shift+t), and train & test a DQN agent (for speed purposes) as follows:

- python sb-SuperMarioBros.py train DQN
- [training will take a while, be patient]
- [when training is over, look at your file and identify the value of SEED_NUMBER]
- python sb-SuperMarioBros.py test DQN SEED_NUMBER

Alternatively, each member of your team can train an agent using a different algorithm. In this way, you will end up with three agents, one per algorithm (DQN, A2C, PPO).

Compare the learnt behaviour of the agent(s) above against random behaviour by editing your program, setting the variable `num_training_steps` to 100, and pretend to train it (due to such small number of steps) as follows:

- python sb-SuperMarioBros.py train DQN

What is their performance of those DQN agents according to avg. reward? Discuss with your peers.

Even if training doesn't end during the workshop, analyse their performance as training progresses.

Task 3 [homework]: **Get familiarised with the API of Stable-Baselines3**

3.1 Read the parameters (and default values) of DQN, A2C and PPO implementations. Links:

- classstable_baselines3.dqn.DQN(policy, env, learning_rate=0.0001, buffer_size=1000000, learning_starts=100, batch_size=32, tau=1.0, gamma=0.99, train_freq=4, gradient_steps=1, replay_buffer_class=None, replay_buffer_kwargs=None, optimize_memory_usage=False, target_update_interval=10000, exploration_fraction=0.1, exploration_initial_eps=1.0, exploration_final_eps=0.05, max_grad_norm=10, stats_window_size=100, tensorboard_log=None, policy_kwargs=None, verbose=0, seed=None, device='auto', _init_setup_model=True)

- classstable_baselines3.a2c.A2C(policy, env, learning_rate=0.0007, n_steps=5, gamma=0.99, gae_lambda=1.0, ent_coef=0.0, vf_coef=0.5, max_grad_norm=0.5, rms_prop_eps=1e-05, use_rms_prop=True, use_sde=False, sde_sample_freq=-1, rollout_buffer_class=None, rollout_buffer_kwargs=None, normalize_advantage=False, stats_window_size=100, tensorboard_log=None, policy_kwargs=None, verbose=0, seed=None, device='auto', _init_setup_model=True)

- classstable_baselines3.ppo.PPO(policy, env, learning_rate=0.0003, n_steps=2048, batch_size=64, n_epochs=10, gamma=0.99, gae_lambda=0.95, clip_range=0.2, clip_range_vf=None, normalize_advantage=True, ent_coef=0.0, vf_coef=0.5, max_grad_norm=0.5, use_sde=False, sde_sample_freq=-1, rollout_buffer_class=None, rollout_buffer_kwargs=None, target_kl=None, stats_window_size=100, tensorboard_log=None, policy_kwargs=None, verbose=0, seed=None, device='auto', _init_setup_model=True)

3.2 Look at the source code of each of the following: DQN, A2C, PPO.