Reinforcement Learning – Lunar Lander

Kangjie Yu – MBD Sep 2023

To start, I referred to the PPO documentation in Stable Baselines 3 (https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html) to check the included parameters and their default values. Since we are using Google Colab for training, I initially kept the default values and trained the model for 10,000 timesteps to see the performance. The result wasn't very good, but it seemed to be converging. This suggested that the parameters were reasonably well-designed, as they are recommended in the documentation. I then increased the timesteps to 700,000, which took approximately 20 minutes. With this, I got my first model that could successfully land on the lunar surface without crashing. From there, I started tweaking other parameters.

I focused on several key parameters: learning rate, batch size, number of steps, clip range, gamma, normalize advantage and etropy coefficient.

Learning Rate: This affects the stability of learning. A high learning rate can lead to faster exploration but might cause oscillation and degrade performance. I tried 0.0001, 0.01, 0.0005, etc., but 0.0003 seems to be the best one.
Batch Size: Larger batch sizes require more memory and can slow down the learning process, but they can also contribute to stability. I increased the batch size from 64 to 128 when the learning rate is low.(e.g.: 0.0001 learning rate, 128 batch size, 4000 n_steps)
Number of Steps: Increasing the number of steps helps in collecting more data for each update. Number was increased to 4000 when learning rate is low.
Clip Range: This restricts how much the policy can change in one update. Finding the right value is crucial to avoid instability. 0.2 performed pretty well.
Gamma: This discount factor balances immediate and future rewards. The default value of 0.99 seems high enough to ensure the agent considers future rewards sufficiently.
Normalize Advantage: Keeping this true helps prevent excessive policy updates due to high advantages.
Entropy Coefficient: Adding an entropy term to the loss function prevents the agent from converging to a local optimum too quickly. The default value was 0, and I changed it to 0.01 to encourage more exploration. This did help improve the model, although the improvement wasn't dramatic.

To optimize these parameters, I first lowered the learning rate to 0.0001 to find a better solution. To compensate for the slower learning speed, I increased the batch size to 128 and the number of steps to 4000. However, the results were not satisfactory, possibly indicating that a longer training time was needed. I also played with clip_range, timestamps and other features, but most of them were not performing as well as the default values.

Overall, the initial default parameters were a good starting point, but fine-tuning parameters like the learning rate, batch size, clip range, and entropy coefficient could help achieve better performance. Here are the results after making these adjustments:

N_epochs	**N_steps**	**Gamma**	**Clip_range**	**Normalize_average**
10	4000	0.99	0.2	True
Ent_coef	**Vf_coef**	**Learning_rate**	**Use_sde**	**Target_kl**
0.01	0.5	0.0003	False	None
Batch_size	**Timestamp**	**Score**		
128	1000000	279		