Deep reinforcement Learning

Barto Radman (2062836)

May 29, 2024

1 Generalization of Trained Policies to Different Environment Dynamics

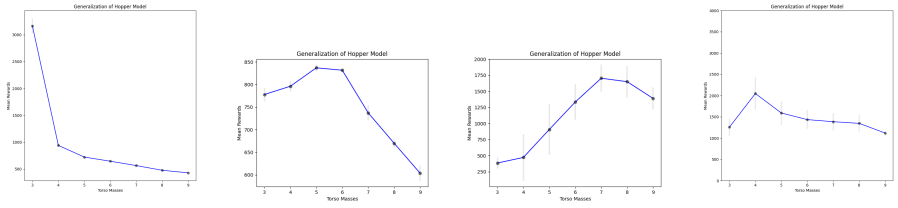
First part of the assignment tasked training of a reinforcement learning algorithm on the Hopper-v4 environment. Agents were trained on different environment parameters (torso mass) across three different seeds. Same agents then had to be tested and evaluated on environments with changing torso masses (ranging from 3 to 9kg) to test the generalizability through simulation to simulation demonstration.

PPO was chosen as an algorithm of choice due to its robustness and ease of implementation. I wanted to test and adjust the state of the art model for the initial task. Two different sets of agents were trained, baseline hyperparameters and adjusted hyperparameters. PPO especially performs well with its baseline parameters, thus the hyperparameters were iteratively adjusted to observe the differences in acquired rewards. After training the models, the performances achieved during training were in the range of two to three thousand. Nine models in total were used to test generalizability across masses (3,6,9 kg across seeds 42, 123, 88). The models training was rendered after training and some of the agents managed to hop until the end of the environment.

Models for the torso mass of 3kg performed the best during training, while the 6 and 9 kg models performed worse with the same hyperparameters across the three seeds. It could be possible that hyperparameter tuning could be done across torso masses, as the heavier models exploit less hopping forward associated actions and more staying alive associated actions to gain reward.

Last, the models were tested on a variety of changing environments. In order to test generalizability of trained policies, various parameters of the environment can be changed. For the purpose of this experiment, the torso mass were changed across the testing environments. Below are the generated plots of agents' performance across different torso masses. The plots display a loss in performance when the environment masses differ from the mass that the model was trained upon. Policies trained on a specific mass are not stable in even slightly different environments [**??**]. The mean rewards were collected for interactions between three models with different seeds then averaged to create the plots.

The fourth plot is an attempt at an ensemble model which could generalize better to a range of masses. The model is trained on the whole range of torso masses for four million time steps and manages to acquire rewards for all of the environments, showing a possibility for policy adaptation to a changing environment.



(a) (b) (c) (d)

Figure 1: Performance of agents across different torso masses and ensemble model (a) 3kg model (b) 6kg model (c) 9kg model (d) ensemble model

2 On-policy vs Off-policy Algorithms

Task of the second assignment was exploration of differences between on- and off-policy algorithms. Policy in reinforcement learning refers to the function which maps states of the environment to the agent's actions. The key difference between such algorithms is the way that the agent interacts with, and adapts to the environment [1]. On-policy agents learn from the data directly collected by executing the current policy being used, adjusting the current policy through maximizing rewards directly dictated by experiences in the environment following that policy. On the other hand, off-policy algorithms can learn from data collected through agent's interaction using a different policy, such as previous or exploratory policies [3]. Off-policy algorithms can improve the policy from any behavior, not necessarily the behavior learned from the current policy.

Algorithms chosen for this assignment were Proximal Policy Optimization (PPO) and Twin Delayed Deep Deterministic Gradient (TD3). PPO was chosen as an on-policy algorithm due to its robustness for continuous action environments and balance between exploration and exploitation. PPO updates the old policy so it is not drastically different from the old policy, widely considered to be state of the art in reinforcement learning (Stable Baselines3). TD3 on the other hand can learn from more gathered data, while utilizing tricks to make more conservative updates, reducing variance and stabilizing learning [3]. Two algorithms were optimized using the same seed for each training (seed=0). First each algorithm was trained and evaluated on its baseline hyperparameters. Then hyperparameters were gradually changed, and agents retrained, to observe changes in behavior. Each evaluation was based on one million training time steps.

Initial baselines hyperparameter initialization of the agents provided poor performance on the BipedalWalker-v3 environment. Baseline PPO managed to achieve the maximal reward of 192, while baseline TD3 remained in negative rewards. What is immediately evident from training the baseline models is that PPO needs more interactions with the environment due to discarding old policy data. However, PPO remained quicker to train and update due to the possibility of running multiple environments in parallel, while continuing to gradually gather more rewards. TD3 performed worse, further emphasizing the importance of hyperparameter optimization. PPOs robustness allowed the baseline to outperform TD3. Next, hyperparameters were updated for each model. Interesting parameters to mention are learning rate, discount factor (gamma) and clip range. Learning rate determines how much the policy parameters are adjusted. Learning rate is important to balance due to the low values being slow to converge, while large values could lead to unstable behavior. Gamma determines the importance of future actions, while the clip range constraints large updates. PPOs performance increased slightly with hand tuned hyperparameters, reaching the reward of 225 while continuing to learn. TD3s performance gradually kept decreasing even after half a million time steps due to its sensitivity to hyperparameters. Likely the model could not learn a good policy and kept exploring inefficiently.

Last experiment was conducted utilizing optimized parameters from SB3 documentation (https://github.com/DLR-RM/rl-baselines3-zoo). Hyperparameters were adjusted gradually for the improvement of the agent's performance and according to SB3 guidelines. Each agent was trained for 250000 steps, getting trained for a number of iterations until the environment was solved. Unsurprisingly, both of the agents kept performing better than the baseline and hand tuned models. PPO took a significantly lower time to train (30 minutes) while TD3 needed much longer wall clock time (4 hours). Both of the agents learned to solve the environment (gathering 300+ reward), TD3 taking approximately half a million environment steps and PPO taking two million environment steps. TD3 showed continuous and stable improvement in the environment, whereas PPO would require more time steps to stabilize its performance. PPO, being an on-policy algorithm, continued improving past one million time steps.

This short experiment confirmed the efficiency assumptions of on- and off-policy agents. In practice, PPO is much easier to implement and use, providing faster wall clock learning and generalizability to different environments while being more simple to tune. TD3 would have been much harder to train even in such a simple environment due to its dependence on tuned hyperparameters.

References

[1] Plaat, A. (2023). Deep Reinforcement Learning, a textbook. https://doi.org/10.1007/978-981-19-0638-1

[2] Rajeswaran, A., Ghotra, S., Ravindran, B., & Levine, S. (2017). EPOPT: Learning Robust Neural Network Policies Using Model Ensembles.

[3] Sutton, R. S., & Barto, A. (2020). Reinforcement learning: An introduction (Second edition). The MIT Press.