Delivery requests

Tommaso Massaglia
s292988@studenti.polito.it

Farzad Imanpour Sardroudi
s289265@studenti.polito.it

Alireza Talakoobi
s289641@studenti.polito.it

I. 2.1: State and action space, masses

A. State Space

There are 11 different variables in the state space which represent the current coordinate, angle, and velocity of different parts such as hinges and joints of the leg. The hopper moves along a 2D space so it has no y coordinate. Each measure is a continuous value in range $[-Inf; Inf]$.

The state space dimensionality is $\mathbf{R}^{11}$

B. Action Space

The action space is a *gym.Box* object with size 3, each array inside the box represents one of the hinges of the hopper (leg hinge, thigh hinge, foot hinge) and specifies the maximum and minimum value of the torque that can be applied to each hinge, in our case $[-1, 1]$, which is a continuous value (a *float32*).

An action is a three dimensional vector always with continuous values between -1 and 1.

C. Masses

The hopper has four masses, one for each of its elements (torso, thigh, leg, foot); the difference between the two environments lies in the torso mass, which has a shift of -1 between source and target, its value is fixed even in domain randomization.

	torso	thigh	leg	foot
source	2.53429174	3.92699082	2.71433605	5.0893801
target	3.53429174	3.92699082	2.71433605	5.0893801

II. 2.2: Policy Implementation

In table II are reported the best performing hyperparameters found for the different configurations along with their Source-Source (S-S), Source-Target (S-T) and Target-Target (T-T) average return over 50 episodes.

model	Best hyper parameter	S-S	S-T	T-T
Reinforce	n-episodes: 50k, lr: 0.001	55	48	11
Actor Critic	n-episodes: 100k, lr: 0.0001	238	215	220
PPO	n-steps: 1M, lr: 0.001	1552	1030	1420
TRPO	n-steps: 1M, lr: 0.001	1629	1473	1722

III. 3: Uniform Domain Randomization

Implementing UDR we chose to use TRPO as a baseline as it performed better than the other policies we tested. Once the best hyperparameters were tested, we tried with different uniform distribution bounds. Reading the results in the table, we can say that UDR accomplished the goal of improving the performance of the policy, and that the bound to the domain distribution parameters worked better when close to the original value.

Agent	**Hyperparameters**	**UDR distribution**	**S-S return**	**S-T return**
PPO	lr: 0.001 steps: 500000	-	1217	1014
TRPO	lr: 0.001 steps: 500000	-	1338	1265
TRPO+UDR	lr: 0.001 steps: 500000	[3.4,4.5] [2.5,3.5] [4.5,5.5]	1436	1380
TRPO+UDR	lr: 0.001 steps: 500000	[3,5] [2,4] [4,6]	1421	1375
TRPO+UDR	lr: 0.001 steps: 500000	[1,7] [0.1,6] [2,8]	945	780

IV. 4: Improvements

Our choiche of improvement was BayRn.

Agent	Hyperparameters	S-S return	S-T return
TRPO+UDR	lr: 0.001 steps: 500000	1436	1380
TRPO + BAYRN	lr: 0.01 steps: 250000	1451	1563
TRPO + BAYRN	lr: 0.01 steps: 500000	1479	1589
TRPO + BAYRN	lr: 0.001 steps: 500000	1531	1619

Learning wasn't very effected by hyperparameters, and going over 500k steps did not bring any meaningful results, rather it was detrimental at times. A lower learning rate proved to be slightly better, but in other tests reported in the paper we saw that a higher one was more consistent.