Adaptive agents using self-learned curricula

Ahmed Nawaz
a.nawaz@innopolis.university

Introduction

Reinforcement learning is a target
oriented algorithms that learn how to achieve a complex goal or
how to improve over several steps in a particular scenario. With the recent breakthrough in AI technologies made by Neural Networks, a new term evolved that combines RL and Artificial Neural Networks and is called Deep Reinforcement Learning. DRL is a promising approach for developing automatic locomotion control in Robots. One of the modification, proposed by (Yu, Turk et al. 2018) is Curriculum Learning. This techniques aid's the learning by providing external force to joints. This research project aims to use the technique of Curriculum Learning to train Humanoid robot.

Experimental setup

The idea behind the Curriculum learning is to provide assistance to the robot initially to achieve its goal and later on, gradually remove the assistance force to make the robot robust to the environment. The whole process involves dividing the space into equal curriculum while in each curriculum the robot has to learn a lesson. The process starts with the simplest lesson at the beginning and gradually increasing the difficulty towards the original task.

One interesting scenario is to study the behavior of Humanoid Robot in pybullet problem by varying the training conditions. This can be done by applying external forces to the pelvis of humanoid robot. However, the application of external process should be to aid the walking behavior i.e., it should be applied in such a way that it cancels or reduces the lateral inclinations and only be present in forward direction.

For the practical considerations of this task, two primary modifications are made in the code,

1. In the robot_base.py file, a new constructor is added to store the name of the body (bodyName). This will later help to exert force on only Pelvis joints not the other parts of the body. PyBullet provides a function applyExternalForce that is useful in this case. The usage of this function is copied from its documentation and is given below,

   The input parameters are:

required	objectUniqueId	int	object unique id as returned by load methods.
required	linkIndex	int	link index or -1 for the base.

required	forceObj	vec3, list of 3 floats	force/torque vector to be applied [x,y,z]. See flags for coordinate system.
required	posObj	vec3, list of 3 floats	position on the link where the force is applied. Only for applyExternalForce. See flags for coordinate system.
required	flags	int	Specify the coordinate system of force/position: either WORLD_FRAME for Cartesian world coordinates or LINK_FRAME for local link coordinates.
optional	physicsClientId	int	

Table. 1

2. The second modification is made in robot_locomotors.py, method calc_state(self). First by running a loop on body parts, the velocity of Pelvis is obtained. This is a 3d vector that contains the velocities along x, y and z axes. Next a 3D vector is generated using uniform distribution and applied to the Pelvis.

Results

The adaptive behavior of the robot as described in the paragraph above is tested on humanoid robot in PyBullet environment. The two modifications were made and the behavior of the humanoid robot is observed with the already trained neural networks. I have observed that for some of the seeds there is not much difference in the behavior of the robot, while for other seeds the robots has evolved its behavior in such a way that it has improved the reward. The updated robot_locomotors.py file is available at the following git.

https://github.com/ahmednawaz803/Behavioral-and-Cognitive-Robotics-Course/tree/master/Project

References

Yu, W., et al. (2018). "Learning symmetric and low-energy locomotion." **37**(4 %J ACM Trans. Graph.): Article 144.