RL with Real Robots UPDATED BY: Dhawal Gupta at October 22, 2019

1 Setting up of a Task

This section addresses the question of how to setup a real world robotics task so that off the shelf implementation of algorithms are able to perform reasonably and effectively. The paper uses rllab implementation of TRPO. Elements that are to be considered while setting put a real world robotics task include

- medium of data transmission
- concurrency
- ordering
- delay of computation
- low level actuation and their frequency of operation.

1.1 UR5 Reacher

The UR5 Reacher is a 6 Joint Robotic Arm manufactured by Universal Robotics. The low level controller is called as *URControl* and it can be programmed by the use of a script called *URScript* or a TCP/IP connection. Each packer to *URControl* contains sensorimotor information comprising of angular position, target velocities etc.

The second and third joint are also actuated from the base only. **Observation Vector** : The observation vector for the robot includes the

- joint angles
- joint velocities
- vector difference between and fingertip coordinates
- previous action (this helps in learning with delays)

Concurrency : In simulated task, normally each action and computation of the robot can be easily synchronized with the simulation environment, where each time step progresses according to the agent, whereas in the real world, the environment keeps on changing while the agent is performing its updates, this results in the agent always operating on delayed sensorimotor information.

The computational steps for the UR5 are split into 2 parts i.e. *Robot Communication Process* and *RL process*. where the RL process is further broken down into an environment and agent process, where the environment build up the observation vector and reward form the sensorimotor information and agent computes actions, and updates, and the basic loop goes as follows : We start with an action command and construct it into a proper format and send it to the actuation thread with pushes it forward, which results in the completion of an action cycle, the sensor thread waits for a sensor packet observation and forwards it to the environment process , which constructs the observation vector and reward for the environment, this is now used as a tuple to update the policy and then an action is produced based on the the observed vector, and the cycle repeats. TRPO is used for learning updates, where the updates are performed between different episodes, they occur in the agent thread and not on every action cycle or time step.

The robot actuation time in the mentioned setting defaults to 8ms. The action cycle time , is the time step duration between 2 subsequent action updates by the agent policy.

1.2 Action Space

Choosing a control variable which can have a strong effect on the agents state space is a good idea for action. The paper uses velocity as a mode of action rather than using position control (which requires double smoothing). The paper present a graph which shows the correlation between different properties of motor like their acceleration, torque and current with respect to the action taken like velocity or position after a certain number of packets and it seems as if the effect of action is reflected after 2 packets normally. Also in the case of velocity, the effect of action dies normally dies down after the step, whereas the effect of position still remains.

2 Steps and Tricks to set Neural Reinforcement Controllers

2.1 Modelling a Control Task

1. **State Information :** State information should be derived from sensor data and should be rich enough to suppose Markov property, redundancy is not a problem.
2. **Actions :** Action suffer a trade-off, where low amount of action results in faster learning as compared to more accurate control, and actions should be adequate to enable agent to reach goal states.
3. **Control Interval** δ_t : The controller is normally able to adapt to δ_t, and this suffers from the same issue where small δ_t gives more accurate and quality control versus faster learning.
4. **Terminal/Non Goal States :** Normally goal states are represented by a set of tolerable goal values around the vicinity of true goal point. All these points are said to be in the set of X^+. In the case of non terminal tasks the episode doesn't end after reaching X^+ and has to maintain that state. In these cases the target policy need to include policy which stabilizes the system within the target region. $\mathbf{X}^*$ also suffers from the fact that smaller X^+ means accurate control but slower learning.
5. **Choice of** X^- X^- are the set of undesirable states and whenever a state inside them is encountered the episode is stopped and the agent receive a huge negative reward.
6. **Choice of immediate and final reward** : They should closely resemble the specification of problem, and immediate rewards might direct the agent to the goal, but might not reflect the original intention of the task.
7. **Discounting** : Discounting can help simplify the problem and requires less assumption making learning simpler
8. **Choice of** X^0 : The starting state should be from the intended working space, and we can start the learning from simple states first and then increase the range to improve the learning process.
9. **Length of Episode** N : Choice of N is not that critical theoretically, but practically it can considerably influence learning behaviour

2.2 Tricks

1. **Scaling Input Variables** : Like in supervised learning, all inputs should be scaled to 0 means and standard deviation 1.

2. X^{++} **Trick** We call the set of state action for which we can fix the reward to goal are called X^{++}, this can increase the speed of learning process.

3. **Artificial Training Transitions :** Because of sparsity of the occurrence of goal states, we can inject some artificial transitions to gaol states from the X^{++} states. These are called as hint-to-goal heuristic

4. **Growing Batch :** The basic idea is to increase the batch size to get more relevant transitions when the performance of the policy increases.

5. **Training Neural Q Function :** Choice of MLP is not that important, but it is important to have a good optimization algorithm.

6. **Exploration :** A simple ϵ greedy policy if often sufficient, and if the starting states are well distributed, then in conjunction with growing batch we do enough exploration while learning and the advantage is that that following of greedy policy leads the agent to show the optimal policy while training itself.

7. **Delays** : Delays cant be simply ignored, a simple remedy it this is to use previous action information in the current state information.

3 Unstated Problem Constraints

This paper goes on to discuss the effect of goal states constraints on the learning of the agent, details of which are often alluded in different papers. They discuss that goal regions that are farther away from the robot are easier to learn in the higher dimension case, whereas there is literally no learning from 3 to 6 degrees for location that include the goal regions near a robot. This goes onto tell tell the importance of mentioning the goal regions in the problem definition of a reinforcement learning agent.

They are also going to release an ROS Gym a bridge between Robot Operating System and Open AI Gym which will smoothen the process of conducting experiments on available models in the ROS using the OpenAI Gym definitions, though I was not able to find it online yet.