Modular Reinforcement Learning for Policy Stitching

Atul Balaji
EE16B002
Department of Electrical Engineering
Indian Institute Of Technology, Madras
ee16b002@smail.iitm.ac.in

Rohithram R
EE16B031
Department of Electrical Engineering
Indian Institute Of Technology, Madras
ee16b031@smail.iitm.ac.in

Abstract

Learning in robotics environments involves dealing with sparse and binary rewards, which generally increase the difficulty of the training process. We use the Hindsight Experience Replay (HER) algorithm for learning various robotic manipulation tasks such as reaching and picking in this sparse reward setting. We have made use of the Jaco arm environment, a comprehensive, multi-joint framework. Our main contribution has been to convert this environment into a goal-based one in order that it will be compatible with HER.
We demonstrate our approach by implementing two robotic manipulation tasks: Reach and Pick, by modifying these as goal-based environments and subsequently implementing Hindsight Experience Replay. We also attempt to implement naive sequential stitching for the composite Pick and Place task, where we attempt to execute the two skills sequentially.

1 Introduction

In the context of robotic manipulation, the rewards are **sparse**, binary and are received only after the goal is achieved. For example, consider the task of sliding a puck into a goal. The environment uses a sparse reward of -1 if the goal is not achieved and 0 if it is achieved (within some tolerance). Learning from sparse rewards is significantly more difficult than from dense rewards. It is sometimes possible to define and make use of shaped (dense) rewards, but, sparse rewards are known to be more realistic in robotic environments.
To overcome the issue of learning in the sparse reward setting, we use the Hindsight Experience Replay algorithm which is best suited for learning in sparse reward environments. The environments have been made goal-based so that there is compatibility with the HER algorithm. This also gives us the option to define sub-goals to learn each primitive skill in the case of sequential execution of two or more skills, and to provide dense rewards if necessary.
This project covers the implementation part of a larger effort to develop algorithms for accurate policy stitching (execution of two or more skills in sequence such that the transitions are smooth).

2 Related Work

In this section, we give a brief overview of the algorithms and techniques used in this project, as well as those whose ideas are central to its formulation.

2.1 Deep Deterministic Policy Gradients (DDPG)

Deep Q-Networks (DQNs) are effective for learning in discrete action spaces, but, in high dimensional action spaces, discretization of actions is not feasible since the number of actions is extremely large, rendering DQNs [3] ineffective. Deep Deterministic Policy Gradients (DDPG) [4] is a model-free, off-policy RL algorithm for learning in continuous action spaces. It serves as a reliable deep

.

reinforcement learning approach for continuous control problems, such as robotic manipulation.
In DDPG, two neural networks are maintained: a target policy (actor) $\pi : S \rightarrow A$ and an action-value function approximator (critic) $Q : S$ x $A \rightarrow R$. The critic approximates the actor's action-value function Q_π.
Episodes are generated using a behavioral policy which is a noisy version of the target policy: $\pi_b(s) = \pi(s) + N(0, 1)$. The critic is trained in a similar way as the Q-network in DQN but the targets y_t are computed using actions returned by the actor:

$$y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$$

The actor is trained with mini-batch gradient descent on the loss function $L_a = E_s[Q(s, \pi(s))]$, where s is a sample from the replay buffer. The gradient of the loss function with respect to the parameters of the actor can be computed by backpropagation through the combined actor and critic networks.

2.2 Hindsight Experience Replay (HER)

Dealing with sparse rewards is one of the biggest challenges in Reinforcement Learning. Yet, they are of relevance in robotic arm manipulation and control in physical environments, where it is not convenient to design dense reward functions. Once again, consider the example of sliding a puck into a goal. Rewards of -1 are given when the goal is not achieved and 0 only when it is achieved. During training, since most of the attempts at hitting the target are unsuccessful, rewards of -1 will be received and therefore most of the episodes do not contribute to learning.
The key insight here is that, though the agent has not achieved the specified goal, it has, however landed the puck in another location, which could have been the goal (achieved goal). In HER [1], the achieved goal is included as part of the learning process by treating it as a virtual goal (pretending that this is the goal the agent wanted to achieve). This means that some goal has been achieved, though it is not the one specified initially. This provides a non-trivial reward to the agent and thus facilitates learning.
The algorithm of HER is based on the concept of experience replay but with an important extra step. HER must be used along with an off-policy algorithm such as DQN or DDPG which provide a behavioral policy. First, a goal g is sampled from a distribution and then, the agent experiences episodes of states, actions and rewards. The transitions $(s_t||g, a_t, r_t, s_{t+1}||g)$ are stored in a replay buffer as in standard experience replay. The additional step in HER is that, after this, each trajectory is replayed with the goal now being set as the final state achieved in the episode. This way, HER also makes use of failure episodes (where goal is not achieved) for learning.

2.3 Transition Policies

Consider the problem of learning to perform composite tasks. These tasks are made up of two or more subtasks (skills) which are called primitive skills. In such cases, it is necessary to smoothly navigate from the ending state of the first skill to the initial states (in the initiation set) of the next skill. This becomes a challenge when the rewards are sparse and binary.
The transition policies framework [2] proposes a solution to this problem. In order to create a dense reward function, it introduces the concept of proximity predictors. A proximity predictor outputs the proximity of the end state of the first skill to the initiation set of the next skill and acts as a dense reward function for the transition policy.
The framework consists of primitive skills $(p_1, p_2, ...p_n)$, primitive policies, transition policies and a meta-policy. The meta policy $\pi_{meta}(p_c, s)$ chooses a primitive skill p_c to be executed. Prior to this, the transition policy for p_c is executed to bring the current state to the feasible initial states (initiation set) for p_c.
The proximity predictor is used in training to provide the proximity reward, which is a measure of how close the transition states are to the initiation set of the next skill. It is trained to minimize the mean squared error of proximity prediction:

$$L_p(w, B^S, B^F) = \frac{1}{2} E_{(s,v)\sim B^S}[(P_w(s) - v)^2] + \frac{1}{2} E_{s\sim B^F}[P_w(s)^2] \quad (1)$$

where B^S and B^F are collections of states from success and failure transition trajectories, respectively.

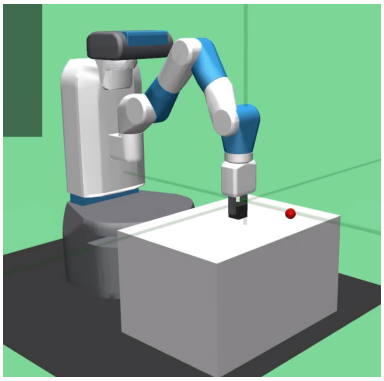

Figure 1: Fetch Pick and Place

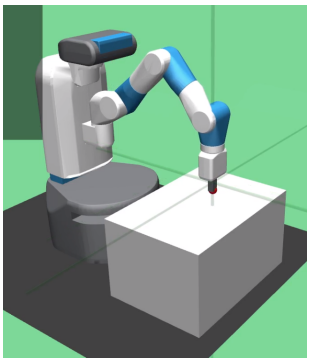

Figure 2: Fetch Reach

3 Environment

As was mentioned earlier, this project makes use of the **Jaco** environment, a multi-joint robotic arm environment. It consists of 6 joints (4 degrees of freedom), including a hand, with 3 fingers. Actions are used for torque control of the joints and the fingers. The environments are simulated using the Mujoco physics engine [5].

4 Approach

In this project, we have trained our agent using Hindsight Experience Replay, along with DDPG used to sample actions from a behavioral policy. The applicability of HER has been tested on a few relatively simple goal-based robotics environments such as Fetch, where the dynamics of the robot are of lower complexity. The above figures represent the Pick and Place, and Reach tasks on the Fetch environment. The optimal policies for these skills are learned to a high degree of accuracy after training for 1.5 million epochs. These environments function as the baselines for our project.
In order to ascertain the feasibility of this algorithm on environments more complicated than Fetch, we have conducted experiments on the Jaco environment described above, which consists of a robotic arm, which can rotate about its axis and has three movable fingers for grasping objects. The reason behind choosing this environment is the availability of a **larger action space** and continuous actions that can be used to control the fingers. This also makes it possible to learn more nuanced tasks such as Tennis serve (toss + hit), locomotion, etc., thereby making it more useful for application on real world physical robots.
For the Jaco environments, before implementing HER, it was necessary to make the environment goal-based. This was done by sampling a goal near the hand (with some noise added), which would act as the desired goal. The observation vector which is used for training is of shape 67 and consists of gripper (hand) position, object position, object velocities, gripper state, gripper velocities, gripper acceleration. Also, a distance threshold is fixed for the maximum distance between the achieved goal and desired goal. Based on this, the reward is computed as being -1 if the distance exceeds the threshold and 0 otherwise.

5 Experiments

The environments presented below are extensions of the basic Jaco environment used in the transition policies framework [2]. Objects such as boxes and balls have been incorporated into the environment, in order to suit the respective tasks as follows.

- **Reaching Task:**
  In the JacoReach environment, an object (red box) is placed at a distance from the arm (random reset every episode) and the goal is to reach the object (within some tolerance). A distance threshold of 0.08 has been used for this task.
- **Picking Task:**
  Our second experiment makes use of the JacoPick environment. This environment consists

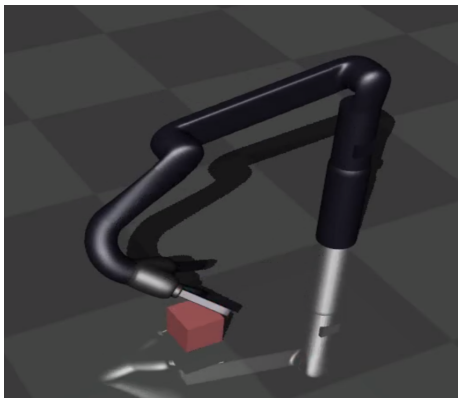

Figure 3: **Jaco Reach**: Test success rate = 0.8

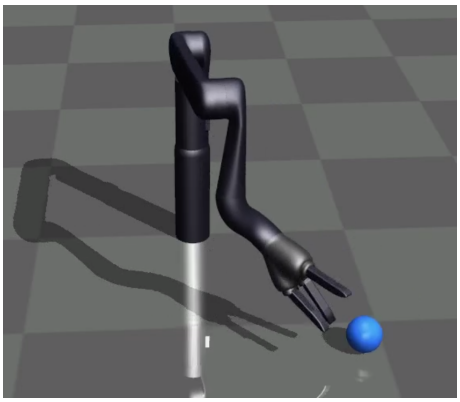

Figure 4: **Jaco Pick**: Test success rate = 0.7

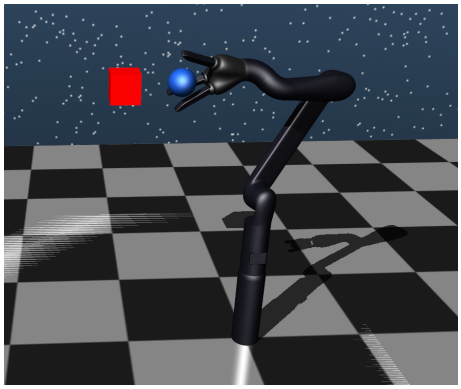

Figure 5: **Jaco Pick and Place**: In few of the episodes, we have observed that the tasks of picking the blue ball (object) and placing it on the red box (target) are stitched sequentially.

of an object (blue ball) which must be picked by the arm. The distance threshold for this task is set as 0.01.

- **Pick and Place task:**
  We have also attempted to implement a composite task, which we call Pick and Place. This environment consists of an object (blue ball) that needs to be picked and subsequently placed on a target (red box). This task therefore consists of two primitive tasks - Picking and Placing, which must be stitched together. For this purpose, we have attempted naive sequential stitching, which involves executing both skills sequentially, without considering overlap of start and end states. The results for this composite task, however have not been successful and the placing is done for only a small fraction of episodes.

Task	Train success rate (out of 1)	Test success rate (out of 1)
Jaco Reach	0.9	0.8
Jaco Pick	0.8	0.7
Jaco Pick and Place	0.25	0.2

6 Conclusion

In this project, we have demonstrated the effectiveness of using Hindsight Experience Replay in the Jaco environment, after making it goal-based, for some robotic arm manipulation tasks, namely Reach and Pick. Naive sequential stitching of two skills has also been attempted for the Pick and Place task.
Future improvements and developments would mainly revolve around:
(a) implementations of naive sequential stitching which yield higher success rates.

(b) developing algorithms for policy stitching of two or more skills that outperforms naive sequential stitching and achieves results that fall in the lines of the transition policies approach

References

[1] Marcin Andrychowicz et al. "Hindsight Experience Replay". In: *CoRR* abs/1707.01495 (2017). arXiv: 1707.01495. URL: http://arxiv.org/abs/1707.01495.

[2] Youngwoon Lee et al. "Composing Complex Skills by Learning Transition Policies". In: *Proceedings of International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rygrBhC5tQ.

[3] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[4] David Silver et al. "Deterministic Policy Gradient Algorithms". In: *ICML*. 2014.

[5] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control." In: *IROS*. IEEE, 2012, pp. 5026–5033. ISBN: 978-1-4673-1737-5. URL: http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.