Combining Hindsight Experience Replay with Hierarchical Reinforcement Learning

Armaan Sethi

I. Introduction

THERE has been a wide range of success in learning policies for sequential decision-making tasks by using reinforcement learning combined with neural networks. These successes include perfecting game-play of Atari games, defeating the best human Go player, hitting a baseball, screwing a cap on a bottle, door opening and helicopter control. Sparsity in rewards has long been a challenge in reinforcement learning. However, recently OpenAI has created an algorithm that works exceptionally well for these tasks called Hindsight Experience Replay (HER). HER is an algorithm for reinforcement learning that specializes in learning from failure. When a failure occurs, instead of just recording the actions taken lead to a failure, it records that the actions could have led to a success if the goal was in a different place. Another common approach in reinforcement learning is using Hierarchical Reinforcement Learning (HRL) in which a higher-level policy generates subgoals that should be learned. In this paper, we explore the possibility of combining HER and HRL, applying HER to not only just goals, but also actions generated by higher-level policies.

Specifically we apply this combined method on the "Reaching", "Pushing", "Sliding", and "Pick & Place" tasks (See Sec. IV for environment details).

II. Related Work

OpenAI introduced a suite of challenging continuous control tasks and presented a set of concrete research ideas for improving reinforcement learning algorithms. They released environments for existing robotic hardware, such as the Fetch robotic arm and the Shadow Dexterous Hand. In this paper, we only used the Fetch environments.

A. Hindsight Experience Replay

One of the biggest challenges in Reinforcement Learning is sparse rewards, previous attempts have focused on introducing complex reward functions to solve the issue. However, Hindsight Experience Replay specializes in learning from failures by not only recording the actions taken lead to a failure, it records that the actions could have led to a success if the goal was in a different place, thus allowing the system to effectively deal with sparsity while using a very simple binary reward function.

B. Hierarchical Reinforcement Learning

Many complicated tasks cannot be effectively expressed with a single policy. Hierarchical Reinforcement Learning was introduced to solve this issue by breaking the task down into a hierarchy of sub-actions at different levels of abstraction. Each of the smaller sub-task is then trained with reinforcement learning. This type of decomposition allows for learning of complex tasks and easier knowledge transfer between tasks with shared components.

III. Problem Definition

The problems used are defined by the environments provided by OpenAI. The tasks including reaching, pushing, sliding, and a more complicated "Pick & Place" task. These tasks are based on the 7-DoF Fetch robotics arm, which has a two-fingered parallel gripper. In the reaching task, the Fetch robot must move the gripper to a specified target position. However, since the reaching task is very simple and was learned very quickly, (1 or 2 epochs) I did not evaluate it. In the pushing task, the Fetch robot must move a box, that is placed on a table, to a target location also on the table. In the sliding task, a puck is placed on a long slippery table and the target is out of the robot's reach. This forces the robot to hit the puck with a force in order for the puck to stop at the location due to friction. In the "Pick & Place" task, the Fetch robot must grasp a box and move it to the target location that may be located on the table surface or in the air above it.

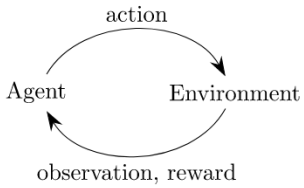
The environment that OpenAI provides consists of a couple different spaces and a step function. The action space describes the format of all valid actions. In these environments, the action space always consisted of a 4 vector of floats between -1 and 1. This meant that every action specified must be given in this format. There is no documentation on what this vector corresponds to, but this is on purpose. Reinforcement learning does not need to know what the actions are in order to learn how to accomplish the task. Instead, it will explore the action space in order to learn what actions are predicted to return the highest reward.

The observation space describes the format of all of the observations that will be received from the environment. In these environments, the observation space consisted of the desired goal, achieved goal, and an observation array of size 10 that contains floats. The positions of the goals were simply the Cartesian coordinates. The observation array is also something that should be learned by the agent, and so there is no documentation. After many observations it is possible to conclude that most values corresponds to the configuration of the arm. However, some values have still remained a mystery to me.

The last important aspect provided by the OpenAI gym is the step function. The step function is how the agent performs

an action in the environment. The only parameter is an action (that is in the action space), and 4 objects from that time step. These objects are the observation, the reward, a boolean that specifies if it's time to reset the environment (when it is time for the episode should be terminated), and an info dictionary for any extra information.

Fig. 1. Agent-Environment Loop.



IV. Methods

A. Deep Deterministic Policy Gradients (DDPG)

The model-free reinforcement learning algorithm that I used was Deep Deterministic Policy Gradients. This algorithm was used because it is the same algorithm used by the original Hindsight Experience Replay paper. Any model-free reinforcement learning algorithm could work with HER and HRL. In DDPG there are two neural networks that work together. This is often called an actor-critic algorithm. The first neural network explores the action space. The second neural network tries to predict the reward from given action that the actor chooses, and the observation. It gives feedback on the action chosen by the actor, so it is often called the critic.

B. Hindsight Experience Replay (HER)

Hindsight Experience Replay involves expanding upon any model-free reinforcement learning algorithm. In normal reinforcement learning algorithms do not learn anything from failure. However, HER implements something humans do instinctively. Even though the original goal was a failure, the agent still accomplished something. By substituting the original goal with a goal that was achieved, the reinforcement learning algorithm can obtain a learning signal since it has achieved some goal; even if it wasnt the one that we meant to achieve originally. If we repeat this process, it will eventually learn how to achieve arbitrary goals, including the goals that we really want to achieve. This allows us to learn complicated tasks, even if we have never achieved the original goal in the early attempts.

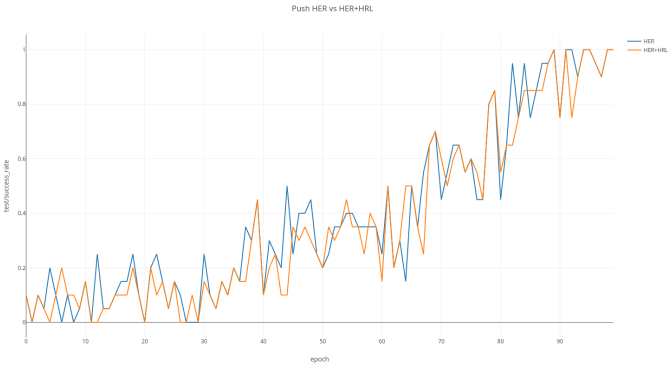
C. Hierarchical Reinforcement Learning (HRL)

Hierarchical Reinforcement Learning is can mean a number of things. It can mean having actions generated by a higher-level policy in order to create subgoals in complicated tasks. It can also mean simply splitting up a task into different subtasks. In this task I split the tasks into simpler subtasks. For the "Fetch Push" and "Fetch Pick and Place" environment, the only subtask was reaching the block that must be moved. For the "Fetch Slide" environment the only subtask was reaching the puck that the robot is interacting with. Since these tasks were relatively simple, it was difficult to come up with good subtasks.

V. Results

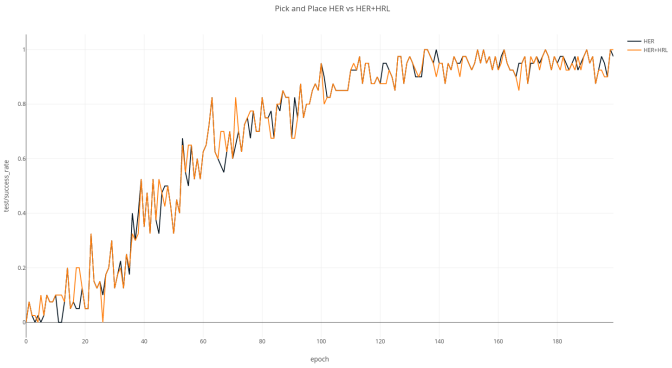
I did not evaluate the "Fetch Reach" environment since it was very simple and reached 100% success in less than 3 epochs. In order to evaluate the methods I ran both the HER and the HER combined with HRL on the same computer, with the same seed, number of epochs, and number of CPU's. At first I had a separate version of my combined algorithm, but in order to evaluate it fairly I decided to use the HER baseline provided by OpenAI as foundation for my algorithm. Then I added my changes to the algorithm by adding a subtask per task.

Fig. 2. Fetch Push



For the "Fetch Push" environment it was very difficult to find any differences between HER alone and HER combined with HRL. This can be due to the task being relatively simple, and the agent learned the subtask and the original task at the same time. This task was learned very well and the agent will perform with near 100% success every time. A video of this task can be found here: https://youtu.be/PcBb0IYE4F0.

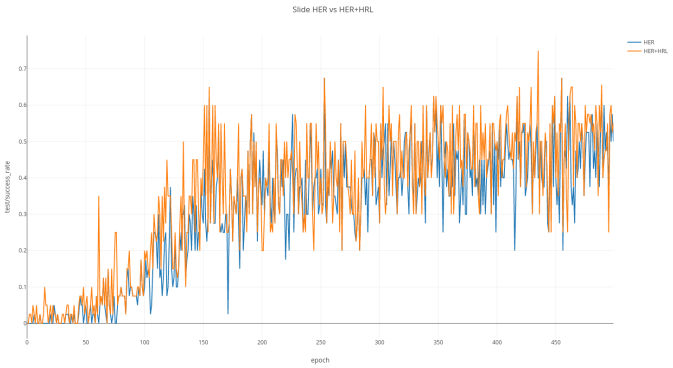
Fig. 3. Fetch Pick and Place. - 200 epochs, 2 CPU's



The "Fetch Pick and Place" environment was very similar to the "Fetch Push" environment in terms of results. It took more epochs to start getting 100% success rate on the test data, but it eventually got there. My combined method did not seem to effect much, but it did not cause it to perform worse. The

trained agent also performs very well, but has trouble when the goal is very high. A video of this task can be found here: https://youtu.be/pPzTOkPKF2o.

Fig. 4. Fetch Slide - 500 epochs, 4 CPU's



The "Fetch Slide" environment never reached 100% success over the 500 epochs. At first I tested it with 1 CPU, and neither method never reached more than 20% success, but with 4 CPU's my method reached a 70% success rate on the test data. HER alone only reached a 60% success rate on the test data. This dramatic increase in the success rate shows that with many more CPU's it may be possible to reach 100% success rate. It also seems that for difficult tasks HER alone may be worse than my method. A video of this task can be found here: https://youtu.be/7k19-bpJLTA.

VI. Conclusion and Future Work

There are many things to be concluded from this experiment. I could not produce the results from the HER paper, but this was because I did not use a computer as powerful as they used. So, I would like to test my method on a more powerful computer and see if the results change at all. I would also like to test my method on more complicated tasks. The tasks I tested this on were relatively simple and did not have a complicated structure to them. An example of a complicated task would be a combination of the tasks I evaluated. Such as, pick a box and place it at the goal followed by pushing the puck to the same goal. I would also like to add obstacles to the environment and see how it effects learning. There are methods that allow these tasks to be transfered to be real robots, and if I had more time I would have tried to evaluate this on the Fetch. This project allowed me to explore reinforcement learning for the first time and I learned a lot.

Acknowledgment

Thanks to OpenAI for their environments and baselines.

Thanks to Professor Alterovitz for teaching and giving this opportunity.

References

[1] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, Wojciech Zaremba. (2018). Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *arXiv preprint arXiv:1802.09464.*

[2] Levy, A., Platt, R., and Saenko, K. (2017). Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948.*

[3] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In Advances in *Neural Information Processing Systems*, pages 50555065.

[4] Dayan, Peter and Hinton, Geoffrey E. Feudal reinforcement learning. In Hanson, S. J., Cowan, J. D., and Giles, C. L. (eds.), Advances in *Neural Information Processing Systems 5*, pp. 271278. Morgan-Kaufmann, 1993.