Data-Efficient Hierarchical Reinforcement Learning

Authors: Ofir Nachum, Shixiang Gu, Honglak Lee, Sergey Levine

Presented by: Samuel Yigzaw
OUTLINE

Introduction
Background
Main Contributions
Related Work
Experiments
Conclusion
INTRODUCTION
- Deep reinforcement learning has performed well in areas with relatively small action and/or state spaces
  - Atari games
  - Go
  - Simple continuous control tasks
- When action and state spaces are both large and continuous, much work needs to be done
If you told your robot maid to go pick up groceries for you, how would it do that?

If it was a human maid, the task would be accomplished by breaking down the requirements necessary in order to complete the goal.

- Top level: go to store, buy groceries, come back
- Breakdown of “go to store”: leave house, walk down street, enter store
- Breakdown of “leave house”: walk to front door, open door, walk through door, lock door
- Breakdown of “walk to front door”: basic motor actions which aren’t consciously processed
Hierarchical Reinforcement Learning

- This is an inherently hierarchical method of accomplishing tasks
- Hierarchical Reinforcement Learning (HRL) is the area of RL which focuses on trying to bring the benefits of hierarchical reasoning to RL
- In HRL, multiple layers of policies are learned, where higher level policies act over which lower level policies to run at some moment
- There are many presumed benefits to this:
  - Temporal and behavioral abstraction
  - Much smaller action spaces for higher level policies
  - More reliable credit assignment
HIRO: Hierarchical Reinforcement learning with Off-policy correction

- 2-layer design
- Uses a high-level policy to select goals for a low-level policy
- Provides low-level policy with a goal to try to achieve (specified as a state observation)
- Uses off-policy training with a correction in order to increase efficiency
BACKGROUND
Off-policy Temporal Difference Learning

- The main learning algorithm used is TD3, a variant of DDPG

- DDPG
  - Q-function $Q_\theta$, parameterized by $\theta$
    - Trained to minimize $E(s_t, a_t, s_{t+1}) = \left(Q_\theta(s_t, a_t) - R_t - \gamma Q_\theta(s_{t+1}, \mu_\phi(s_{t+1}))\right)^2$
  - Deterministic policy $\mu_\phi$, parameterized by $\phi$
    - Trained to maximize $Q_\theta(s_t, \mu_\phi(s_t))$, over all $s_t$

- Behaviour policy used to collect experience is augmented with Gaussian noise
  - Helpful for off-policy correction
MAIN CONTRIBUTIONS
Hierarchy of Two Policies

- High-level policy $\mu^{\text{hi}}$ and low-level policy $\mu^{\text{lo}}$
- $\mu^{\text{hi}}$ observes state and produces a high-level action (a goal state) $g_t \in R^d$
  - Every c steps, sample a new goal: $g_t \sim \mu^{\text{hi}}$
  - Otherwise, use a transition function, $g_t = h(s_{t-1}, g_{t-1}, s_t)$
- $\mu^{\text{lo}}$ observes $s_t$ and $g_t$ and produces a low-level action $a_t \sim \mu^{\text{lo}}(s_t, g_t)$
- Environment yields reward $R_t$
- Low-level policy receives intrinsic reward $r_t = r(s_t, g_t, a_t, s_{t+1})$
  - $r$ is a fixed parameterized reward function
- Low-level policy stores the experience $(s_t, g_t, a_t, r_t, s_{t+1}, h(s_{t-1}, g_{t-1}, s_t))$ for off-policy training
- High-level policy stores the experience $(s_{t:t+c-1}, g_{t:t+c-1}, a_{t:t+c-1}, R_{t:t+c-1}, s_{t+c})$ for off-policy training
Parameterized Rewards

- Goal $g_t$ is specified as the difference between current state $s_t$ and desired state $s_t + g_t$

- Simple goal transition model could be $h(s_t, g_t, s_{t+1}) = s_t + g_t - s_{t+1}$
  - This leaves the desired goal constant as $s_t$ changes

- Intrinsic reward is a parameterized reward function based on Euclidean distance between the current observation and goal observation
  - $r(s_t, g_t, a_t, s_{t+1}) = -\|s_t + g_t - s_{t+1}\|_2$

- The low-level policy is trained with an input-space that includes $s_t$ and $g_t$

- Intrinsic rewards allow the lower-level policy to receive dense relevant reward signals immediately, before any task-specific rewards are available
Basic Design

1. Collect experience $s_t, g_t, a_t, R_t, \ldots$

2. Train $\mu^{lo}$ with experience transitions $(s_t, g_t, a_t, r_t, s_{t+1}, g_{t+1})$ using $g_t$ as additional state observation and reward given by goal-conditioned function $r_t = r(s_t, g_t, a_t, s_{t+1}) = -||s_t + g_t - s_{t+1}||^2$.

3. Train $\mu^{hi}$ on temporally-extended experience $(s_t, \tilde{g}_t, \sum R_{t:t+c-1}, s_{t+c})$, where $\tilde{g}_t$ is re-labelled high-level action to maximize probability of past low-level actions $a_{t:t+c-1}$.

4. Repeat.
Two-Level HRL Example
Off-Policy Corrections for Higher-Level Training

- Previous two-level HRL algorithms use on-policy training
  - As low-level policy is trained, the high-level policy is trying to solve a non-stationary problem
  - Old off-policy experience may have different transitions for the same state and goal
- Off-policy algorithms generally have better sample efficiency than on-policy algorithms
- HIRO applies an off-policy correction factor during training
- Allows HIRO to take advantage of off-policy sample efficiency benefits
Off-Policy Corrections for Higher-Level Training

- Recall the higher-level policy experience \((s_{t:t+c-1}, g_{t:t+c-1}, a_{t:t+c-1}, R_{t:t+c-1}, s_{t+c})\)
- Goal: convert to state-action-reward transitions \((s_t, g_t, \sum R_{t:t+c-1}, s_{t+c})\)
- These can be pushed onto the replay buffer of any standard off-policy algorithm
- Problem: past instances of lower-level policies will exhibit different behavior given the same goals \(g_t\) and end up in different states \(s_{t+1:t+c}\)
Off-Policy Corrections for Higher-Level Training

- Idea: change the $g_t$ of past high-level policy experience in order to make the observed action sequence more likely with the current low-level policy

- In the past: $a_{t:t+c-1} \sim \mu^{10}(s_{t:t+c-1}, g_{t:t+c-1})$

- Solution:
  - Find $\tilde{g}_t$ to maximize $\mu^{10}(a_{t:t+c-1}|s_{t:t+c-1}, \tilde{g}_{t:t+c-1})$ for the current $\mu^{10}$
  - Relabel the high-level experience with $\tilde{g}_t$: $(s_t, \tilde{g}_t, \sum R_{t:t+c-1}, s_{t+c})$

- \[
\log \left( \mu^{10}(s_{t:t+c-1}, \tilde{g}_{t:t+c-1}) \right) \propto -\frac{1}{2} \sum_{i=t}^{t+c-1} \left\| a_i - \mu^{10}(s_i, \tilde{g}_i) \right\|_2^2 + \text{constant}
\]

- To approximately maximize this, 10 candidate goals are chosen:
  - 8 candidate goals are sampled from a Gaussian centered around $s_{t+c} - s_t$
  - Additionally, the original $g_t$ and $s_{t+c} - s_t$ are candidates as well
RELATED WORK
To help learn useful lower-level policies, some recent work uses auxiliary rewards

- Either hand-crafted rewards or exploration-encouraging rewards
- HIRO uses a parameterized reward function

To produce semantically distinct behavior, some recent work pretrain the lower-level policy on diverse tasks

- This requires suitably similar tasks and is not general

Hierarchical Actor-Critic uses off-policy training, but without the correction

FeUdal Networks also use goals and parameterized lower-level rewards

- Goals and rewards are computed in terms of a learned state representation, not directly
- HIRO uses raw goal and state representations, so it can immediately train on intrinsic rewards
EXPERIMENTS
## Comparative Analysis

<table>
<thead>
<tr>
<th></th>
<th>Ant Gather</th>
<th>Ant Maze</th>
<th>Ant Push</th>
<th>Ant Fall</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRO</td>
<td>3.02±1.49</td>
<td>0.99±0.01</td>
<td>0.92±0.04</td>
<td>0.66±0.07</td>
</tr>
<tr>
<td>FuN representation</td>
<td>0.03 ± 0.01</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>FuN transition PG</td>
<td>0.41 ± 0.06</td>
<td>0.0 ± 0.0</td>
<td>0.56 ± 0.39</td>
<td>0.01 ± 0.02</td>
</tr>
<tr>
<td>FuN cos similarity</td>
<td>0.85 ± 1.17</td>
<td>0.16 ± 0.33</td>
<td>0.06 ± 0.17</td>
<td>0.07 ± 0.22</td>
</tr>
<tr>
<td>FuN</td>
<td>0.01 ± 0.01</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>SNN4HRL</td>
<td>1.92 ± 0.52</td>
<td>0.0 ± 0.0</td>
<td>0.02 ± 0.01</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>VIME</td>
<td>1.42 ± 0.90</td>
<td>0.0 ± 0.0</td>
<td>0.02 ± 0.02</td>
<td>0.0 ± 0.0</td>
</tr>
</tbody>
</table>
Ablative Analysis

Ant Gather  Ant Maze  Ant Push  Ant Fall

- HIRO
- With lower-level re-labelling
- With pre-training
- No off-policy correction
- No HRL
CONCLUSION
Summary of Main Contributions

- A general approach for training a two-layer HRL algorithm
- Goals specified in terms of a difference between desired state and current state
- Lower-level policy is trained with parameterized rewards
- Both policies are trained concurrently in an off-policy manner
  - Leads to high sample-efficiency
- Off-policy correction allows for the use of past experience for training the higher-level policy
Future Work

- The algorithm was evaluated on fairly simple tasks
  - State and action spaces were both low-dimensional
  - Environment was fully-observed

- Further work could be done to apply this algorithm or an improved version to harder tasks