Posterior Goal Sampling for Hierarchical Reinforcement Learning

Ilies Ghanzouri
Department of Mechanical Engineering
Stanford University
Stanford, CA 94305
ghanz@stanford.edu

Abstract

Efficient exploration is a critical topic in Reinforcement Learning. Having the RL agent to commit to solutions too quickly without enough exploration may lead to local minima or total failure. Hierarchical reinforcement learning decomposes a reinforcement learning problem into a hierarchy of subtasks such that higher-level parent invoke lower-level child tasks as if they were primitive actions.
The goal of this project is to design a reinforcement learning algorithm that is able to efficiently learn hierarchical policies in hard exploration problems. In hard exploration problems, we must apply deep exploration in order to maximize future rewards and not only rely on immediate gains. We consider the case of hierarchical problems that involve two levels, a high-level policy (over goals) and a low-level policy (over actions).
A large body of work exists on provably-efficient exploration in RL [1, 2] which typically requires that an agent maintain a posterior distribution over optimal policies/value functions, rather than a single point estimate. Feudal RL methods [3–5] maintain two point estimates for a high-level policy (over goals) and a low-level policy (over actions). While the latter can be trained quite efficiently, the high-level policy may still fail to learn in hard exploration problems.
In this project, we aim to improve deep exploration of the high level-policy learning. Specifically, based on the h-DQN algorithm [5], our approach [1] uses a hierarchical-Ensemble DQN in order to enhance deep exploration for the higher-level policy on the Long Corridor environment. Our findings showcases the limits of ensemble DQN and suggest to improve on it with a Bootstrapped DQN method. The paper [1] showed promising deep exploration results as it showed that it can achieve deep exploration better than Ensemble DQN.

[1] Find code at https://github.com/ghanzouri/CS330-Posterior-Goal-Sampling-for-Hierarchical-Reinforcement-Learning

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

1 Introduction

Deep Reinforcement Learning (RL) is an area of machine learning where an agent learns how to behave in an environment by performing an action and seeking high rewards. It adopts large neural network policies and value functions for robust generalization capacity to deal with high dimensional complex tasks. Large amounts exploration of the environment is required to find the optimal state action value $Q(s, a)$ and efficient exploration remains a major challenge for reinforcement learning. Moreover, RL has recently become a topic of interest due to its ability to solve problems without knowledge of the underlying dynamics. Deep RL has achieved great successes in a wide range of challenging problems. Model free approaches are used over a large set of applications including control theory, natural language processing, image recognition, and medical diagnoses.

Hierarchical Reinforcement Learning (HRL) extends traditional RL methods to solve more complex tasks by establishing a hierarchy of subtasks between multiple layers of policies. Its modeling stands on a technique that defines and plans over temporally abstracted macro-actions in order to achieve delayed rewards over long time horizons.
The HIRO paper [4] proposes a two-level hierarchical policy to solve complex tasks. The higher-level policy takes in states observations from the environment and produces high-level actions (goals). The lower-level controller tracks the goal and state observation while being supervised by the higher-level controller. The lower-level policy produces an atomic action and directly applies it to the environment. While the environment reward will be used by the higher level policy for training, the lower level policy will be rewarded for satisfying the higher-level policy goals. Moreover, it utilizes off-policy corrections samples for higher-level training using fewer environment interactions, which re-labels an experience in the past with a high-level action chosen to maximize the probability of the past lower-level actions. Previous methods required careful task-specific design and on-policy training which is difficult to apply in real world cases scenarios. Furthermore, HIRO [4] maintains two point estimates for a high-level policy (over goals) and a low-level policy (over actions) through Deep Deterministic Policy Gradient (DDPG) algorithms, a model-free off-policy algorithms, which has been successful for state-of-the-art performance in high-dimensional continuous control tasks.
Similarly to HIRO algorithm [4], Hierarchical-DQN [5] involves a top-level value function that learns a policy over intrinsic goals, and a lower-level function learns a policy over atomic actions to satisfy the given goals. It maintains point estimates the higher and lower level polices through Deep Q-Networks (DQN). DQN is another model-free reinforcement learning and has recently enabled the successful application of reinforcement learning techniques in new environments of complex state. A particularly notable success of DQN has been its application to a library of Atari games, where it rivaled or surpassed human professional benchmark scores. h-DQN [5] builds on top of DQN to tackle reinforcement learning problem with delayed reward structure by proposing a hierarchical deep reinforcement learning algorithm.

Efficient exploration over intractably large state spaces remains a major challenge for reinforcement learning. The latter two hierarchical RL methods [4, 5] maintain two point estimates (through DDPG and DQN architectures respectively) for a high-level policy (over goals) and a low-level policy (over actions). While the lower-level policy can be trained quite efficiently, the high-level policy may still fail to learn in hard exploration problems. A substantial body of work exists on provably-efficient exploration in RL [1, 2] which typically requires that an agent maintain a posterior distribution over optimal policies/value functions, rather than a single point estimate. Bootstrapped DQN [1] allows deep exploration with deep neural networks. It modifies DQN to approximate a distribution over Q-values via the bootstrap. At the start of each episode, bootstrapped DQN samples a single Q-value function from its approximate posterior. The agent then follows the policy which is optimal for that sample for the duration of the episode. The authors proposes an architecture that has a shared network and K bootstrap heads. Each bootstrap head trains on a different buffer E_i, where each E_i has been constructed by sampling n data points from the original experience buffer E with replacement. Also, along with (s_t, a_t, r_t, s_{t+1}), they push a bootstrap mask m_t into the replay buffer, that filters specific training data points. Masks are created by sampling from a masking distribution. The authors used a Bernoulli(p) distribution. If the mask is $(1, 1 \cdots 1)$, then this becomes an ensemble Q learning method.

In this project, based on the Bootstrapped DQN method [1], we implement an ensemble Q learning method on the hierarchical-DQN (h-DQN) [5] to improve deep exploration of the high level-policy.

We replicate the original paper's experiments on a discrete stochastic decision process (Long Corridor environment) and present an analysis of our implementation.

2 Method

We compare three reinforcement learning algorithms: Deep Q-Networks (DQN), hierarchical-DQN (hDQN), and hierarchical-Ensemble DQN (our implementation).

2.1 DQN

Q-learning operates by applying incremental estimation to the Bellman equation :

$$Q(s,a) = Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big)$$

where s, a, r are the state, action, and reward observed in a single example, α is the learning rate, and γ is a discount factor.
DQN is an extension of Q-learning which trains a deep neural network to predict Q-values from states. By training a neural network model instead of tabulating Q values explicitly, DQN enables the generalization of value observations to update the Q-value estimates of states with similar features. This makes DQN viable for application to environments with high-dimensional or continuous state.

2.2 Hierarchical DQN (h-DQN)

The paper [5] introduced h-DQN as a hierarchical extension to DQN. h-DQN makes decisions at two hierarchical levels: a meta-controller which selects a goal based on the current state, and a controller which selects primitive actions based on the current state and selected goal until either the goal or a terminal state is reached. An intrinsic reward is provided to the controller if the goal is reached successfully. Both the controller and meta-controller learn from the outcomes of their decisions using DQN; the meta-controller learns goal values based on the extrinsic reward obtained from goal selection, while the controller learns action values from the intrinsic reward obtained by selecting a particular action given the current state and goal.

2.3 Hierarchical-Ensemble DQN

We implement a version of the h-DQN algorithm where the higher-level DQN is replaced by an ensemble DQN value update. The higher-level policy learns from an ensemble of multi-head Q functions. The latter are initialized randomly. During training, each head is trained individually. After training, we sample from one of the heads and produces a high-level actions (goal). Then, the lower-level policy takes in goal and state observation and produces an action and applies it to the environment. The environment reward is taken by the high-level policy for training while the lower-level policy is rewarded when satisfying the high-level goal.

3 Experiments

3.1 Ant Maze

3.1.1 Results

First of all, I made experiments on the HIRO [4] algorithm on the Ant Maze environment. We replicated the results of the paper. The paper originally used 10 million steps and we trained ours with 2,5 millions steps with similar hyperparameters. We achieve similar performance around 0.8 average success rate in Figure 1. After having some difficulties on figuring where the high policy was being trained on the Tensorflow open source code (here) , I have shifted my work towards developing our approach on the h-DQN algorithm.

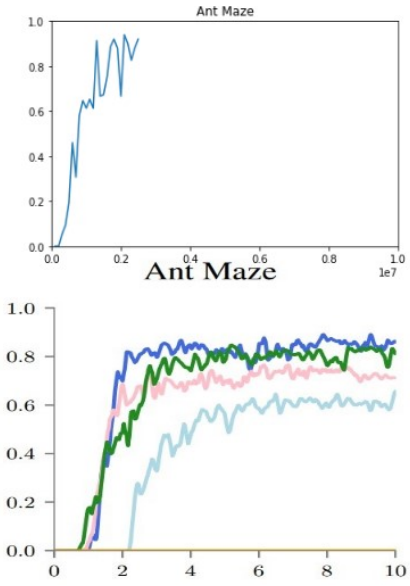

Figure 1: Average success rate vs environment steps for the Ant Maze with x-axis in millions of environment steps. Our results is on the top, and HIRO's results at the bottom

3.2 Discrete Stochastic Decision Process, Long Corridor environment

3.2.1 Results

The paper h-DQN [5] evaluates hierarchical-DQN by comparing it to a DQN baseline in a stochastic decision process with six states (Figure 2).

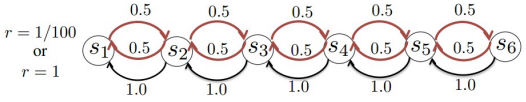

Figure 2: A stochastic decision process where the reward at the terminal state s_1 depends on whether s_6 is visited ($r = 1$) or not ($r = 1/100$)

We replicated the results of the h-DQN paper [5] on Figure 3 by training DQN and hierarchical h-DQN for $10,000$ of steps, a learning rate $\alpha = 0.01$ and 3 test runs per agent. The paper originally used $50,000$ of steps, a learning rate $\alpha = 0.00025$ and 10 test runs per agent. We compared the average reward (Figure 3) for DQN and h-DQN in this environment. We observed similar results to those of [5] with a significant increase of average reward with h-DQN implementation.

In addition to replicating the results of the paper [5], we evaluate our hierarchical ensemble DQN implementation in Figure 4. We design our ensemble DQN with 5 Q-heads functions and we train it with a learning rate $\alpha = 0.01$ and 3 test runs per agent for $10,000$ steps. As we can observe in Figure 4, our current ensemble DQN is unable to significantly improve on the standard h-DQN. We see that the average score of our implementation is lower that the h-DQN score. One way to improve our approach is to further experiment by tweaking hyperparameters such as the number of Q-heads functions or the total number of training steps. An alternative is to adopt a bootstrapped DQN architecture (further discussed in Section 4) instead of the ensemble DQN for the high-level policy learning.

4 Discussion and Conclusions

We verified that the procedure of h-DQN algorithm [5] can be replicated with very similar results (Figure 3). We then presented our implementation of ensemble DQN for higher-level policy learning

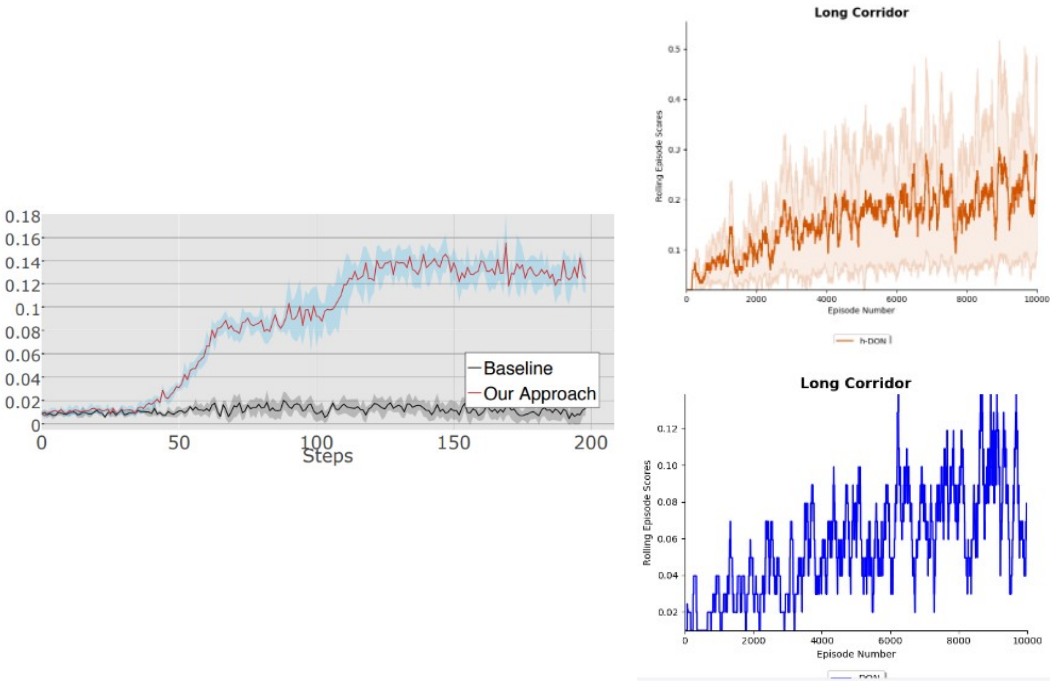

Figure 3: Average reward vs number of steps. Original paper results on the left (h-DQN in red and DQN in black) and ours on the right (h-DQN in orange and DQN in blue).

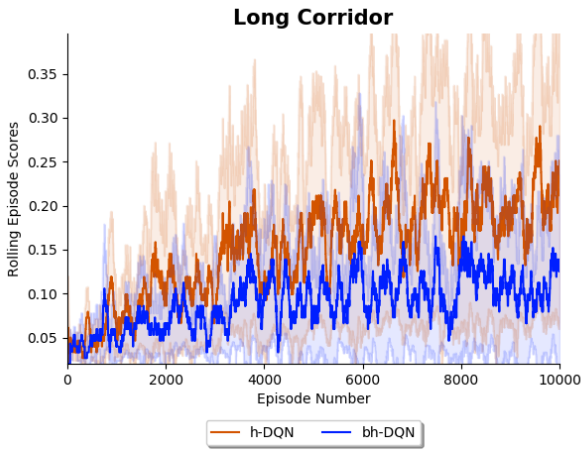

Figure 4: Implementation of h-Ensemble DQN (in blue) compared to standard h-DQN

which is unable to significantly ameliorate on h-DQN with our current hyperparameters. An alternative for efficient deep exploration is the usage of Boostrapped DQN. The latter showed promising deep exploration results in the paper [1]. The authors consider a simple example (Figure 5) to demonstrate the effectiveness of bootstrapped DQN at deep exploration.

In this example (Figure 5), the agent starts at s_2. There are N steps, and $N + 9$ timesteps to generate the experience buffer. The agent is said to have learned the optimal policy if it achieves the best possible reward of 10 (go to the rightmost state in $N - 1$ timesteps, then stay there for 10 timesteps), for at least 100 such episodes.

The results are presented in Figure 6. The blue dots indicate when the agent learned the optimal policy. A red dot shows that it took more than 2000 episodes. What these graphs showcases is that

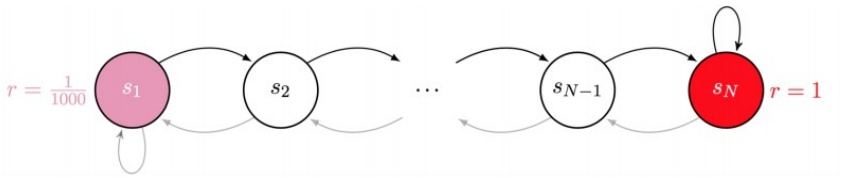

Figure 5: Source: [1], section 5.1: Scalable environments that requires deep exploration.

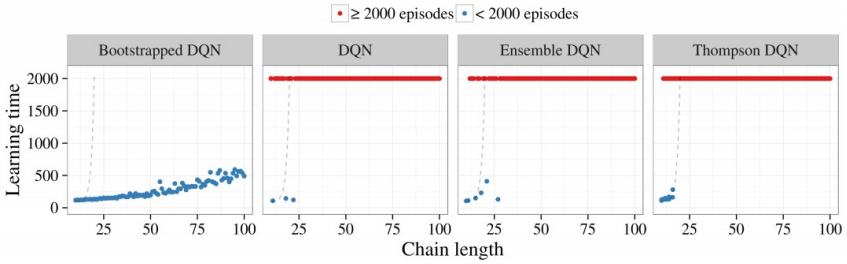

Figure 6: Source: [1], section 5.1: Only Bootstrapped DQN demonstrates deep exploration.

bootstrapped DQN can achieve deep exploration better than Ensemble DQN. Thus, future work should implement a Bootstrapped Hierarchical RL for high-level policy learning.

References

[1] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 4026–4034. Curran Associates, Inc., 2016.

[2] Ian Osband, Benjamin Van Roy, Daniel Russo, and Zheng Wen. Deep exploration via randomized value functions, 2019.

[3] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 271–278. Morgan-Kaufmann, 1993.

[4] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning, 2018.

[5] Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, 2016.