Sparsity, Observability, and Temporal Abstraction in Hierarchical Deep Q-Learning

Gabriel B Margolis [1]

Abstract

Hierarchical-DQN (h-DQN) was introduced by Kulkarni et al. (2016) to address the challenge of learning hierarchical value functions across different temporal scales. We replicate the original paper's experiments on a discrete stochastic decision process and present an analysis of this environment and its fully observable variant. Our results verify that the original paper's results are repeatable, and our analysis provides insight on why the paper's discrete stochastic decision process is a useful example.

1. Introduction

Deep Q-Networks (DQN) have 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 across many differently-structured games without significant domain-specific modification (Mnih et al., 2015).

Although DQN works well for a large number of Atari 2600 games, it fails miserably on others. Notably, DQN achieves a score of zero on Montezuma's Revenge, an Atari game with a sparse, delayed reward structure unlike the dense reward distribution of many other games on the platform. Kulkarni et al. (2016) addressed this shortcoming by proposing a hierarchical deep reinforcement learning algorithm, h-DQN. Hierarchical modeling is a techinique which defines and plans over temporally abstracted macro-actions in order to achieve delayed rewards over long time horizons.

In this paper, we implement the h-DQN algorithm of Kulkarni et al. and demonstrate its success in environments of sparse, delayed reward. Although our constrained computational resources limit us from training h-DQN on Montezuma's Revenge, a demonstration which took millions of training iterations in the original paper, we are able to replicate the results the original paper obtained in a small discrete stochastic decision process environment.

[1]Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Correspondence to: Gabriel B Margolis <gmargo@mit.edu>.

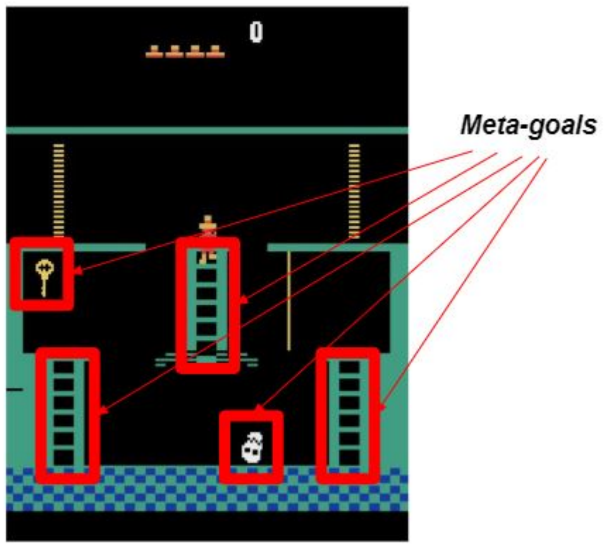

Figure 1. Montezuma's Revenge, an Atari game with sparse, delayed reward. h-DQN outperforms DQN in this environment by identifying meta-goals (red) and a macro-actions which produce intrinsic reward for reaching each meta-goal.

2. Method

We implement four reinforcement learning algorithms: Q-learning, Deep Q-Networks (DQN), hierarchical-DQN (h-DQN), and hierarchical Q-learning.

2.1. Q-learning

Q-learning is an off-policy, value-based reinforcement learning algorithm. In Q-learning, a tabular list of state-action pair value estimates is maintained and updated with a new experience $\langle s, a, r, s' \rangle$ according to the equation:

$$Q[s,a] \leftarrow (1-\alpha)Q[s,a] + \alpha(r + \gamma \max_{a'} Q[s', a'])$$

Exploration in Q-learning is performed by random action selection, while exploitation is performed by selecting the available action with maximum Q-value given the current state. We balance exploration and exploitation using an epsilon-greedy strategy, exploring with a probability ϵ which is annealed from 1.0 to 0.1 over the course of training.

2.2. DQN

DQN (Mnih et al., 2015) 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.

To train a neural network for DQN, the network's loss function is defined as:

$$L = E_{(s,a,r,s')}[(r + \gamma \max_{a'} Q(s', a') - Q(s,a))^2]$$

The learning rate of the neural network serves the same function in DQN that α serves in Q-learning. As in Q-learning, we employ an epsilon-greedy strategy in our implementation of DQN to collect experiences.

2.3. Hierarchical DQN (h-DQN)

Kulkarni et al. (2016) introduced h-DQN as a hierarchical extension to DQN, in order to address the poor performance of DQN in domains of sparse, delayed reward such as Montezuma's Revenge. 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 by a critic 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.

2.4. Hierarchical Q-learning

To simply demonstrate the benefits of h-DQN's hierarchical architecture, we implement a version of the h-DQN algorithm where the DQN parameter update is replaced by a Q-value update, as in Q-learning. We refer to this algorithm as hierarchical Q-learning.

Algorithm 1 h-DQN

```
 1: Initialize experience replay memories $\{D_1, D_2\}$ and
    parameters $\{\Theta_1, \Theta_2\}$ for the controller and meta-
    controller respectively.
 2: Initialize exploration probability $\epsilon_{1,g} = 1$ for the con-
    troller for all goals $g$ and $\epsilon_2 = 1$ for the meta-controller.
 3: for $i = 1$, num_episodes do
 4:   Initialize environment and get state description $s$
 5:   $g \leftarrow$ epsGreedy$(s, G, \epsilon_2, Q_2)$
 6:   while $s$ is not terminal do
 7:     $F \leftarrow 0$
 8:     $s_0 \leftarrow s$
 9:     while not ($s$ is terminal or goal $g$ reached) do
10:       a $\leftarrow$ epsGreedy$(s, g, A, \varepsilon_{1,g}, Q_1)$
11:       Execute $a$ and obtain next state $s'$ and extrinsic
          reward $f$ from environment
12:       Obtain intrinsic reward $r(s, a, s')$ from internal
          critic
13:       Store transition $(s, g, a, r, s', g)$ in $D_1$
14:       updateParams$(L_1(\Theta_{1,i}), D_1)$
15:       updateParams$(L_2(\Theta_{2,i}), D_2)$
16:       $F \leftarrow F + f$
17:       $s \leftarrow s'$
18:     end while
19:     Store transition $(s_0, g, F, s')$ in $D_2$
20:     if s is not terminal then
21:       $g \leftarrow$ epsGreedy$(s, G, \epsilon_2, Q_2)$
22:     end if
23:   end while
24:   Anneal $\epsilon_2$ and $\epsilon_1$
25: end for
```

3. Experiments

3.1. Discrete Stochastic Decision Process

3.1.1. Results

The paper by Kulkarni et al. evaluates hierarchical Q-learning by comparing it to a non-hierarchical Q-learning baseline in a stochastic decision process with six states (Figure 2).

One would not expect deep reinforcement learning to provide an advantage over Q-learning in this six-state stochastic decision process. DQN offers benefits in complex, highly featured environments such as video games, where the value of a state may generalize to states with similar features as identified by a neural network. In the discrete stochastic decision process environment, where states are practically featureless, one would not expect to attain a benefit from DQN over ordinary Q-learning. Indeed, in this first experiment, Kulkarni et al. sensibly forgo neural networks and strictly evaluate the effect of their hierarchical approach on the performance of old fashioned Q-learning.

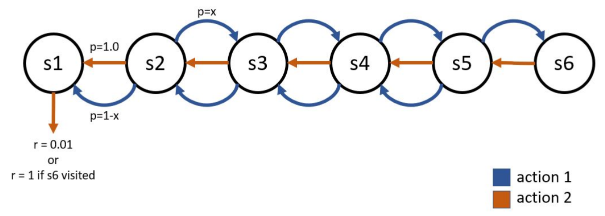

Figure 2. A stochastic decision process where the reward at the terminal state s_1 depends on whether s_6 is visited ($r = 1$) or not ($r = 1/100$)

We replicated the results of Kulkarni et al. in this discrete stochastic decision process environment over 20,000 episodes. We compared the episode reward (Figure 3) and state visitation rate (Figure 4) for Q-learning and hierarchical Q-learning in this environment. We observed similar results to those of Kulkarni et al., with some caveats. The average extrinsic reward obtained by our hierarchical Q-learning implementation was 0.10, as compared to the 0.13 value claimed in Kuklarni et al. We also observed that as ϵ was annealed in non-hierarchical Q-learning, the variance in the reward of Q-learning became small, while in the original paper this variance remained high. Both of these differences may have been the result of an adaptive annealing strategy that was mentioned but not detailed in the original paper. The full details of our implementation including hyperparameters are available in Appendix A.

In addition to replicating the results of Kulkarni et al., we sought to evaluate the return of pure exploitation based on the learned Q-values for each algorithm in the discrete stochastic decision process. After 20,000 steps, we dropped all ϵ terms from 0.1 to 0.0 and evaluated the exploitative return for 5000 additional episodes. We found that hierarchical Q-learning achieved an average reward of 0.20 during this exploitation phase, while DQN achieved a return of 0.01. In section 3.2.2, we show that hierarchical Q-learning's exploitative reward is equal to the reward obtained by the optimal strategy.

3.1.2. Analysis

Why is the stochastic decision process environment in Figure 2 particularly challenging for an agent performing non-hierarchical Q-learning? Kulkarni et al. assert throughout their paper that this environment, along with Montezuma's revenge, is characterized by "long-range delayed feedback," and that hierarchical approaches confer benefits under such a condition. Upon inspection, however, reward sparsity alone is not the property of this environment that causes hierarchical Q-learning to outperform Q-learning. Rather, the partial

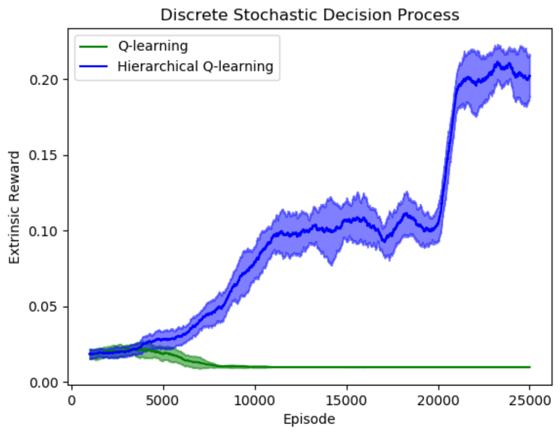

Figure 3. Episode reward in the discrete stochastic decision process environment. In this environment, hierarchical q-learning learns to reach the higher reward $r = 1$ by visiting s_6, while non-hierarchical q-learning converges to a policy that achieves only the lower reward $r = 1/100$. The mean and standard deviation of 20 trials are shown, smoothed over 1000 epochs. After 20000 episodes, exploration rates were set to 0 and the expected return of pure exploitation was evaluated for 5000 additional episodes.

observability and hierarchical structure of this environment allow hierarchical Q-learning to solve a much less sparse problem than Q-learning by estimating the hidden state of the system. A hierarchical agent that believes s_6 has already been visited (by virtue of having just selected it as a goal) can choose to quickly transition back to the terminal state under this condition, while an agent that cannot observe the visitation status of s_6 must choose the same move in every state under a deterministic strategy, and will take a much longer walk between visiting the sixth state and terminating.

In order to illustrate this point, in section 3.2 we convert the environment to an equivalent fully observable MDP (Figure 5) and again evaluate the performance of hierarchical and non-hierarchical Q-learning. This fully observable MDP has the same states, actions, and transition probabilities as our discrete stochastic decision process, but the agent can now observe whether s_6 has been visited.

3.2. Fully Observable Discrete Stochastic Decision Process

3.2.1. Results

We evaluate Q-learning and hierarchical Q-learning in our fully observable MDP environment (Figure 5). Each algorithm learns for 20,000 episodes, and its reward is then evaluated under purely exploitative behavior for 5,000 additional episodes. Implementation details for each algorithm

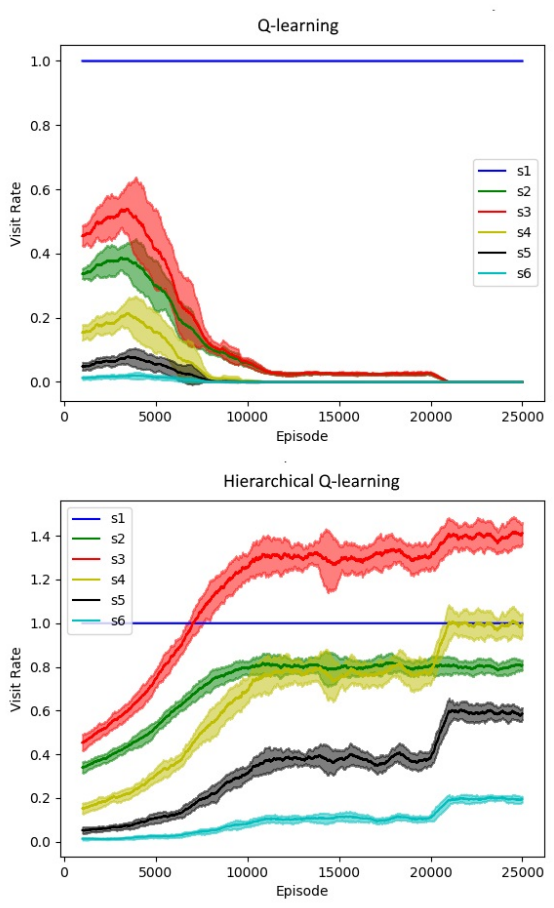

Figure 4. Rate at which states are visited by each learning algorithm in the discrete stochastic decision process. Mean and standard deviation of 20 trials, smoothed over 1000 episodes, are plotted.

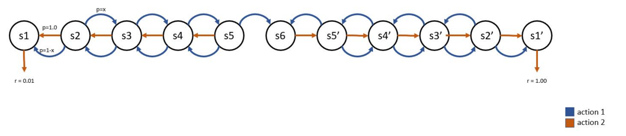

Figure 5. A stochastic MDP where information about whether s_6 has been visited is fully observable. The two terminal states are s_1 ($r = 1/100$) and s'_1 ($r = 1$). The states, actions, and transition probabilities of this MDP are identical to those of the decision process in Figure 1.

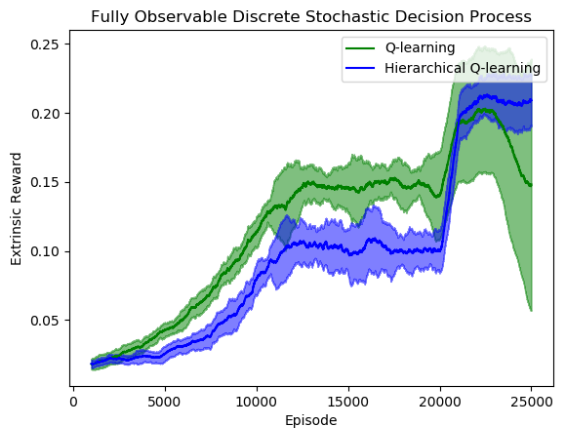

Figure 6. Episode reward in the fully observable MDP environment. In this environment, both agents learn to obtain the optimal reward in the best case, but the variance of the exploitative average reward obtained by Q-learning is much higher. The mean and standard deviation of 20 trials are shown, smoothed over 1000 epochs. After 20000 episodes, exploration rates were set to 0 and the expected return of pure exploitation was evaluated for 5000 additional episodes.

are provided in appendix A.

Figure 6 shows the reward of each algorithm in the fully observable MDP environment. We observe that both Q-learning and hierarchical Q-learning match the average reward of the optimal policy (0.21) in the best case. However, the variance of the exploitative returns of Q-learning in episodes 20000-25000 is greater than that of hierarchical Q-learning, and mean reward for Q-learning decreases dramatically during exploitation, indicating that Q-learning was using the stochasticity of epsilon-greedy action selection as a key element in its learned policy.

3.2.2. Analysis

In this MDP environment, we can analytically evaluate the expected payoff of an optimal strategy. Since action 1 is the only action to be taken in $s_1 ... s_5$ with a nonzero probability of advancing closer to s'_1, the location of maximum reward, we see that the optimal strategy for these states is to take action 1. If s_6 is reached, there is no transition back to s_5 since we cannot 'un-visit' s_6, and as a result an agent that reaches s_6 is guaranteed an eventual reward of 1. Thus, we only need to evaluate the probability of the agent reaching s_6 following the optimal strategy of only choosing action 1. We can write this as a Markov chain with transition matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Solving for the steady-state distribution with initial state $x_i = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$, we find that there is an 0.2

probability of reaching s_6 under this strategy and a 0.8 probability of terminating in s_1. So, the expected reward of the optimal strategy in both the MDP and the discrete stochastic decision process environment is 0.208. To verify this, we run a Monte-Carlo simulation in each environment with an expert who only moves right. Over 100,000 simulations, we obtained average rewards of 0.208 ± 0.0003 in both environments.

4. Discussion and Conclusions

We verified that the procedure of Kulkarni et al.'s Experiment 1 can be replicated with very similar results. Although some aspects of the original paper's description of their hierarchical Q-learning implementation were confusing or self-contradicting (see Appendix B for more discussion), it was useful that the authors took the time to provide a simple environment, the discrete stochastic decision process, which can demonstrate h-DQN's fundamental behavior with limited computation.

We then presented an analysis of a fully observable variant on the environment in Experiment 1 of Kulkarni et al. From our evaluation of Q-learning and hierarchical Q-learning in this fully observable MDP, we conclude that hierarchical Q-learning achieves an optimal policy in the original paper's experiment, and also that when compared to Q-learning in an environment of equal sparsity, hierarchical Q-learning learns the optimal policy more consistently than Hierarchical Q-learning.

In their original h-DQN paper, Kulkarni et al. demonstrated that hierarchical reinforcement learning is a powerful method for planning in domains of sparse reward, particularly when a useful temporal abstraction of a problem can be made and the resulting hierarchical structure can be exploited. This paper's presentation of a simple discrete stochastic decision process example was highly illustrative of its algorithm's power. Future work in the field of reinforcement learning should strive to provide such simple environments which demonstrate the power of reinforcement learning algorithms, and also to thoroughly elaborate on why the environments they present are interesting.

References

Kulkarni, T., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.

Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–541, 2015.

A. Implementation Details

The full code for our implementation of h-DQN is available on GitHub: `https://github.com/gmargo11/hDQN`.

A.1. Discrete Stochastic Decision Process Experiment

In the discrete stochastic decision process example, we trained Q-learning and hierarchical Q-learning for 20,000 episodes each. The decision process we used for this experiment had states, actions, transitions, and rewards identical to those of Kulkarni et al.'s Experiment 1. We performed 20 repeated trials of each algorithm with different random seeds. We then evaluated each algorithm for an additional 5,000 episodes under pure exploitation, with all ϵ values set to zero.

A.2. Fully Observable MDP Experiment

In the fully observable MDP example, we trained Q-learning and hierarchical Q-learning for 20,000 episodes each. The decision process we used for this experiment had actions, transitions, and rewards identical to those of Kulkarni et al.'s Experiment 1, but each state $s_i, i \in [1, 5]$ was supplemented with an additional state s'_i representing the state where s_6 has already been visited. We performed 20 repeated trials of each algorithm with different random seeds. We then evaluated each algorithm for an additional 5,000 episodes under pure exploitation, with all ϵ values set to zero.

A.3. Hyperparameters

Hyperparameters were tuned by hand and were fixed across all experiments.

Q-learning Our hyperparameters for Q-learning were $\alpha = 0.6, \gamma = 0.9$. We annealed ϵ, the exploration rate, from 1.0 to 0.1 by a constant subtracted step of $1/12000$ each episode.

Hierarchical Q-learning Our hyperparameters for Hierarchical Q-learining were $\alpha = 0.6, \gamma = 0.9$ for both the controller and meta-controller. We annealed ϵ_2, the meta-controller exploration rate, from 1.0 to 0.1 by a constant subtracted step of $1/12000$ each episode. We annealed $\epsilon_{1,g}$, the exploration rate for the controller when the goal is set to g, by a constant subtracted step of $1/2000$ each time the goal g was chosen. This method of annealing $\epsilon_{1,g}$ departed from the method mentioned in Kulkarni et al., who stated that they adaptively annealed $\epsilon_{1,g}$ based on the average success rate of reaching the goal.

B. Ambiguity in the Original Paper

Kulkarni et al. (2016) described their results in the stochastic decision process environment as follows: ”We compare the performance of our approach (without the deep neural networks) with Q-Learning as a baseline (without intrinsic rewards) in terms of the average extrinsic reward gained in an episode. In our experiments, all ϵ parameters are annealed from 1 to 0.1 over 50,000 steps. The learning rate is set to 0.00025. Figure 3 plots the evolution of reward for both methods averaged over 10 different runs. As expected, we see that Q-Learning is unable to find the optimal policy even after 200 epochs, converging to a sub-optimal policy of reaching state s1 directly to obtain a reward of 0.01. In contrast, our approach with hierarchical Q-estimators learns to choose goals s4, s5 or s6, which statistically lead the agent to visit s6 before going back to s1. Therefore, the agent obtains a significantly higher average reward of around 0.13.”

This summary is somewhat self-contradictory. The note that the learning rate was set to 0.00025 is incompatible with the assertion that Q-learning, which does not have a learning rate but an alpha parameter, was used instead of DQN, which has a learning rate. Training is stated to take place over 200 epochs, but the length of an epoch is not defined. An associated plot in Figure 3 of Kulkarni et al. shows 200 ”steps”, assumed to mean epochs. Meanwhile another associated plot in Figure 4 of Kulkarni et al. shows 12,000 episodes, rather than the 50,000 claimed in the summary of results. Because these implementation and evaluation details were unclear, we tuned our hyperparameters ourselves to replicate this paper’s results.