Progress Part 1

Yiheng Lin, Zhihao Jiang

Introduction

It is widely accepted that h-DQN (Hierarchy Deep Q-Learning) is developed to tackle the efficiency problem of $\epsilon-$greedy Q-Learning when the reward is sparce. However, after we study the problem setting and reimplement the method proposed in (ref 1), we find another big advantage of h-DQN over traditional Q-Learning and DQN (Deep Q-Learning) (ref 2) is that h-DQN can potentially discover the hidden state which is not observable by the agent but is crucial for receiving reward. To make our idea more clear, let us consider the following game (proposed in (1)):

Game Setting

The state set is $S = \{1, 2, \cdots, 6\}$ and the action set is $A = \{1, 2\}$. The agent start at state 2 and state 1 is the terminal state. At state i, if the agent takes action 1, it will go to state $i - 1$ with probability 1; if the agent takes action 2, it will go to state $i + 1$ with probability 0.5;otherwise, it will go to state $i - 1$. The reward is received at state 1: if the agent has been to state 6, it will receive reward 1; otherwise, it will receive reward 0.01.

The authors of (ref 1) claim that traditional Q-Learning cannot learn to take action 2 in order to move towards state 6 even after a long period of training (200 epochs) and converges to a sub-optimal policy. Our experiment shows the same result.

Actually, despite the length of the path of the optimal policy and the probability setting on state transition, we think the major difficulty in this game is the hidden state of whether the agent has reached state 6. At state i ($2 \leq i \leq 5$), when state 6 has not been reached, the agent should take action 2 in order to reach 6, which means $Q(i, 2) > Q(i, 1)$; when state 6 has already been reached, the agent should take action 1, which means $Q(i, 2) < Q(i, 1)$ (Although keep taking action 2 is a possible optimal policy, we think it is almost impossible for the agent to keep taking action 2 after state 6 has been reached, since it cannot see the difference in the final reward.). In traditional Q-Learning and DQN, QValue function should converge as learning proceed, thus it is difficult for them to learn the optimal policy even unlimited time is given.

However, if the hidden state is given to tell the agent whether it has reached state 6, (for example, 0 represents it has not visited state 6, 1 represents it has visited state 6) then the problem is converted to a MDP of 12 states. Then at state i, the agent can seperately learn $Q((i, 0), 2) > Q((i, 0), 1)$ and $Q((i, 1), 2) < Q((i, 1), 1)$. We think this will make the problem much easier for traditional Q-Learning. We will do more experiments to verify this claim. The subgoal in (ref 1) operates similarly, the meta-controller should first set the subgoal to state 6 and then set the subgoal to state 1 after state 6 is visited.

H-DQN Algorithm

We carefully studied algorithm 1 proposed by (ref 1) (please find the pseudocode in appendix 1) and reimplement this algorithm on the game setting described in the last section based on (ref 3, github). We can see salient improvement compared with traditional Q-Learning method, although the performance is worse than the authors of (ref 1) claimed. After supervising the training process, we think there are some potential improvements we can do to judge the effectiveness of h-DQN algorithm.

As for the performance metric, the authors of (ref 1) uses the number of visits to each state during 1000 runs to judge the agents tendency of taking action 2. However, a lot of visits to state 6 is resulted by keep taking action 2 after arriving at state 6. So the agents can visit state 6 multiple times in one run.

Theorem 1. *If the optimal policy is chosen (i.e. keep taking action 2 until state 6 is arrived), the probability of arriving at state 6 is $\frac{1}{5}$.*

Proof. Since we only consider whether state 6 can be reached, in the proof, we can let state 6 be a terminal state. Let x_i be the probability of arriving state 6 from state i ($1 \leq i \leq 6$), then we have $x_1 = 0, x_6 = 1$, and

$$x_i = 0.5 * x_{i-1} + 0.5 * x_{i+1}, (2 \leq i \leq 5) \tag{1}$$

Solving the equations and we can get $x_2 = \frac{1}{5}$.

Notice that if we adopt $\epsilon-$greedy policy, $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

Acknowledgement: