Accelerated Q-Learning for Fail State and Action Spaces

In-Won Park and Jong-Hwan Kim
Department of Electrical Engineering & Computer Science
KAIST
Daejeon, Republic of Korea
{iwpark, johkim}@rit.kaist.ac.kr

Kui-Hong Park
Advanced Technology Research TFT
Korea Telecom (KT)
Seoul, Republic of Korea
kidonge@kt.co.kr

***Abstract*—Accelerated Q-learning algorithm is proposed for environment having both goal and fail states. It extends Q-learning, a well-known scheme in reinforcement learning. Unlike this conventional Q-learning, the proposed algorithm keeps track of the past failure experiences as a separate fail state-action value, Q_F. Agent uses this value along with a goal state-action value, Q_N, which is calculated and updated using conventional Q-learning, to modify the exploratory behavior during learning phase. Effectiveness of the proposed accelerated Q-learning algorithm is verified in a grid world environment. The proposed algorithm significantly reduces a convergence speed to find out the optimal path from start state to goal state while maximizing its receiving rewards.**

***Index Terms*—Accelerated Q-learning algorithm, success and failure experiences of the agent.**

I. Introduction

Reinforcement learning (RL) is one of machine learning methods, which includes the temporal difference algorithm proposed by Sutton [1] and the Q-learning proposed by Watkins [2]. In the RL, there is an interaction between agent and environment where the agent has to go through numerous trials in order to find out the optimal greedy action. This algorithm has been extensively used in many applications such as scheduler [3], autonomous helicopter flight [4], simple 3D biped [5], and many more.

Q-learning has been used in many applications because it does not require the model of environment and is easy to implement. State-action value, a value for each action from each state, converges to the optimal value as state-action pairs are visited many times by the agent. To improve the speed of learning, SA-Q-learning proposed by Guo finds out the proper balance between exploration and exploitation and learns the optimal policy [6]. Recently, human feedback and guidance are applied to Q-learning to interact with people and to learn new things that are socially relevant to the humans [7], [8].

In this kind of reinforcement learning, agent may reach one of fail states in environment. Agent considers positive or negative feedback as a reward or a punishment, respectively. As an example, fail state can be defined as a situation when goalkeeper of soccer robot allows a goal or when humanoid robot falls down due to external forces. In these cases, agent receives negative reward from the environment as a punishment. By receiving the negative rewards, agent can acquire useful knowledge from them because the probability of choosing state-action value that reaches the fail state is decreased.

This paper aims at proposing a novel algorithm to improve the performance of Q-learning, which employs a separate failure state-action value along with successful experiences of the agent. Note that the proposed algorithm is only applicable for environment having fail state and action space. If the notion of fail states does not exist, the proposed algorithm would be the same as conventional Q-learning.

Separating fail state-action value can be considered as negative feedback, which provides information to decide whether the past action was bad and to predict future actions. Thus, keeping track of failure experiences is about both the past and future intentions for selecting action. If fail states can be predicted few steps before, agent can have more exploratory behavior in state and action space compared to conventional Q-learning. Note that selected action in the proposed algorithm is based on considering both success and fail experiences.

Two-mode Q-learning proposed by Park [9] separates all episodes into failure or success trial. This idea comes from both psychological and economic point of views that human beings tend to treat 'failure' and 'success' separately. However, this algorithm requires huge amount of memory and higher computation cost compared to conventional Q-learning. Subsequently, this paper modifies two-mode Q-learning such that the proposed algorithm requires small memory size, which can be set by user, and reduces convergence speed to maximize the reward that agent receives.

To demonstrate effectiveness of the proposed algorithm, learning property is investigated in a grid world problem through simulation. Simulation result confirms that the performance of proposed algorithm is convincingly better than conventional Q-learning.

This paper is organized as follows: Section II describes brief review on Q-learning. Section III proposes an algorithm for the environment having both goal and fail states to improve convergence speed and to have more exploratory state and action space than that of conventional Q-learning. Section IV describes simulation results and concluding remarks follow in Section V.

1-4244-2384-2/08/$20.00

TABLE I
PSEUDO CODE FOR Q-LEARNING

Initialize all $Q(s,a)$ values
Repeat (for each episode)
  Choose a starting state, s
  Repeat (for each step)
    Choose an action, a, at s using policy derived from Q
    Take the action, a, and observe a reward, r, and a next state, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$
    $s \leftarrow s'$
  Until s' is a goal state
Until a desired number of episodes terminated

II. PRELIMINARIES

In reinforcement learning, the purpose of agent is to discover the optimal action in an environmental state so as to maximize a reward in long term. Q-learning, one of the well-known schemes in the reinforcement learning [2], is easy to implement and is not affected by a learning policy. The value of taking action, a, in state, s, under a policy, π, is called a state-action value and denoted as $Q^{\pi}(s,a)$. Simply, this value represents a value for each action from each state, which depends on a received reward, current value and other values as well. As the state-action value, Q, converges to the optimal value, the agent selects only action with the maximum Q value in a given state.

In Q-learning, agent has the ability to learn state-action value using selected actions in the state without the model of environment. Updating equation of the Q value is as follows:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)) \quad (1)$$

where α is the learning rate and γ is the discount rate. As γ gets closer to 0, the agent will tend to consider only immediate reward, whereas the agent will consider future reward with greater weight as γ becomes closer to 1.

Table I shows the pseudo code for Q-learning algorithm. Once an action is selected, the agent receives an immediate reward and Q value of state-action pair is updated. Since excessive exploratory action selections prevent the progress of Q-learning, the trade-off between exploration and exploitation is needed. For instance, in the ϵ-greedy method, the ϵ value is decreased as the number of episode increases to be exploration in the beginning and exploitation as the learning progresses. Note that each episode ends at a goal state, followed by a reset to a starting state.

Unlike temporal-difference learning, actor-critic learning, and SARSA, Q-learning is off-policy learning. In other words, the state-action value function of a selected action made by the agent is learned irrespective of the policy due to the presence of $\max$ operator in (1). Thus, Q-learning is insensitive to the policy of exploration. Even though the trade-off between exploration and exploitation is needed in the Q-learning similarly as in other reinforcement learning algorithms, the convergence of Q value is not affected by the policy of exploration. Consequently, Q value of the state-action pairs will converge to the optimal value if the state and action pairs are visited many times by the agent.

III. ACCELERATED Q-LEARNING HAVING FAIL STATES

A failure experience of agent is a situation when the agent enters into a fail state. In this case, environment has fail states in its state and action space. In Q-learning, the agent receives a punishment, i.e. a negative reward, from the environment when it reaches fail state. By receiving the negative reward, the Q value of the state-action pair reaching the fail state is decreased.

For instance, a success experience of goalkeeper for a soccer robot can be defined as blocking a ball, whereas a failure experience can be defined as allowing a goal. Thus, the goalkeeping ability of soccer robot and its performance determines whether an agent enters into either a goal state or a fail state. The proposed algorithm employs the failure experiences to modify the exploratory behavior of learning agent.

Toward this effect, a separate failure value, Q_F, that keeps track of failure experiences is introduced into the state-action value, $Q(s,a)$, of Q-learning. The resulting algorithm is presented below.

A. The Main Principle

The agent may encounter a failure experience during learning stage in reinforcement learning. Failure experiences can be regarded as a situation where the number of actions that have led to a goal (termination) state is greater than the threshold number, which is initially set by the user. When environment has one or more fail states, a situation where the agent reaches one of these fail states is regarded as a failure experience as well. In order to cope with these failure experiences, the main principle of proposed algorithm is that the agent increases the probability of not going into the fail state and will have a greater chance of entering into a goal state in the next episode.

In proposed algorithm, there are two separate state and action spaces to consider both conventional Q-learning and failure experience of the agent. Normal Q value, Q_N, is defined as state-action value of the conventional Q-learning where Q_N is updated by using (1). Fail Q value, Q_F, is defined as state-action value based on the failure experience of the agent. Fig. 1 shows the block diagram of proposed algorithm.

Fail Q value based on the failure experience of the agent is calculated as follows:

$$\begin{aligned} e^{Q_F/\tau} &= 1 - p_F\,, \\ Q_F &= \tau\ ln(1 - p_F) \end{aligned} \quad (2)$$

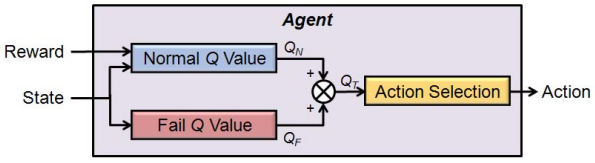


Fig. 1. Block diagram of accelerated Q-learning having fail states

where τ is a temperature to control the trade-off between exploration and exploitation, and p_F is a failure probability, which is obtained by the failure experience of the agent and applied to the state-action pairs, Q_F. Since failure probability has a value between 0 and 1, Q_F is always a negative value in (2).

Action selection depends on the total value, Q_T, which is a summation of the normal Q value, Q_N, obtained by (1) and the fail Q value, Q_F, calculated by (2). Among several action selection methods including random, greedy, ϵ-greedy, and Boltzmann action selections, Boltzmann action selection method is used in the proposed algorithm because it can directly control negative fail Q values, Q_F. Action selection of the agent using Boltzmann equation is as follows:

$$\begin{aligned} Q_T &= Q_N + Q_F\,, \\ p\,(s,a) &= \frac{e^{Q_T(s,a)/\tau}}{\sum_{a' \in A} e^{Q_T(s,a')/\tau}} \end{aligned} \tag{3}$$

where $p(s,a)$ represents the probability of choosing action, a, at state, s. As mentioned in (2), τ controls the amount of exploration. To increase the performance of learning, τ in (2) and (3) should start at high value in the early stage, and decrease to reduce the characteristic of exploration as the number of episode increases.

In Q-learning, the agent requires a lot of experience through exploration of the state-action space in order to find the optimal action selection. However, the agent uses the probability of failure experiences over past trials in the proposed algorithm and it has a greater chance of avoiding a fail state as the learning progresses. In other words, the convergence of Q value is improved by using the failure experience of agent and the agent has more experience of the meaningful state-action space. Note that the other schemes such as eligibility traces and function approximation method for generalizing the state and action space can be used in the proposed algorithm because Q_F is a temporary value.

B. Discounting Fail Q Value

In environment with fail states, the number of selected actions in each episode is considered to apply failure probability. Once a suitable state-action length is calculated, failure probability is applied from a backtracked position using the state-action length to a specific state-action pair, which leads an agent to fail state. According to the environment, the size of state-action length that led the agent to the fail state is varied. In other words, the state-action length, l, depends on the environment and is calculated as follows:

$$l = \begin{cases} 2c & \text{if} \quad m > 2c \\ c+1 & \text{else if } m > c+1 \\ \min(c-1, m-1) & \text{otherwise} \end{cases} \tag{4}$$

where c is the constant value, and m is the size of state-action length leading to the fail state.

As mentioned above, constant c controls the length of past state-action pairs to assign failure probability. If the state space is large, a value of c should be adjusted to high in order to set more failure probabilities for the past state-action pairs. In (4), the maximum number of l is $2c$. Thus, the proposed algorithm requires $2c$ of memory, which stores the past trace of state-action selection.

Once the agent goes to fail state, the equation of applying the failure probability is as follows:

$$\begin{aligned} p_{F_i} &= f(p_{min}, p_{max}, l, i) \\ &= p_{min} \cdot \exp\{\ln(\frac{p_{max}}{p_{min}})\frac{i-(m-l)}{l}\} \end{aligned} \tag{5}$$

where i is the index of action, and p_{min} and p_{max} represent the minimum and the maximum failure probability value, respectively. Note that the failure probability is only assigned from the index of $m-l$ to the index of m. Since p_{max} cannot be defined as 1 in (2), it is set to a value just less than and close to 1. Note that if a failure probability of action is p_{max}, the action almost surely leads the agent to the fail state.

Fig. 2 shows an example of the state-action traces leading the agent to a goal state ($(n-2)$th and $(n+1)$th episodes) and a fail state ($(n-3)$th, $(n-1)$th, and nth episodes). a_{f1}, a_{f2}, and a_{f3} represent actions that take place just before the fail states of the $(n-3)$th, $(n-1)$th, and nth episodes, respectively. Thus, p_{max} is set to corresponding $Q_F(s, a_{f1})$, $Q_F(s, a_{f2})$, and $Q_F(s, a_{f3})$. According to the length, l, the failure probability is applied to the past state-action pairs in fail episode using (5), and fail Q values are calculated using (2).

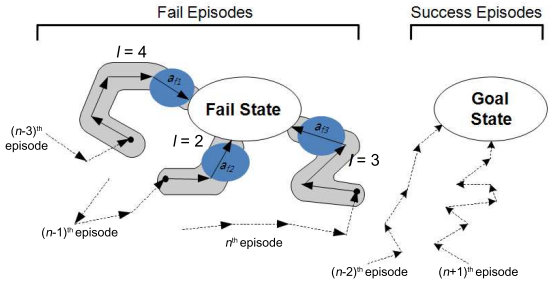


Fig. 2. State-action traces of fail and success episodes

Table II shows the pseudo code for the proposed accelerated Q-learning for fail state and action spaces. The proposed algorithm calculates Q_F values using the failure experience of agent. With the introduction of considering failure experience separately, the proposed algorithm improves the convergence speed and has more exploratory state and action space than that of the conventional Q-learning.

IV. EXPERIMENTS

The proposed accelerated Q-learning having fail states was tested on the 20 x 20 grid world and compared with the results of Q-learning algorithm equipped with the Boltzmann exploration strategy.

TABLE II
PSEUDO CODE FOR THE ACCELERATED Q-LEARNING HAVING FAIL STATES

Initialize all $Q_N(s,a)$ values
Initialize all $Q_F(s,a)$ values
Repeat (for each episode)
  Choose a starting state, s
  Repeat (for each step)
    Choose an action, a, at s using policy given in (3)
    Take the action, a, and observe a reward, r, and a next state, s'
    If (Fail State)
      Calculate length, l, using (4) to be discounted in Q_F
      Assign failure probability, p_F, from $m-l$ to m using (5)
      Update Q_F using (2)
    Else
      $Q_N(s,a) \leftarrow Q_N(s,a)$ + α(r + γ $\max_{a'}$ $Q_N(s',a')$ - $Q_N(s,a)$)
      $s \leftarrow s'$
  Until s' is a goal state
Until a desired number of episodes terminated

A. Grid World

In the grid world shown in Fig. 3, the square marked with "S", "F", and "G" represent the start state, the fail states, and the goal states, respectively. Numbers on the bottom and the righthand side represent the index of state. For most states, there are four possible actions for the agent to move (action 1: Move-To-East, action 2: Move-To-West, action 3: Move-To-South, action 4: Move-To-North). Since the environment of grid world used in simulation is a deterministic one, state transition does not have a probabilistic characteristic.

In order to consider the situation of encountering a wall, the agent has no possibility of moving all the way in the given direction. Thus, there are 2 or 3 possible actions out of 4 actions for the edge states. $c = 3$, $p_{min} = 0.1$, and $p_{max} = 0.9$ were used in the simulation.

When the agent enters into one of four goal states, it receives 100 as a reward. When the agent reaches the fail state, it receives -100 as a punishment. Otherwise, no reward is provided to the agent. In the grid world, the purpose of the agent is to find out the optimum path to arrive at the goal state starting from the start state, and to maximize the reward it receives.

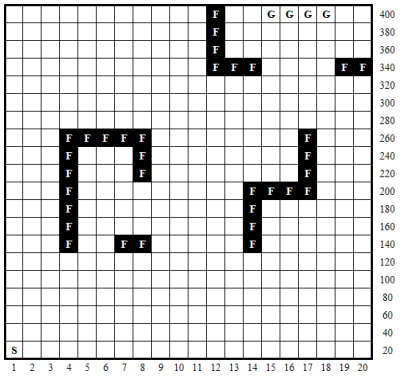


Fig. 3. 20×20 grid world having fail states

B. Results

500 episodes were executed to converge state-action value, Q_T, to the optimal. Considering that grid world is a deterministic environment, both learning rate, α, and discount rate, γ were set to 0.9. Note that high value of α might cause an oscillatory behavior for finding the optimal Q values in nondeterministic environment. In the Boltzmann action selection, the temperature, τ, was determined as follows:

$$\tau = \tau_{max} - k \tag{6}$$

where τ_{max} is set to 200, and k is the number of episode. The value of τ decreases as the number of episode increases, and the minimum value of τ was set to 1.

Fig. 4 shows the number of actions before reaching the goal states where x-axis represents the number of episodes. Straight line represents the result of conventional Q-learning where dotted line represents the result of proposed algorithm. After each episode, the randomness in Boltzmann action selection was removed and counted the number of actions before the goal state. In the early phase, agent could enter into the fail state. Thus, if the number of counted actions was greater than 1,000, this greedy action selection stopped and continued to the next episode.

As shown in Fig. 4, the convergence of proposed algorithm is faster than that of Q-learning. The conventional Q-learning reached the goal state after 287^{th} episode with 35 actions, whereas the proposed algorithm accomplished with 33 actions after 81^{st} episode. Total received reward for Q-learning was 975.7 where this value was 971.3 for the proposed algorithm.

The performance of proposed algorithm was convincingly enhanced because Q_F predicts the fail state in a few steps before. This allows the agent to have more exploratory state and action space compared to the conventional Q-learning. Since Q_F does not update in every step, the computational cost of proposed algorithm is almost the same as the conventional approach except when the agent reaches fail states. Additional costs are only a separate fail state-action value, Q_F, and the memory size of $2c$, which stores the past failure state-action trace.

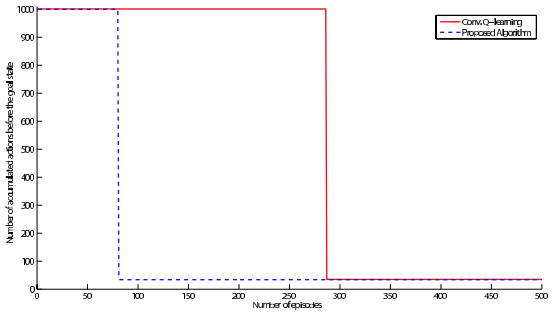
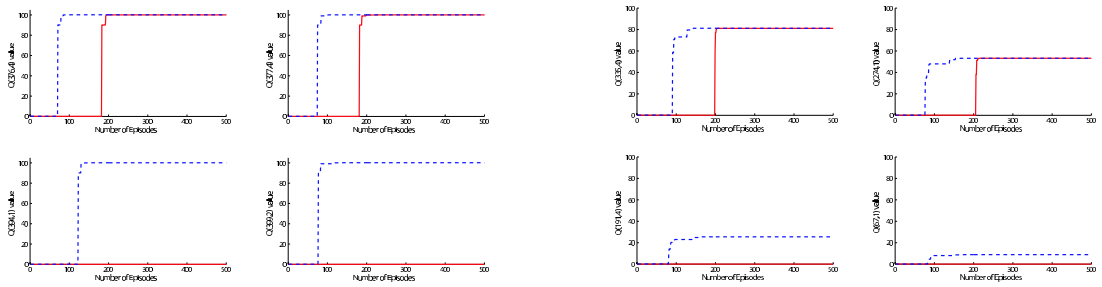


Fig. 4. Learning graph between conventional Q-learning and the proposed algorithm



(a) States located near goal states

(b) Arbitrary chosen states

Fig. 5. Convergence of $Q(s, a)$ values between conventional Q-learning (solid line) and the proposed algorithm (dotted line)

Fig. 5 represents the convergence of $Q(s, a)$ values as the number of episode increases. Note that state-action pairs located near goal states were examined in Fig. 5(a), whereas arbitrary chosen state-action pairs were done in Fig. 5(b). Similar to the learning graph, solid and dotted line display the result of using conventional Q-learning and the proposed algorithm, respectively.

As shown in Fig. 5, the proposed algorithm converged to an expected reward value in a fewer episodes compared to the conventional Q-learning. Conventional Q-learning never visited several states such as $Q(394, 1)$, $Q(394, 2)$, $Q(191, 4)$, $Q(67, 1)$, etc, and updated the corresponding state-action value. In other words, Q-learning always fell into one of fail states in the early stage of episode. After 200^{th} episode, the agent selected its behavior in the focus of exploitation because τ_{max} was set to 200. By employing the failure experience as separate state-action pair, the agent had more exploratory state and action space, and enhanced the convergence speed to find out the optimal path while maximizing its reward value. This optimal path from the start state to the goal state with 33 actions is shown along with Q values in Fig. 6.

V. Conclusion

This paper proposed a novel algorithm, which considers both success and failure experiences of agent. It employed a separate fail Q value, Q_F, that keeps track of failure experiences and uses this value to modify the exploratory behavior of learning agent. If conventional Q-learning was applied in the real experiment, a lot of iterations were required to obtain the optimal Q values to guarantee system performance. Thus, accelerated Q-learning algorithm was proposed for environment having both goal and fail states, which used small amount of past state-action traces and significantly reduced the convergence speed to determine the optimal states while maximizing its reward value. Verifying the proposed algorithm in nondeterministic environment and in real experiment are left as a further work.

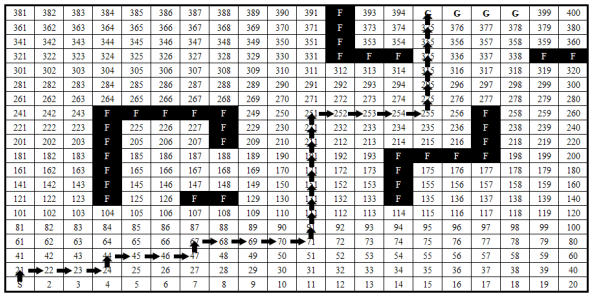


Fig. 6. Results of optimal path in 20×20 grid world having fail states

References

[1] R.S. Sutton, D. Precup and S. Singh, "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning," *Artificial Intelligence*, vol. 112, pp. 181-211, 1999.

[2] C.J.C.H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.

[3] A. McGovern, E. Moss and A.G. Barto, "Building a Basic Block Instruction Scheduler with Reinforcement Learning and Rollouts," *Machine Learning*, vol. 29, pp. 141-160, 2002.

[4] A.Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger and E. Liang, "Autonomous Inverted Helicopter Flight via Reinforcement Learning," *Experimental Robotics IX*, START 21, pp. 363-372, 2006.

[5] R. Tedrake, T.W. Zhang and H.S. Seung, "Stochastic Policy Gradient Reinforcement Learning on a Simple 3D Biped," *Proc. of 2004 IEEE/RSJ Int. Conf. on Intelligent Robot and Systems*, pp. 2849-2854, Oct. 2004.

[6] M. Guo, Y. Liu and J. Malec, "A New Q-learning Algorithm Based on the Metropolis Criterion," *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 5, pp. 2140-2143, 2004

[7] A. Lockerd and C. Breazeal, "Tutelage and Socially Guided Robot Learning," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3475-3480, Oct. 2004.

[8] A. Thomaz and C. Breazeal, "Asymmetric Interpretations of Positive and Negative Human Feedback for a Social Learning Agent," *IEEE Int. Conf. on Robot and Human Interactive Communication*, pp. 720-725, Aug. 2007.

[9] K.-H. Park and J.-H. Kim, "Two mode Q-learning," *IEEE Congress on Evolutionary Computation*, pp. 2449-2454, Dec. 2003.