Fuzzy-Q Learning for Autonomous Robot Systems

Il Hong Suh+, Jae-Hyun Kim+, and Frank Chung-Hoon Rhee++
+Intelligent Control and Robotics Laboratory
++Computational Vision and Fuzzy Systems Laboratory
Department of Electronic Engineering
Hanyang University
Ansan, KOREA

Abstract

It is desirable for autonomous robot systems to posses the ability to behave in a smooth and continuous fashion when interacting with an unknown environment. Since Q-learning is normally used for optimizing a series of discrete actions, it may be undesirable when applied to a real environment which involves continuous states and actions. In this paper, we propose a new method of Q-learning that incorporates a fuzzy interpolation technique which is used to approximate a continuous state. Our learning method can estimate a current state by its neighboring states and has the ability to learn its actions similar to that of Q-learning. Thus, our method can enable robots to react smoothly in a real environment. Simulation results involving an autonomous robot are given to show the validity of our method.

1. Introduction

Recently, many intelligent robot systems have been developed to perform various tasks which include navigation, control, and recognition [1~3]. To implement such a robot system, it is important for the system to properly react in an unknown environment by learning its actions through experience. For this purpose, reinforcement learning methods have been receiving increased attention for use in robot systems due to its appealing attributes.

Prior works on reinforcement learning have been conducted on the basis of discrete states and events [4~6]. By using such learning methods, a robot can learn its proper actions by interacting with an unknown environment. Among these methods, Q-learning has been widely used, where the Q-values that are generated are considered as action values. These values are then used to find the proper action in each state. However, since Q-learning deals with discrete actions and states, an enormous amount of states may be necessary for an autonomous robot to learn an appropriate action in a continuous environment. Hence, this may require a great deal of time to learn the proper actions for all the states. To overcome this problem, variations of the Q-learning algorithm have been developed [7~9]. In one of these algorithms, the Q-values are evaluated by using Tagaki-Sugeno-Kang (TSK) type fuzzy rules [9], in which a set of consequent coefficients are modified by a steepest descent method. Since the Q-values generated by the rule base does not posses statistical information, it cannot ensure that the algorithm will find the optimal policy [9].

To solve such a problem, we propose new method of Q-learning that deals with continuous actions and states. In this method, a fuzzy interpolation technique is utilized as a state approximator, where the neighboring states are used to estimate the current and next state. In addition, we present a Q-value distribution model that can be used to rapidly compute a current maximum Q-value. Based on this model, a Q-value is updated on every reinforcement signal using the well-known Q-value update equation[5]. Thus, this approach enables the Q-values to converge.

2. The Q-learning Algorithm

Reinforcement type learning methods can be modeled as a discrete time cyclic interaction between an autonomous robot and an environment [6]. The interaction process can be explained as follows. First, the robot senses the current state and executes an action for that state. Then, a change in the environment creates the next state and assigns the robot a reward for that particular action. Based on the reward, the robot evaluates the exerted action in that state. Q-Learning provides a robot (agent) the ability to act optimally by evaluating the consequences of its actions. Specifically, a Q-function is used to estimate the future reinforcement for performing its actions. This enables a robot to select a proper action for a particular state. The Q-learning algorithm can be summarized as follows.

Q-learning Algorithm

<Initialize>

1. Initialize the Q-table by assigning arbitrary values.
2. Construct an initial policy f_i based on the initial values in the Q-table. That is,

$$f_i \leftarrow a \text{ such that } Q_i^a(t+1) = \max_{b \in A}\{Q_i^b(t)\}, \quad (1)$$

where t is the tth iteration, i is the current state, f_i is the

action of the current state in the policy table, and $Q_i^a(t+1)$ is the Q-value of current action a in current state i in the next iteration.

<Repeat forever>

3. For the current state i, execute an action a based on the policy table with probability p or a random action with probability 1-p. The random action plays an important role in obtaining the optimal policy.
4. Receive a reward r from the environment.
5. Update the Q-value using

$$Q_i^a(t+1) = \alpha Q_i^a(t) + (1-\alpha)(r_i^a + \gamma \max_{b \in A}\{Q_{i+1}^b(t)\}), \quad (2)$$

where α ($0<\alpha<1$) can be considered as a learning rate and γ as a discounting factor.
6. Update the policy f.

It is to be noted that the Q-value for an optimal action can be defined by combining the immediate reward r, transition probability function T, and the Q-value for the next state as

$$Q_i^a(t+1) = r(t) + \gamma \sum_{i+1 \in S} T(i,a,i+1) \max_{b \in A}\{Q_{i+1}^b(t)\}, \quad (3)$$

if transition probability function T is given *a priori*.

Now, we will show that the Q-value update equation in (2) is equivalent to (3) as the number of iterations approachs to infinity. For this, note that $Q_i^a(t+1) = Q_i^a(t)$ in the steady state in Q-learning. Thus, we can obtain

$$Q_i^a(t) = r_i^a(t) + \gamma \max_{b \in A}\{Q_{i+1}^b(t)\}, \quad (4)$$

if we let $T(i,a,i+1)=1$ for all i and a. Hence, (3) becomes identical to (4).

3. The Fuzzy Q-Learning Algorithm

Fuzzy Q-learning methods may be considered as an extension of its original version that was mentioned in Section 2. Unlike the conventional Q-learning method, its fuzzy versions need not learn every state in the state space. In fact, optimizing the actions for only a few representative states is needed. With this in mind, we perform fuzzy interpolation to estimate the current states action.

3.1 The fuzzy Q-table for a continuous state space

In Q-learning, the Q-table may be considered as a set of action values for all state-action pairs. Therefore, for a Q-learning scheme to perform satisfactorly in a continuous state space, an infinite number of states and actions in the Q-table is required. Hence, to remedy this constraint, we use a continuous triangle-type Q-table model in which a particular action has the highest Q-value and the opposite action has a zero value. Thus, the relationship among all the actions is simply defined as an Euclidean distance. This is shown in Figure 1, where Q_{max} and a_{max} denote the maximum amplitude and width, respectively.

3.2. Fuzzy interpolation in Q-learning

Due to the ambiguity of defining a state that lies between known states, we define the present state by employing a fuzzy interpolation of its surrounding states. The membership values for the surrounding states are used to define the relationship between the current state and its neighboring states. This is illustrated in Figure 2. Here, we define the membership function as an exponential as shown below.

$$\mu_{ij} = \frac{e^{-\lambda \left\| s_{ij} - s_i \right\|}}{\sum_{j=1}^{N} e^{-\lambda \left\| s_{ij} - s_i \right\|}}, \quad (5)$$

where λ is a scale factor, s_i is the current (ith step) state and s_{ij} is the jth neighboring state of the current state. The current state can be defined by

$$s_i = \sum_{j=1}^{N} \mu_{ij} s_{ij}, \quad (6)$$

where N is the number of neighboring states and μ_{ij} is defined as a membership value of the current state in the jth neighboring state. In addition, if we interpret the current state Q-value as Q_i and estimate these values by using a fuzzy interpolation method, Q_i may be obtained as

$$Q_i = \sum_{j=1}^{N} \mu_{ij} Q_{ij}, \quad (7)$$

where Q_{ij} is defined the maximum Q-value of the jth neighboring state of the ith current state.

Using the above definition, the action for the current state is simply determined as the maximum of all Q-values. That is,

$$a_i = \arg(\max_{\forall a}\{\sum_{j=1}^{N} \mu_{ij} Q_{ij}^a\}). \quad (8)$$

Figure 3 shows an example of the current action generated by equation (8) for the current state from Figure 2. After executing the current action, the current state changes to its next state and is assigned a reward. The updated Q-value that utilizes the reward may be calculated as

$$Q_i^{a_i}(t+1) = \alpha Q_i^{a_i}(t) + (1-\alpha)(r_i^{a_i}(t) + \gamma Q_{i+1}^{a_i}(t)). \quad (9)$$

It is here noted that (9) can be written as

$$Q_i^{a_j}(t) = r_i^{a_i}(t) + \gamma Q_{i+1}^{a_i}(t), \qquad (10)$$

as t goes to infinity. Therefore, (10) is equivalent to (4). This implies that the Q-value updated by the proposed fuzzy learning also has statistical information as in the conventional Q-learning. The difference between equation (9) and (2) is that from the Q-value for the neighboring states in fuzzy Q-learning, the present Q-value can be obtained. By substituting equation (7) into (9), we can obtain the maximum Q-value for the next iteration. In order to estimate the action for the current state that is obtained from optimizing the neighboring states, inversely (using equation (9)), the updated amount of Q-value for the current state can be used to determine the updated amount of Q-values for each neighboring state. That is,

$$dQ_{ij}^{a_i}(t+1) = \mu_{ij}(Q_i^{a_i}(t+1) - Q_i^{a_i}(t)). \qquad (11)$$

Therefore, using equation (11), we can update the Q-value for the current action for the ith neighboring state after considering the membership for the updating amount of the Q-value as

$$Q_{ij}^{a_i}(t+1) = Q_i^{a_i}(t) + dQ_{ij}^{a_i}(t+1). \qquad (12)$$

Based on the current action and the updated amount of the Q-value, we can modify the proper action to have a maximum Q-value. There exits two types of updating methods. One is, if the updated Q-value is smaller than the largest Q-value in current state and is larger than the Q-value of current action, then the largest Q-value moves in the direction of the current action to update the Q-value. In addition, if the updated Q-value is smaller than Q-value of current action, then the largest Q-value moves in the opposite direction of the current action. This update scheme can be represented by

$$a_{ij}^k(t+1) = a_{ij}^k(t) + \eta \,\mathrm{sgn}(a_{ij}^k(t) - a_i^k(t)), \qquad (13)$$

where $\eta = \dfrac{2a_{\max}^k}{Q_{\max}} \left| dQ_{ij}^{a_i}(t+1) \right|$.

In (13), η is used to determine the amount to update the k-th axis action value and sgn() is used to determine the direction of the actions movement. Second, if the updated Q-value is larger than the current maximum Q-value, then the action is used as the one with the largest Q-value. That is.

$$a_{ij}^k(t+1) = a_i^k(t). \qquad (14)$$

As an example, Figure 4 shows that the triangle-shaped Q-table model moves towards the current action. The fuzzy Q-learning algorithm can be summarized as follows.

Fuzzy Q-Learning Algorithm

<Initialize>

1. Initialize all necessary parameters.

<Repeat forever>

2. For the current state, compute the membership values of the current state in the neighboring states.
3. Compute the current action a by equation (8).
4. Execute the current action and receive a reward r from the environment.
5. Compute the current Q-value by equation (7).
6. Compute the Q-values of its neighboring states by equation (11,12).
7. Update the current action in the current state by equation (13,14) based on Q-table model.

4. Simulation Results

To illustrate the effectiveness of our proposed method of Q-learning, a simulation of an autonomous visual robot that is used to achieve a target feature in a 2-D feature space is shown. The environment specifications are shown in the following table.

Table 1. The environment for Simulation

Environment	Specification
Row state axis	0 - 350 state space
Column state axis	0 - 350 state space
Action	4-direction vectors (Q)
Sampling Time	All direction vectors (FQ)
	0.032 sec

The reward used in this simulation is defined as

$$r = \frac{(\|x_{i+1} - x_g\| - \|x_i - x_g\|)}{K}, \qquad (15)$$

where x_i and x_{i+1} denote the current and next feature vectors, respectively, x_g is a target feature vector and K is the step size of the feature space. Also, the convergence for learning is defined as the number of steps in an iteration that is within +5% from the optimal number of steps and the difference between current and previous number of steps is smaller than 5% for every decade of iterations.

In a 2-D space, the robot starts from an initial feature vector (50,50) and its goal is to reach target feature vector (320,320). The resolution of each state axis is set at 35. As shown in Figure 5, in the initial iteration, the robot wanders here and there updating the Q-values for its performed actions. However, the maximum Q-value of each state converges after 400 iterations, thus the agent follows a policy with the sum of Q-values that is maximized. On the other hand, in the case of fuzzy Q-learning, the robot behaves well under such an optimal policy after 47 iterations. This is shown in Figure 6. As expected, our fuzzy Q-learning method shows a faster

convergence than the conventional Q-learning method in which the goal is achieved in a smaller number of steps. The robot can also learn a smoother action set, thus enabling it to react to a current state with a better tuned action. We also simulated the same task with a smaller resolution of each state axis. We find that our fuzzy Q-learning works satisfactory (see Figure 7) as long as the state space of a rectangle region can be linearly approximated.

5. Conclusion

In this paper, we proposed a new method of Q-Learning that uses a fuzzy interpolation technique for approximating continuous states. In order to show the validity of our method, we conducted some computer simulations in which a simple Q-learning was compared with our method. Results show that the required number of iterations needed to reach a goal using our method was much less. Also, we conclude that our method is suitable for applications that require continuous states and actions as in a real environment. However, in both Q-learning and Fuzzy Q-learning, setting the parameter of the random action ratio can be difficult. We are currently working on an automatic estimation method to obtain these parameters by detecting the extent of learning.

6. References

[1] M. A. Salichs, E. A. Puente, D. Gachet, and J. R. Pementel, "Learn\ing behavioral control by reinforcement for an autonomous mobile robot", *IEEE Conference on R&A*, vol.1, pp. 1436-1441, 1993.

[2] H. Berenji , P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcement," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, Sept. 1992.

[3] L. J. Lin, "Programming robots using reinforcement learning and teaching," In *Proc. of the Ninth National Conference on Artificial Intelligence*, 1991.

[4] G. Tesauro, *Practical issues in temporal difference learning*. Machine Learning, 1992.

[5] C. Watkins. P. Dayan, "Q-learning, technical note." Machine Learning, Vol. 8, pp.279-292, 1992.

[6] C. Watkins. "Learning from delayed rewards," Ph.D. Thesis. University of Cambridge, England, 1989.

[7] P. Y. Glorennec. "Fuzzy Q-learning and dynamical fuzzy Q-learning." *IEEE Conference on R&A*. vol. 1. pp. 474-479. 1994.

[8] H. R. Berenji. "Fuzzy Q-learning: A new approach for fuzzy dynamic programming." *IEEE Conference on R&A*. vol. 1. pp. 486-491, 1994.

[9] T. Horiuchi. A. Fujino. O. Katai. and T. Sawaragi. "Fuzzy interpolation-based Q-learning with continuous states and actions," *IEEE Conference on Fuzzy Systems*, vol. 1, pp. 594-600, 1996.

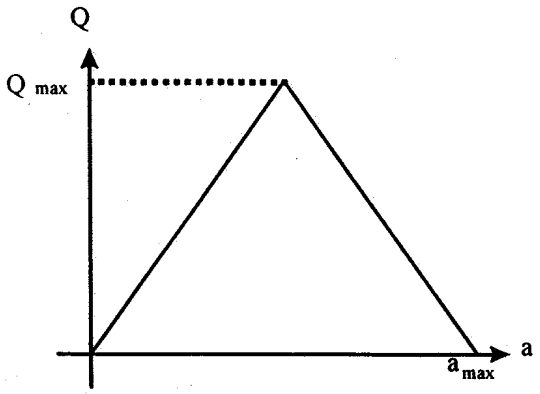


Figure 1. Triangle-type modeling of the Q-values

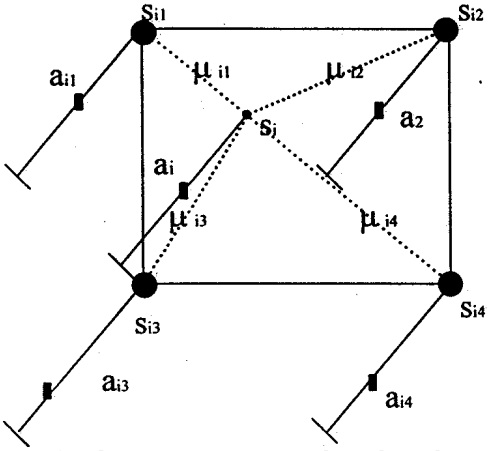


Figure 2. Current state estimation by using its neighboring states.

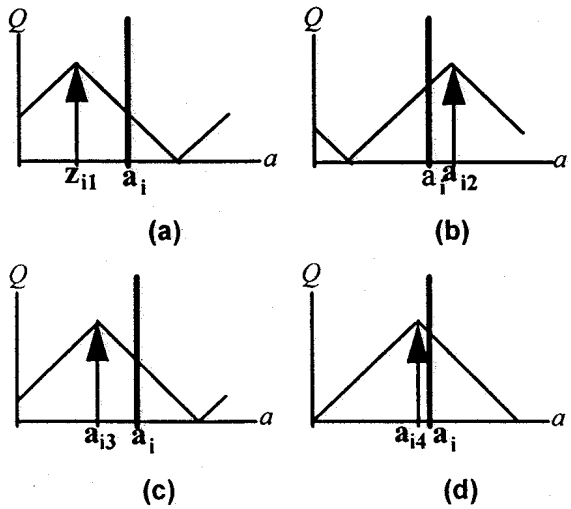
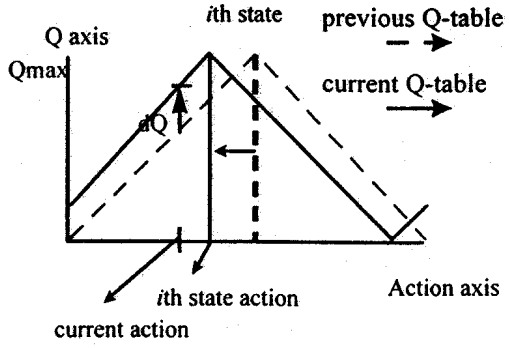
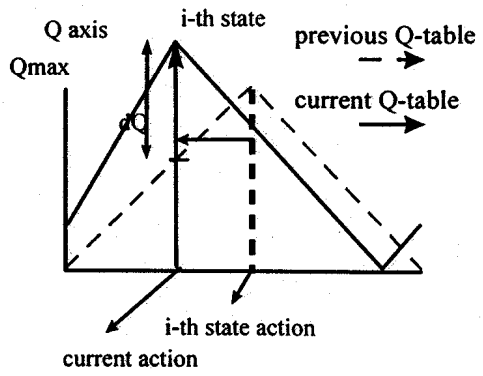


Figure 3. An example of the current action a_i and the actions, $a_{i1}, a_{i2}, a_{i3}, a_{i4}$, of its neighboring states, $S_{i1}, S_{i2}, S_{i3}, S_{i4}$,

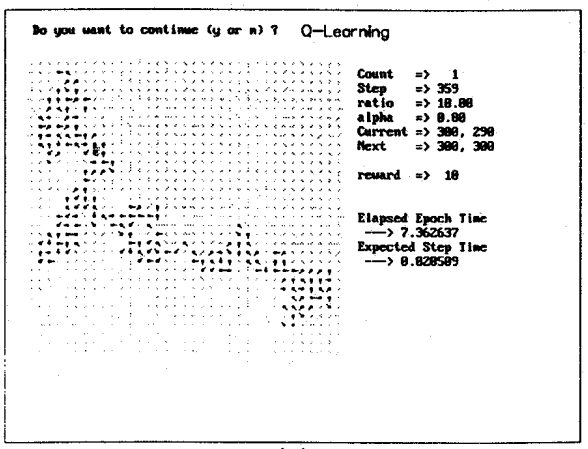


(a)

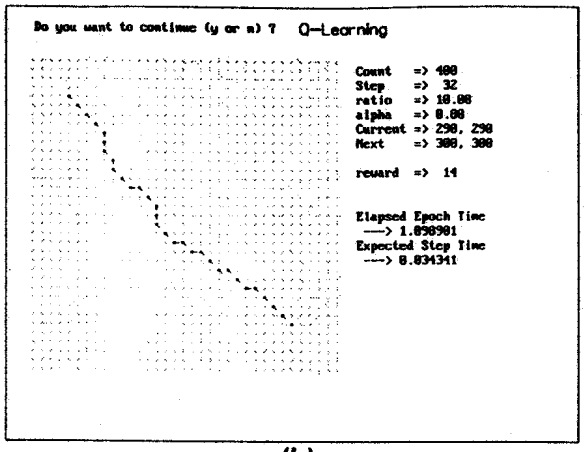


(b)

Figure 4. The Q-value update scheme: (a) width modification and (b) height modification.

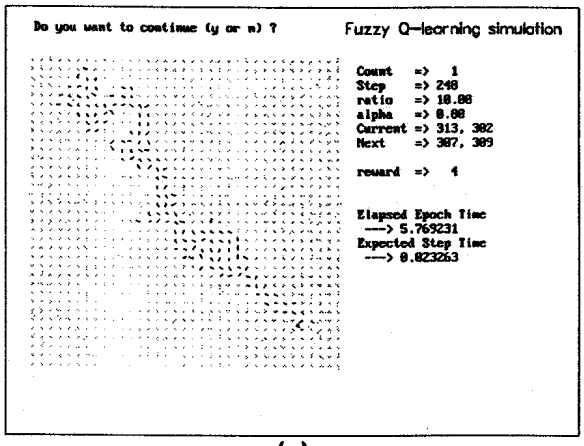


(a)

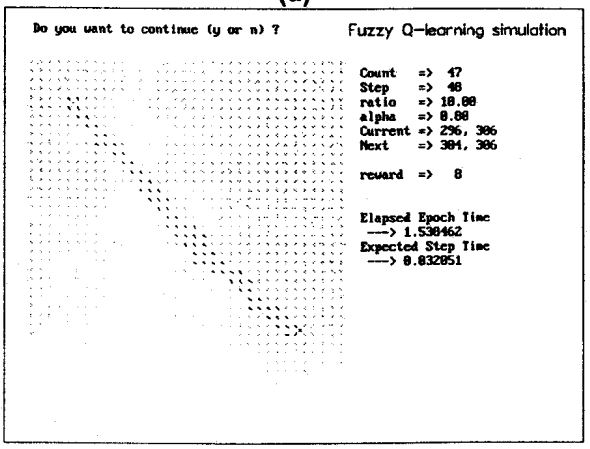


(b)

Figure 5. The Q-learning simulation results: (a)1st iteration and (b)400th iteration.



(a)



(b)

Figure 6. The fuzzy Q-learning simulation results: (a)1st iteration and (b)47th iteration.