Improvement Continuous Valued Q-learning and its Application to Vision Guided Behavior Acquisition

Yasutake Takahashi[1], Masanori Takeda[3], and Minoru Asada[1]

[1] Dept. of Adaptive Machine Systems,
Graduate School of Engineering Osaka University, Suita, Osaka 565-0871, Japan
[2] Nara Institute of Science and Technology, Japan

Abstract. Q-learning, a most widely used reinforcement learning method, normally needs well-defined quantized state and action spaces to converge. This makes it difficult to be applied to real robot tasks because of poor performance of learned behavior and further a new problem of state space construction. We have proposed Continuous Valued Q-learning for real robot applications, which calculates contribution values to estimate a continuous action value in order to make motion smooth and effective [1].
This paper proposes an improvement of the previous work, which shows a better performance of desired behavior than the previous one, with roughly quantized state and action. To show the validity of the method, we applied the method to a vision-guided mobile robot of which task is to chase a ball.

1 Introduction

Reinforcement learning has been receiving increased attention as a method with little or no a priori knowledge and higher capability of reactive and adaptive behaviors for robots through the interactions with their environment[2]. However, Q-learning, a most widely used reinforcement learning method, normally needs well-defined quantized state and action spaces to converge. This makes it difficult to be applied to real robot tasks because of poor performance of learned behavior and further a new problem of state space construction.

We have proposed Continuous Valued Q-learning for real robot applications[1]. The proposed method interpolates continuous values between roughly quantized states and actions. This contributes to realize smooth motions with much less computational resources. This paper proposes an improvement of the previous work through the discussions what are problems in the previous one and how they are solved. The proposed method obtained the better performance of desired behavior than the conventional real-valued Q-learning method, with roughly quantized state and action. To show the validity of the method, we applied the method for a vision-guided mobile robot of which task is to chase the ball.

2 Continuous Valued Q-Learning

2.1 State Representation and Action Interpolation

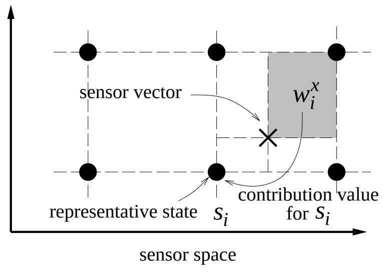


Fig. 1. Calculation of contribution value w_i for the representative state s_i in the case of two-dimensional state space

A basic idea for continuous value representation of the state, the action, and the reward in Q-learning is to describe them as contribution vectors of the representative states, actions, and rewards, respectively. First, we quantize the state/action space adequately. Each quantized state and action can be the representative state $s_1, \cdots,$ or s_n and the representative action $a_1, \cdots,$ or a_m, respectively. The state (action) representation is given by a contribution vector of the representative state $\boldsymbol{s} = (w_1^s, \cdots, w_n^s)$ ($\boldsymbol{a} = (w_1^a, \cdots, w_m^a)$). A contribution value indicates the closeness to the neighbor representative state (action). The summation of contribution values is one.

There are several methods to calculate a contribution value, and we select a following one because of the simplicity of the calculation. For the readers' understanding, here we assume that the N-dimensional sensory information directly construct the N-dimensional state space, and the motor space has M-dimensions. The robot perceives the current sensory information as a state vector $\boldsymbol{x} = (x_1, x_2, \cdots x_N)$, and executes motor command $\boldsymbol{u} = (u_1, u_2, \cdots u_M)$.

First, we tessellate the state space into N-dimensional hyper cubes[1]. The vertices of all hyper cubes correspond to the representative state vectors $\boldsymbol{x}^i = (x_1^i, x_2^i, \cdots x_N^i)$ $i = 1, \cdots, n$ (here, n denotes the number of the vertices), and we call each vertex the representative state s_i. The contribution value w_i^s for each representative state s_i when the robot perceives the input $\boldsymbol{x} = (x_1, x_2, \cdots x_n)$ is defined as the volume w_i^s of the box diagonal to the state s_i. Mathematical formulation is given by

$$w_i^s = \prod_{k=1}^{N} l_i(x_k) \quad , \text{where} \quad l_i(x_k) = \begin{cases} 1 - |x_k^i - x_k| & \text{if} \quad |x_k^i - x_k| \leq 1 \\ 0 & \text{else} \end{cases} . \tag{1}$$

[1] the unit length is determined by normalizing the length of each axis appropriately

Fig.1 shows a case of two-dimensional sensor space. The area $w_i^{\boldsymbol{x}}$ is assigned as a contribution value for state s_i. Thus, the state representation corresponding to the input $\boldsymbol{x}$ is given by a state contribution vector $\boldsymbol{w}^s = (w_1^s, \cdots, w_n^s)$.

Similarly, the representative action vectors $\boldsymbol{u}^j = (u_1^j, u_2^j, \cdots u_M^j)\ j = 1, \cdots, m$, and the representative action a_j are given. When an action contribution vector $\boldsymbol{w}^a = (w_1^a, \cdots, w_m^a)$ is given from the current policy π of the agent, the mapping to the motor command is defined as follows:

$$\boldsymbol{u} = \sum_{j=1}^{m} w_j^a \boldsymbol{u}^j. \tag{2}$$

2.2 Extension to the Continuous Space

According to the state and the action definition mentioned above, we modify the standard Q-learning algorithm as follows:

When the agent goes around its environment, it has a certain policy π, which returns a representative action $a \in A = \{a_1, a_2, \cdots, a_m\}$, for each each representative state $s \in S = \{s_1, s_2, \cdots, s_n\}$. That is:

$$a = \pi(s). \tag{3}$$

Using this policy π, the agent calculates the contribution vector of the representative action as follows:

$$w_j^a = \sum_{i=1}^{n} w_i^s f(s_i, a_j, \pi), \text{ where } \quad f(s_i, a_j, \pi) = \begin{cases} 1 & \text{if} \quad \pi(s_i) = a_j \\ 0 & \text{else} \end{cases}. \tag{4}$$

The Q-value when executing the representative action a_j at the representative state s_i is denoted by $Q(s_i, a_j)$. A Q-value at any state with policy π $(\boldsymbol{x}, \pi)$ is given by:

$$Q(\boldsymbol{x}, \pi) = \sum_{i=1}^{n} w_i^s Q(s_i, a), \quad \text{where} \quad a = \pi(s_i). \tag{5}$$

Given the representative state s_i, the representative action based on the optimal policy π^* is calculated by:

$$a_k = \pi^*(s_i), \quad \text{where} \quad k = \arg\max_j Q(s_i, a_j) \tag{6}$$

The state value function $V(\boldsymbol{x})$ is calculated by:

$$V(\boldsymbol{x}) = \sum_{i=1}^{N} w_i^s V_i \quad \text{, where} \quad V_i = \max_j Q_{i,j}. \tag{7}$$

Then, the Q value when choosing an policy π at the current state $\boldsymbol{x}$, and transiting the next state $\boldsymbol{x}'$ given reward r is updated by:

$$Q(s_i, a_j) \leftarrow Q(s_i, a_j) + \alpha w_i^s f(s_i, a_j, \pi)(r + \gamma V(\boldsymbol{x}') - Q(\boldsymbol{x}, \pi)) \tag{8}$$

where α is a learning rate and γ is a fixed discount factor between 0 and 1.

3 Improvements of the learning algorithm

3.1 Action Contribution Vector

The calculation of the contribution vectors of representative actions from the actual motor command may not work well as expected. We show a simple example. There are two representative states, i.e. “left” and “right”, and 3 representative actions, i.e. “fast”, “slow”, and “stop”. The goal state is the “right” one. The “fast” action is optimal at the “left” representative state, and the “stop” action is optimal at the “right” representative state. When the agent is at the middle position between two representative state, the agent may calculate the slow action” by the interpolation between the optimal actions. Even if the agent execute the ”slow” as the result of action interpolation, the Q-values for ”slow action” will be reinforced on the both representative states. Since the ”slow” action is not the optimal action at the both state, this reinforcement causes a wrong effect.

To avoid this situation, we use the contribution vectors of representative actions not calculated from the actual motor command, but based on the agent’s current policy π (Equations (5) and (8)). In the last example, if the agent execute the ”slow” as the result of action interpolation of “fast” and “stop”, the “fast” and “stop” action will be reinforced, and “slow” action will not.

3.2 State Value Function

The calculation of a state value function $V(\boldsymbol{s})$ in the previous version[1] was:

$$V(\boldsymbol{x}) = \sum_{i=1}^{N}\sum_{j=1}^{M} w_i^s w_j^{a*} Q_{i,j}. \tag{9}$$

This means that a state value function at the state $\boldsymbol{s}$ should be the Q value at the state taking the action $\boldsymbol{a}^*$ based on the optimal policy. However, this is not a good idea because the state values except for the representative states are not properly estimated unless the optimal action at all states are the same one.

Let us consider a case in which the optimal representative action at a representative state differs from one at the other representative state. The contribution of the each representative action w_j^{a*} has a mean value between the 0 to 1. The estimated state value at each representative state $w_j^{a*}Q_{i,j}$ is always less than the state value at the representative state $V_i = max_j Q_{i,j}$. Then the estimated state value at the middle state is underestimated. This phenomena is not appropriate for the updating the Q-table. We should use the equation(7).

3.3 Q value Estimation

The Q value interpolation in the previous version was:

$$Q(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_i^s w_j^a Q_{i,j} \tag{10}$$

As we mentioned in previous subsections, it is not a good idea to use the action contribution vector in order to calculate or update the Q-table. Therefor, we changed the definition of the state value function from equation (10) to equation (7), and the equation of Q value estimation like (5), or the agent may underestimate the Q value than the state value $V(x)$ if it takes the optimal policy.

3.4 Update Equation

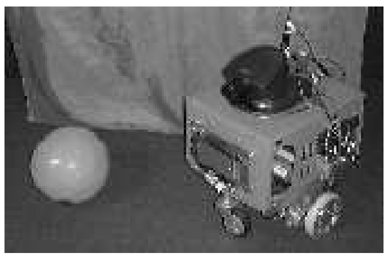
The equation of update Q value in the previous version was:

$$Q_{i,j} \leftarrow Q_{i,j} + \alpha w_i^s w_j^a (r + \gamma V(\boldsymbol{s}') - Q_{i,j}). \tag{11}$$

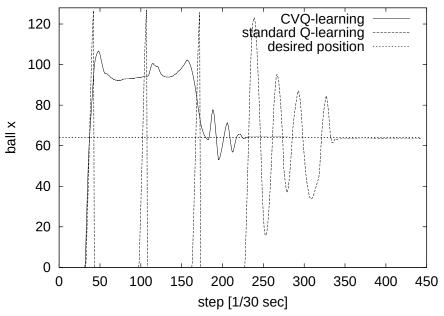
This updates $Q_{i,j}$ even if the estimation is perfect. This phenomena does not lead to the convergence, but causes the divergence of learning. We should use the equation(8).

4 Experimental

In order to show the validity of the proposed method, we apply the method to a mobile robot of which task is to chase a ball, one of the vision-guided behavior acquisition. Fig.2(a) shows a picture in which the mobile robot we have designed and built and the ball to chase are shown. A simple color image processing (Hitachi IP5000) is applied to detect the ball area in the image in real-time (every 33ms).



(a) A mobile robot and a ball



(b) Step responses with/without our method

Fig. 2. Experiment

The robot has a TV camera of which visual angles are 35 degs and 30 degs in horizontal and vertical directions, respectively. The camera is tilted down 23.5

degs to capture the ball image as large as possible. The image area is 128 by 110 pixels, and the state space is constructed in terms of the centroid of the ball image. The driving mechanism is PWS (Power Wheeled System), and the action space is constructed in terms of two torque values to be sent to two motors corresponding to two wheels. These parameters of the robot system are unknown to the robot, and it tries to estimate the mapping from sensory information to appropriate motor commands by the method.

The state and action spaces are two-dimensional ones, and tessellated into 5 by 5 and 3 by 3 grids, respectively. The learning rate α and the discount factor γ are 0.05 and 0.7, respectively. The parameters are constant during the learning.

Action selection during the learning was random and the goal state is a situation that the robot captures the ball region at the center of the image and its size is pre-specified value so that the ball is located just in front of the robot (about 50cm apart). In other words, the goal state is that the coordinates of the centroid of the ball region is (64,55).

To show the efficiency of the exploration and smoothness of the motion, we compare the results with standard Q-learning based on the quantized state and action spaces. We show the difference in the step responses of the robot behavior after the learning. The ball is positioned 3m far from the robot. Fig.2(b) shows the step responses, where the behavior based on the proposed method successfully reaches the goal state with smaller error and smoother motion than the standard Q-learning method. The standard Q-learning method sometimes lost the ball and oscillated left and right before reaching the goal state.

Acknowledgments

We would like to thank Shoichi Noda (Displays, Hitachi Ltd., Japan) for fruitful discussions about modification of the CVQ-learning.

References

[1] Y. Takahashi, M. Takeda, and M. Asada. Continuous Valued Q-learning for Vision-Guided Behavior Acquisition. *Proceeding of the 1999 IEEE International Conference on Multi-sensor Fusion and Integration for Intelligent Systems*, 255–260, 1999.

[2] J. H. Connel and S. Mahadevan, editors. *Robot Learning.* Kluwer Academic Publishers, 1993.