Extended Q-learning : Reinforcement Learning using Self-Organized State Space

Shuichi ENOKIDA, Takeshi OHASI,
Takaichi YOSHIDA and Toshiaki EJIMA

Kyushu Institute of Technology, Dept. of Artificial Intelligence,
680-4 Kawazu, Iizuka, Fukuoka, 820-8502, Japan
{enokida, ohashi, takaichi, toshi}@mickey.ai.kyutech.ac.jp

Abstract. We propose Extended Q-learning. To accommodate continuous state space directly and to improve its generalization capability. Through EQ-learning, an action-value function is represented by the summation of weighted base functions, and an autonomous robot adjusts weights of base functions at learning stage. Other parameters (center coordinates, variance and so on) are adjusted at unification stage where two similar functions are unified to a simpler function.

1 Introduction

Reinforcement Learning has received a great attention as a method that trains an autonomous robot to make an appropriate action aimed at accomplishing a task with little or no premise knowledge. Q-learning[6], a well-known reinforcement learning, tries to learn an action-value function in order to find optimal policy with respect to some object functions.

Typically, even if an autonomous robot has continuous sensor values, they are quantized to reduce learning time. As a result, learning algorithms which work on the discrete space are easily designed and analyzed from a point of optimal control. However, it is costly to design a state space to accomplish a task smoothly. The method which constructs state space automatically by off-line learning is proposed by Asada and at el[1]. This is, without a good quantization of state space, reinforcement learning algorithms including Q-learning suffer from lack of robustness and lack of generalization.

To overcome the above, we propose Extended Q-learning(EQ-learning). EQ-learning is designed to accommodate continuous sensor space without quantization of it and to improve generalization capability of Q-learning. EQ-learning represents the action-value function by the summation of weighted base functions which is generally used to estimate a continuous value[3][5].

An EQ-learning algorithm has two stages called by learning stage and unification stage. A robot adjusts only weights of base functions through interactions at learning stage. In addition to that, at unification stage, a robot makes an effort to unify two similar base functions into one base function to obtain a simpler representation of action-value function. That is, at unification stage, based on

P. Stone, T. Balch, and G. Kraetzschmar (Eds.): RoboCup 2000, LNAI 2019, pp. 129-138, 2001.

two similar base functions, a new base function is calculated and two functions are replaced with it. A simpler representation caused by the unification leads to a robust model which works well even in a noisy environment as well as a general model which also works well in a slightly different environment.

Accordingly, our EQ-learning leads to a generalization of Q-learning and promises a much better learning performance capability than ordinary learning methods, including Q-learning. We performed simulation of two learning methods and preliminary results verified the higher performance capability of EQ-learning than Q-learning.

1.1 Q-learning

Q-learning algorithm is generally used as a method that give us a sophisticated solution to reinforcement learning problems.

If a robot transits from a state $s \in \boldsymbol{S}$ to a state $s' \in \boldsymbol{S}$ by selecting an action $a_i \in \boldsymbol{A}$, this algorithm updates the utility $Q(s, a_i)$ as,

$$Q(s, a_i) \Leftarrow Q(s, a_i) + \alpha(r(s, a_i) + \gamma M(s') - Q(s, a_i)), \tag{1}$$

$$M(s) = \max_{j \in A} Q(s, a_j), \tag{2}$$

where α is the learning rate and γ is the discounting factor.

The reinforcement signal $r(s, a_i)$ is accorded to a robot by an environment, when a robot accomplished a task. Thus, the reinforcement signal is added to the utility in the state which transited to "goal" by one action. The Q-learning algorithm updates the utility to minimize a temporal-difference of two utilities in current state and next state. As a result after a sufficient number of iteration, selecting an action a_i which has the maximum value of the utility $Q(s, a_i)$ in a state s is the optimal policy.

2 Extended Q-learning

2.1 Continuous action-value function

EQ-learning extends Q-learning in order to accommodate continuous sensor space directly. As a result, EQ-learning is able to minimize errors caused by the quantization of continuous sensor values.

EQ-learning represents the utility function by the summation of weighted base functions, as follows,

$$U(\boldsymbol{x}, a_i) = \sum_{m=1}^{N} W_m(a_i) B_m(\boldsymbol{x}). \tag{3}$$

To estimate the optimal utility (action-value) function, it is necessary to adjust these parameters (weight, variance, mean, and so on).

We consider that the reinforcement learning in the quantized sensor space is equivalent to applying the rectangular function as a base function. EQ-learning with rectangular function is equivalent to Q-learning.

$$B_m(\boldsymbol{x}) = \begin{cases} 1 & |\boldsymbol{\mu}_m - \boldsymbol{x}| \leq Width_m, \\ 0 & |\boldsymbol{\mu}_m - \boldsymbol{x}| > Width_m, \end{cases} \tag{4}$$

where $Width_m$ is the width of a rectangular function. The Gaussian function is applied as the base function too.

$$B_m(\boldsymbol{x}) = \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_m)^t \Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_m)). \tag{5}$$

Because these two functions have a locality feature, we choose them as a base function for EQ-learning.

2.2 Adjustment of weights of base functions

Generally, learning parameter increasing will also increase learning time[7]. The number of weight parameters is proportional to the number of base functions. But the number of center coordinate parameters and variance parameters are proportional to the number of base functions and sensor dimensions. Thus, in EQ-learning, an autonomous robot adjusts only weight of base functions in each step to decrease learning time.

The EQ-learning algorithm updates weights of base functions through interactions with a given environment and minimizes the temporal-difference. A robot transits from a state $\boldsymbol{x} \in \boldsymbol{X}$ to $\boldsymbol{x}' \in \boldsymbol{X}$ by selecting an action $a_i \in \boldsymbol{A}$, the temporal-difference $E(\boldsymbol{x}, a_i)$ is,

$$\begin{aligned} E(\boldsymbol{x}, a_i) &= (r(\boldsymbol{x}, a_i) + \gamma M(\boldsymbol{x}') - U(\boldsymbol{x}, a_i))^2, & (6) \\ &= (r(\boldsymbol{x}, a_i) + \gamma \max_{a_k \in A} U(\boldsymbol{x}', a_k) \\ &\quad - \sum_{m=1}^{N} W_m(a_i) B_m(\boldsymbol{x}))^2. & (7) \end{aligned}$$

We obtain a vector $\boldsymbol{e}$ which decreases the temporal-difference by partial derivative of $E(\boldsymbol{x}, a_i)$ with respect to $W_j(a_i)$, j is 1 to N, as follows,

$$\boldsymbol{e} = \left(\frac{\partial E(\boldsymbol{x}, a_i)}{\partial W_1(a_i)} \quad \frac{\partial E(\boldsymbol{x}, a_i)}{\partial W_2(a_i)} \quad \cdots \quad \frac{\partial E(\boldsymbol{x}, a_i)}{\partial W_N(a_i)} \right)^t. \tag{8}$$

Therefore, EQ-learning algorithm updates all of weights of base functions as,

$$\begin{aligned} W_m(a_i) \Leftarrow & W_m(a_i) + \\ & N_m(\boldsymbol{x})(r(\boldsymbol{x}, a_i) + \gamma M(\boldsymbol{x}') - V(\boldsymbol{x}, a_i)), \end{aligned} \tag{9}$$

where,

$$N_m(\boldsymbol{x}) = \alpha \frac{B_m(\boldsymbol{x})}{\sum_j B_j(\boldsymbol{x})}, \tag{10}$$

$$M(\boldsymbol{x}) = \max_{k \in A} V(\boldsymbol{x}, k), \tag{11}$$

where α is the learning rate and γ is the discounting factor.

2.3 Adjustment of other parameters

The more number of base functions is used, the more exactly approximation model can be obtained. However, increasing learning parameter will also increase learning time as well. Thus, in order to obtain an appropriate and simple model, a robot adjusts other parameters (the number of base functions, center coordinates, and variances)automatically by unification of two similar base functions into a single base function. We define that the similarity is calculated by a **K-L information number** and a **Maharanobis distance**.

The probability distributions in center coordinates of two base functions B_n and B_m are,

$$P(\boldsymbol{\mu}_n) = \{p(\boldsymbol{\mu}_n, a_1), \cdots, p(\boldsymbol{\mu}_n, a_K)\}, \tag{12}$$

$$P(\boldsymbol{\mu}_m) = \{p(\boldsymbol{\mu}_m, a_1), \cdots, p(\boldsymbol{\mu}_m, a_K)\}. \tag{13}$$

Thus, the K-L information number $I(P(\boldsymbol{\mu}_n); P(\boldsymbol{\mu}_m))$ of $P(\boldsymbol{\mu}_n)$ related to $P(\boldsymbol{\mu}_m)$ is calculated as,

$$I(P(\boldsymbol{\mu}_n); P(\boldsymbol{\mu}_m)) = \sum_{i=1}^{K} p(\boldsymbol{\mu}_n, a_i) \log \frac{p(\boldsymbol{\mu}_n, a_i)}{p(\boldsymbol{\mu}_m, a_i)}. \tag{14}$$

The Maharanobis distance $d_n(\boldsymbol{x})$ from $\boldsymbol{\mu}_n$ to $\boldsymbol{x}$ is calculated as,

$$d_n^2(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu}_n)^t \Sigma_n^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_n). \tag{15}$$

We define the similarity $s_n(m)$ from the base function B_n to the base function B_m as,

$$s_n(m) = \exp(-aI(P(\boldsymbol{\mu}_n); P(\boldsymbol{\mu}_m)) - bd_n^2(\boldsymbol{\mu} - m)), \tag{16}$$

where a and b are constants. During the learning, the robot unifies two base functions that have high similarity into a single base function to obtain a simpler representation of the utility function. If the parameter $S_{nm}(t)$ is larger than the threshold th, the unification of two base functions (B_n and B_m). The parameter $S_{nm}(t)$ is calculated as,

$$S_{nm}(0) = 0. \tag{17}$$

$$S_{nm}(t+1) = (1-\beta) S_{nm}(t) + \beta s_n(m), \tag{18}$$

where $0 \leq \beta \leq 1$ is the summation rate and t is increased when the robot is accorded with the reinforcement signal.

Based on two similar base functions, a new base function B_{new} is calculated and two base functions are replaced with it, as follows,

$$\boldsymbol{\mu}_{new} = \frac{d_m^2(\boldsymbol{\mu}_n)}{D_{nm}} \boldsymbol{\mu}_n + \frac{d_n^2(\boldsymbol{\mu}_m)}{D_{nm}} \boldsymbol{\mu}_m, \tag{19}$$

$$D = d_n^2((\boldsymbol{\mu}_m) + d_m^2((\boldsymbol{\mu}_n), \tag{20}$$

$$\boldsymbol{\Sigma}_{new} = \boldsymbol{\Sigma}_n + diag(\Delta x_i^2), \tag{21}$$

$$\Delta x_i = \mu_{new}(i) - \mu_n(i). \tag{22}$$

Weights of bases functions are calculated, as follows,

$$W_{new}(a_i) = W_n(a_i)B_n(\boldsymbol{\mu}_n) + W_m(a_i)B_m(\boldsymbol{\mu}_m). \tag{23}$$

By these algorithms, a robot can estimate the utility function by simpler functions which are obtained automatically.

2.4 Action selection rule

The simplest action selection rule is to select an action which has the highest action-value in each state. However, in learning phase, this rule has no chance to obtain a better policy than the current one. To overcome the above, generally, mapping a state $\boldsymbol{x}$ to an action a_i is stochastically, as follows,

$$\boldsymbol{P(x)} = (p(\boldsymbol{x}, a_1), p(\boldsymbol{x}, a_2), \cdots, p(\boldsymbol{x}, a_K)). \tag{24}$$

$$0 \leq p(\boldsymbol{x}, a_i) \leq 1, \tag{25}$$

$$\sum_{i=1}^{K} p(\boldsymbol{x}, a_i) = 1. \tag{26}$$

We decide that the robot is in a state $\boldsymbol{x}$, an action $a_i \in \boldsymbol{A}$ is selected stochastically according to Boltzmann distribution, as follows,

$$p(\boldsymbol{x}, a_i) = \frac{\exp(U(\boldsymbol{x}, a_i)/T)}{\sum_{j=1}^{K} \exp(U(\boldsymbol{x}, a_j)/T)}, \tag{27}$$

where T is the scaling constant and utility function $U(\boldsymbol{x}, a_i)$ is the utility of selecting an action a_i in a state $\boldsymbol{x}$.

3 Simulation and result

In our simulations, we train an autonomous robot to get a ball faster which is allocated randomly in a given environment. Therefore, the robot creates the utility function to get the ball through EQ-learning. In this simulation, the robot can identify the ball through the camera and select one of the given five actions (see Figure 1). The reinforcement signal accorded by the environment is,

$$r(\boldsymbol{x}, a_i) = \begin{cases} 1 \text{ if the robot gets the ball,} \\ 0 \qquad\qquad \text{otherwise.} \end{cases} \tag{28}$$

The other parameters for robot learning are, as follows,

$$\alpha = 0.25, \quad \gamma = 0.99, \quad T = 0.1. \tag{29}$$

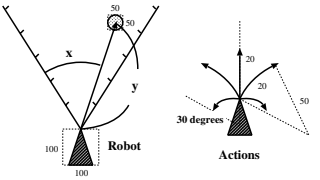


Fig. 1. Robot parameters in our simulation.

In order to assess the capability of our EQ-learning, we report results of a robot that can speed up to get a ball in several environments by checking average steps to carry out the task. These assessment run 1000 times at each number of trials(10, 20, 30, 40, 50, 100, 150, 200) to calculate these average steps to get a ball allocated randomly.

3.1 Simulation experiment

To assess our EQ-learning, we have two set of experiments. For the first set of experiment, a robot adjusts only weights of rectangular functions. This algorithm is equivalent to an ordinary Q-learning algorithm. At the second set, through EQ-learning, the robot adjusts weights and unifies base functions automatically. Therefore, the robot creates the utility function approximation by adjusting all parameters (weights, center coordinates, variances, and the number of base functions). In the first case, 16 (4×4) base functions are allocated in the sensor space, and the robot learns and adjusts these parameters. In the second set, the Gaussian function is used as the base function. In this experiment, parameters for the unification are, as follows,

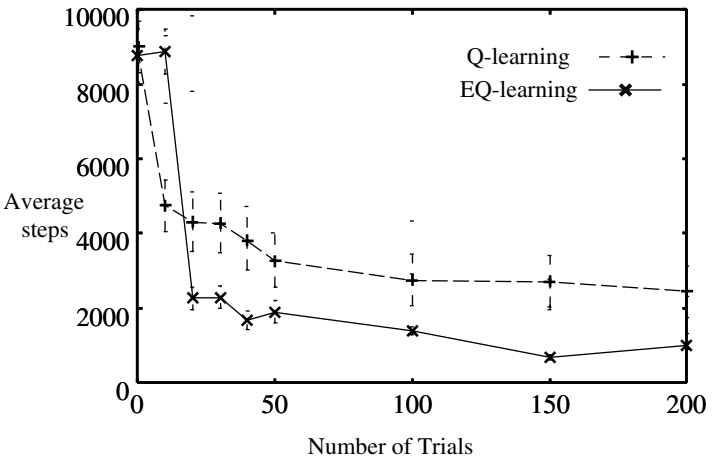
$$a = 1, \quad b = 0.5, \quad \beta = 0.1, \quad th = 0.25. \tag{30}$$

3.2 Result

Average steps and the example of unification of base functions are plotted against trial number in Figure 2. This figure shows that EQ-learning is useful to estimate the utility function. Because of unification, parameters that a robot must adjust decrease in an appropriate number. That is, the robot can get a simpler and appropriate model for learning to get the ball. An example of learning model is shown below in Figure 2. The figure shows the allocation of base functions. Through EQ-learning, an autonomous robot creates the policy which is equivalent to the natural hand-designed solution (if ball is on left, go left; if on right, go right; if straight ahead, go straight).

4 Experiment for the real robot and the result

RoboCup[4] provides a common task for AI and robotics. We try to train a real robot(Figure 3) to learn the behavior of a goalie robot (intercepting space



Number of trials	0	50	100	200
Average numbers of base functions	16	4.2	2.8	2.5
Example of the unification process				

Fig. 2. Result of the reinforcement learnings and the number of base functions at each number of trials (above), and the example of unification (below).

between the ball and the goal) by EQ-learning. The goalie robot has one video camera to identify and track colored objects (ball, goal, and other robots). There is a problem that the goalie should watch not only front, but also left and right side. There are some solutions, however, we decide to equip the goalie with omni-directional vision system. The goalie is able to select an action out of five actions, go left(and right) slowly or quickly, and stop.

4.1 Experiment

In this experiment, we observe that an appropriate model for keeping the goal is obtained automatically by EQ-learning. The goalie extracts the regions of a goal and a ball from the input image by using color information. There are two vectors,$\boldsymbol{v}_b$ and $\boldsymbol{v}_g$, one is from the center of input image (goalie robot) to the center of the ball's region and the other is to the goal's one. We then define that the angle made from two vectors is state space. If the angle is,

$$0 \leq x \leq 90, \quad or \quad 270 \leq x \leq 360, \tag{31}$$

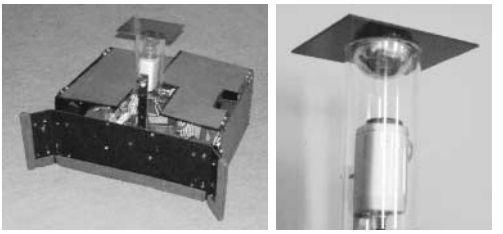


Fig. 3. Real keeper robot which has one omni-direction camera.

it assumes that the ball is in the goal. Thus, the goalie doesn't learn in this state. The reinforcement signal accorded by the environment is,

$$r(x,a) = \begin{cases} 1 & |180 - x(t)| < 20, \\ 0 & \text{otherwise.} \end{cases} \tag{32}$$

In this experiment, parameters for the unification are, as follows,

$$a = 1, \quad b = 0.5, \quad \beta = 0.1, \quad th = 0.25. \tag{33}$$

First allocation of base functions is shown in Figure 5.

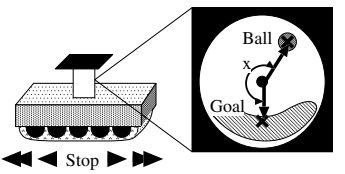


Fig. 4. The goalie can select an action and sense the environment to keep the goal.

4.2 Result

Through EQ-learning, the allocation of base functions is changed into simpler one as shown in Figure 6 and Figure 7. The goalie robot creates an appropriate model and policy which are equivalent to the natural hand-design solution for keeping a goal. The robot has the following policies: (1)go left quickly, (2)go left slowly, (3)go right slowly, (4)go right quickly, which correspond to four base functions in Figure 7 from left to right respectively. As the result, it is expected that the cost to create the learning model and learning time will reduce.

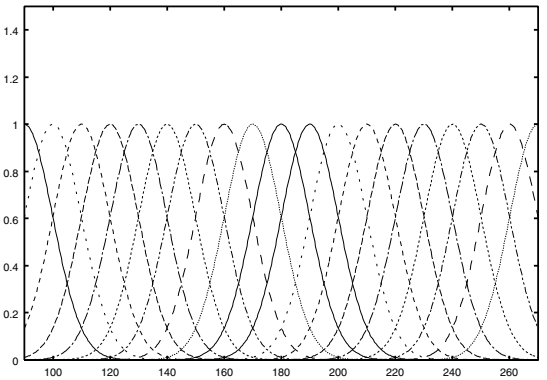


Fig. 5. First allocation of base functions.

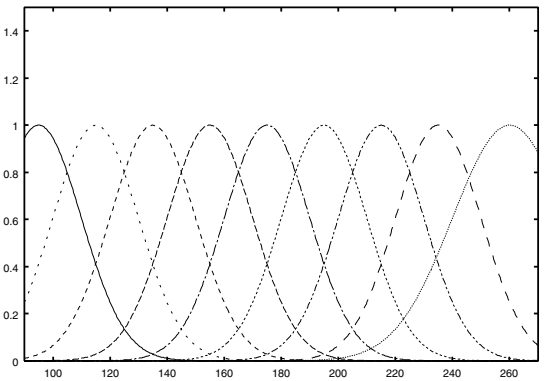


Fig. 6. Simple learning model for keeping a goal.

5 Conclusion and future work

We propose Extended Q-learning that estimates an action-value function by weighted base functions and leads appropriate parameters (weights, center coordinates, variances, and the number of base functions) automatically through the interaction with the environment. As a result, EQ-learning can deal with reinforcement learning algorithm with no quantization of continuous sensor values, and estimate a utility function much better than the ordinary one. Since EQ-learning has the robustness against errors that occurred from the noise of robot sensor parameters, we believe that EQ-learning can deal much better with real robot learning.

Through EQ-learning, we can train a robot to keep the goal. It, however, is fairly simple behavior. The behavior of the soccer needs more complex policy. Therefore, it is not easy for an applied EQ-learning to get the behavior of soccer

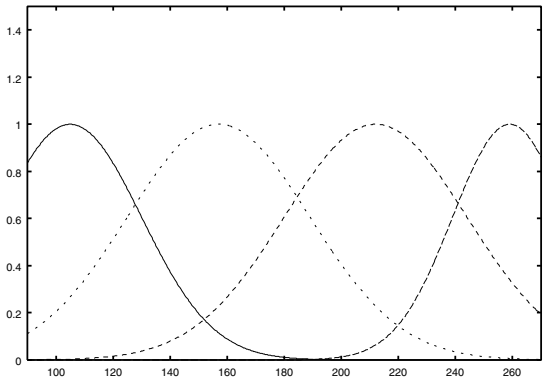


Fig. 7. Simple learning model for keeping a goal.

directly. Thus, we think that hierarchical control structures[2] are useful for this problem. We constitute hierarchical control structures which consists of modules called a subtask and rely on the programmer to design a hierarchy of modules. We train a robot to carry out each subtask by reinforcement learning so that the robot learns the behavior of soccer. We also design the hierarchical control structures to obtain high performance behavior of soccer.

References

1. M. Asada, S. Noda and K. Hosoda : Action Based Sensor Space Segmentation For Soccer Robot Learning. Applied Artificial Intelligence, Vol.12, No.2-3 (1998) 149–164.
2. Digney, B. L. : Learning Hierarchical Control Structures for Multiple Tasks and Changing Environments. From animals to animats 5 : SAB 98,(1998).
3. Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson : Neuronlike Adaptive Elements that can solve difficult learning control problems. (1983) 834–846.
4. Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara, : RoboCup : A Challenge Problem for AI and Robotics. Lecture Notes in Artificial Intelligence Vol.1395 (1998)1–19.
5. Norihiko Ono, Yoshihiro Fukuta : Learning Coordinated Behavior in a Continuous Environment. Lecture Notes in Artificial Intelligence Vol.1221 (1997) 73–81.
6. C.J.C.H. Watkins : Learning from delayed rewards. PhD thesis, University of Cambridge (1989).
7. Whitehead : A complexity analysis of cooperative mechanisms in reinforcement learning. Proc. AAAI-91 (1991) 607–613.