A Study On Applying A Reinforcement Learning For A Real Robot

Shoichi Noda, Minoru Asada, Sukoya Tawaratsumida, and Koh Hosoda
Dept. of Mech. Eng. for Computer-Controlled Machinery
Osaka University, 2-1, Yamadaoka, Suita, Osaka 565, Japan
noda@robotics.ccm.eng.osaka-u.ac.jp

Abstract Reinforcement learning is a useful method to acquire a purposive behavior with little or no *a priori* knowledge about the environment. In applying the reinforcement learning to a real robot, there are many difficult problems, such as long learning time and construction of its state and action space. In this paper, we show a soccer robot which learns to shoot a ball into the goal using the Q-learning, one of the reinforcement learning methods. We give the results of computer simulation and real robot experiments that show the effectiveness of a policy transfer from the computer simulation to the real robot.

Key Words Reinforcement learning, Q-learning

1 Introduction

Reinforcement learning has recently been receiving increased attention as a method for robot learning with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors[1]. However, almost of existing works in the reinforcement learning have only shown computer simulations, and only a few real robot applications are reported, which are simple and less dynamic.

We aim to realize an autonomous robot by using the reinforcement learning. In this paper, we show a soccer robot which learns to shoot a ball into the goal by the method of Q-learning. In applying the reinforcement learning, we have difficult problems how to reduce the long learning time which requires the exponential order in the size of the state space[2] and how to construct the state and action spaces for a complex real enviornment.

In general, the state space should be constructed such that each action causes a state transition. On a real robot, however, one physical action does not always causes a state transition discriminated by physical sensors. To cope with this "state-action deviation" problem, we constructed an action set in such a way that one action consists of a series of the same action which is successively executed until the current state changes. To speed up the learning time, we propose an algorithm which we call Learning from Easy Missions (or LEM). This algorithm reduces the learning rate from the exponential order in the size of the state space to about the liner order.

We implemented the experiments on both computer simulation and real robot. A *Policy transfer* from the computer simulation to the real robot system is not only effective to save the learning time, but also useful to check differences between both and to improve the whole system.

2 Q-learning

Before getting into the details of our system, we briefly review the basics of the Q-learning. For more through treatment, see [3]. We follow the explanation of the Q-learning by Kaelbling [4].

We assume that the robot can discriminate the set $\boldsymbol{S}$ of distinct world states, and can take the set $\boldsymbol{A}$ of actions on the world. The world is modeled as a Markov process, making stochastic transitions based on its current state and the action taken by the robot. Let $T(s, a, s')$ be the probability that the world will transit to the next state s' from the current state-action pair (s, a). For each state-action pair (s, a), the *reward* $r(s, a)$ is defined.

The general reinforcement learning problem is typically stated as finding a policy that maximizes discounted sum of the reward received over time. A policy f is mapping from $\boldsymbol{S}$ to $\boldsymbol{A}$. This sum called the *return* and is defined as:

$$\sum_{n=0}^{\infty} \gamma^n r_{t+n}, \tag{1}$$

where r_t is the reward received at step t given that the agent started in state s and executed policy f. γ is the discounting factor, it controls to what degree rewards in the distant future affect the total value of a policy and is just slightly less than 1.

Given definitions of the transition probabilities and the reward distribution, we can solve the optimal policy, using methods from dynamic programming [5]. A more interesting case occurs when we wish to simultaneously learn the dynamics of the world and construct

the policy. Watkin's Q-learning algorithm gives us an elegant method for doing this.

Let $Q^*(s,a)$ be the expected return or *action-value function* for taking action a in a situation s and continuing thereafter with the optimal policy. It can be recursively defined as:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q^*(s',a'). \tag{2}$$

Because we do not know T and r initially, we construct incremental estimates of the Q values on line. Starting with $Q(s,a)$ at any value (usually 0), every time an action is taken, update the Q value as follows:

$$Q(s,a) \Leftarrow (1-\alpha)Q(s,a) + \alpha(r(s,a) + \gamma \max_{a' \in A} Q(s',a')). \tag{3}$$

where r is the actual reward value received for taking action a in a situation s, s' is the next state, and α is a leaning rate (between 0 and 1).

3 Learning from Easy Missions

In order to realize a real robot using the reinforcement learning, it is a indispensable problem to speed up the learning time. There are several works to speed up it in applying the reinforcement learning. Mahadevan and Connenl [6] proposed a task decomposition method for rapid task learning on real robot. Whitehead proposed the Learning with an External Critic (or LEC) algorithm [2] that the robot receives an advise in each state from the external critic. When the task or the environment is not so simple, however, it is difficult to decompose the whole task into subtasks, and to give a proper advise for each state.

We construct the learning schedule such that the robot can learn in easy situations at the early stage and learn in more difficult situations at the later stage. We call this *Learning from Easy Missions* (or LEM).

Fig.1 The simplest state space.

Instead of critical analysis of the time complexity for LEM, we give an intuitive explanation for it by using a very simple example. Following the complexity analysis by Whitehead [2], we assume the "homogeneous" state space uniformly k-bounded with polynomial width of the depth k and zero-initialized Q-learning. Further, we assume that state transition is deterministic and the robot can take m-kinds actions with equal opportunities. In order to figure out how many steps are needed to converge the Q-learning, we use $O(k)$ state space and simplify the convergence such that the action value function converged if it is updated from the initial value (0) [1].

Fig.1 show an example of such state spaces. Since we assigned a reward 1 when the robot achieves the goal and 0 otherwise, the unbiased Q-learning takes long time. From the above formulation, it needs m trials to transit from the initial state S_k to the state S_{k-1} in the worst case, therefore it takes m^k trials to achieve the goal for the first time in the worst case, and the value of the action value function for only the state S_1 is updated. Next, it needs m^{k-1} trials to update the value of the action value function for the state S_2, and totally it need $(m^k + m^{k-1} + \cdots + m)$ trials to converge the value of the action value function for all the states. Therefore, the unbiased Q-learning can be expected time moderately exponential in the size of k [2].

While, in the LEM algorithm, we set the agent at the state S_1 first and make it try to achieve the goal. In the worst case, it takes m trials. Then, we set the agent at the state S_2 and repeat it. In the worst case, it needs $m \times k$ trials to converge the action value function. Therefore, the LEM algorithm requires at most linear in the size of k.

In actual situations in which the state transition is often stochastic, the state space is not homogeneous, and therefore it seems difficult to correctly decide which state is an easy one to achieve the goal and when to shift the initial situations into more difficult ones. Since the convergence to the optimal policy is guaranteed in the Q-learning scheme, we roughly collect the easy states S_1 in which the agent can achieve the goal with high probability and shift to a slightly more difficult situations when

$$\Delta Q_t(\mathbf{S_1}, a) < \epsilon \ , \qquad 0 < \epsilon \ll 1 \tag{4}$$

where

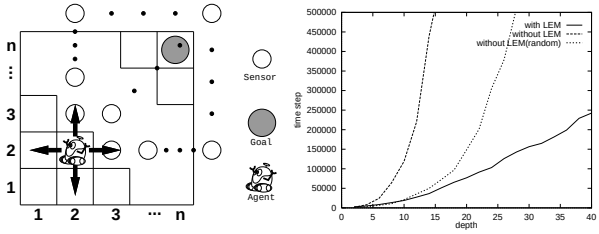
$$\Delta Q_t(\mathbf{S_1}, a) = \sum_{s \in \mathbf{S_1}} |\max_{a \in \mathbf{A}} Q_t(s,a) - \max_{a \in \mathbf{A}} Q_{t-1}(s,a)|$$

The search time versus maximum distance k for a simple *get food* problem in the 2-D grid environment (**Fig.2**(a)), where one step Q-learning algorithm is applied, is shown in **Fig.2**(b). As we expected, the search time with LEM is almost linear in the size of k while that of the normal Q-learning without LEM (initial position is fixed in the left-lower corner or random) indicates the exponential order in the size of k.

4 Task and Assumptions

The task for a mobile robot is to shoot a ball into the goal as shown in **Fig.3**(a). The environment

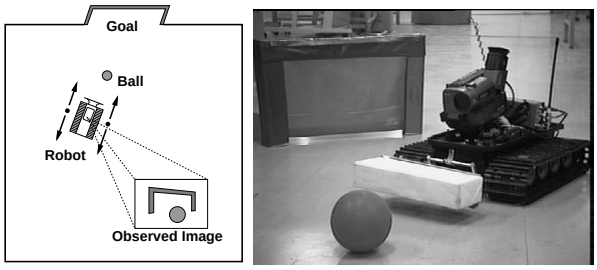
[1] Strictly speaking, this might be incorrect, however, it seems easy to figure out the order of the search time.



(a) 2-D grid environment.

(b) Search time complexity as a function k.

Fig.2 Simple get-food problem.



(a) The task is to shoot a ball into the goal.

(b) A picture of the radio-controlled vehicle.

Fig.3 Task and our real robot.

consists of a ball and a goal, and a mobile robot has a single TV camera. The robot does not know the location and the size of the goal, size and weight of the ball, any camera parameters such as focal length and tilt angle, or kinematics/dynamics of itself. The image positions of the ball and/or the goal are able to be tracked continuously. The robot can select an action to be taken.

Fig.3(b) shows a picture of the real robot with a TV camera and video transmitter.

5 Learning Scheme

In order to apply the Q-learning scheme to the task, we define a number of sets and parameters as follows.

(a) a state set S

Only the information the robot can obtain about the environment is the image supposed to be capturing the ball and/or the goal. The ball image is quantized into 9 sub-states, combinations of three positions (left, center, and right) and three sizes (large (near), middle, and small (far)). The goal image has 27 sub-states, combinations of three parameters each of which is quantized three parts. Each sub-state corresponds to one posture of the robot toward the goal, that is, position and orientation of the robot in the field. In addition to these 243 (27×9) states, we add other states such as these cases in which only the ball or the goal is captured in the image. Totally, we have 319 states in the set $\boldsymbol{S}$.

(b) an action set A

The robot can select an action to be taken against the environment. In real system, the robot moves around the field by a PWS (Power Wheeled Steering) system with two independent motors. Since we can send the motor control command to each of two motors independently, we quantized the action set in terms of two motor commands ω_l and ω_r, each of which has 3 sub-actions (forward, stop, and back motions, respectively). Totally, we have 9 actions in the action set $\boldsymbol{A}$.

The existing applications of the reinforcement learning schemes have constructed the state and action spaces in such a way that each action causes the state transition (ex. one action is forward, backward, left, or right, and states are encoded by the locations of the agent) in order to make the quantization problem (the structural credit assignment problem) easy. This makes a gap between the computer simulations and real robot systems. Each space should reflect the corresponding physical space in which a state or an action can be perceived or taken. However, such construction of state and action spaces sometimes causes a "state-action deviation" problem. To cope with this, the robot continues to take one action until the current state changes.

(c) a reward and a discounting factor γ

We assign a reward value 1 when the ball was entered into the goal or 0 otherwise. This makes the learning very time-consuming. Although adopting a reward function in terms of distance to the goal state makes the learning time much shorter, it seems difficult to avoid the local maxima of the action-value function Q.

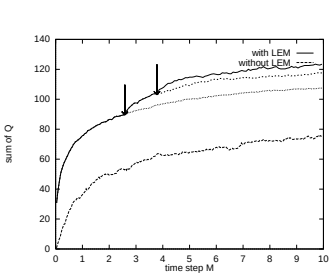
A discounting factor γ is used to control to what degree rewards in the distant future affect the total value of a policy. In our case, we set the value a slightly less than 1 ($\gamma = 0.9$).

6 Experiments

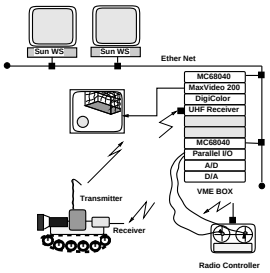
The experiment consists of two parts: first, learning the optimal policy f through the computer simulation, then apply the learned policy to a real situation.

6.1 Simulation

We performed the computer simulation with the following specifications (the unit is correspond to cm). The field is a square of which side length is 200. The goal post is located at the center of the top line of the square and its height and width are 23 and 90, respectively. The robot is 31 wide and 45 long and kicks a ball of diameter 9. The camera is horizontally mounted on the robot (no tilt), and its visual angle is 36 degrees. These parameters are selected so that they can roughly simulate the real world.



(a) Change of the sum of Q-values.



(b) A configuration of the real system.

Fig.4 Simulation result and real system.

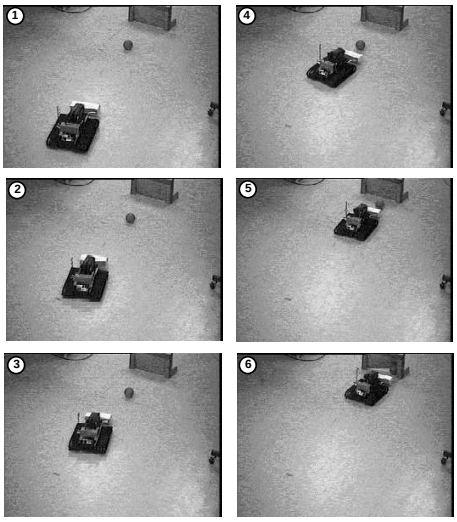


Fig.5 The robot succeeded in shooting a ball into the goal.

Following the LEM algorithm, we began the learning of the shooting behavior by setting the ball and the robot near the goal. Once the robot succeeded in a shooting, the robot begin to learn (the sum of Q is increasing), but after that the robot wonders again in the field. After many iterations of these successes and failures, the robot learned to shoot a ball into the goal when the ball is near the goal. After that, we set the ball and the goal slightly further from the goal, and repeat the robot learning again.

Fig.4(a) shows the change of the sum of Q-values with or without LEM. The Q-learning with LEM is much better than that without LEM. The arrows indicate the time to change the initial states from S_1 in which the robot can shoot easily, to more difficult states S_2, and then S_3.

6.2 Real System

Fig.4(b) shows a configuration of the real mobile robot system. The image taken by a TV camera mounted on the robot is transmitted to a UHF receiver and processed by Datacube MaxVideo 200, a real-time pipeline video image processor. In order to simplify and speed up the image processing time, we painted the ball in red and the goal in blue. We constructed the radio control system of the vehicle, following the remote-brain system[7]. The image processing and the vehicle control system are operated by VxWorks OS on MC68040 CPU which are connected with host Sun workstations via Ether net. We have shown a picture of the real robot with a TV camera (Sony handy-cam TR-3) and video transmitter in **Fig.3**(b).

Fig.5 shows a sequence of the shooting images in which the robot succeeded in shooting a ball into the goal.

7 Discussion and Future Works

We have shown a vision-based reinforcement learning on real robot system, which adopted the Learning from Easy Missions algorithm to speed up the learning rate instead of task decomposition. The state-action deviation problem due to the peculiarity of the visual information is pointed out as a problem in applying the Q-learning to real robot applications, and we constructed an action space to cope with this problem.

Although we roughly constructed the state space, it is still a difficult problem how to construct it. Higher resolution of the state space contributes to the higher performance of the robot behavior, but the learning time becomes unrealistically longer and vice versa. It seems necessary to take into account the purpose and action of the robot in constructing an appropriate state space. Now, we are considering the construction of action-based state space for robot behavior acquisition.

References

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

[2] S. D. Whitehead. "A complexity analysis of cooperative mechanisms in reinforcement learning". In *Proc. AAAI-91*, pages 607–613, 1991.

[3] C. J. C. H. Watkins. *Learning from delayed rewards"*. PhD thesis, King's College, University of Cambridge, May 1989.

[4] L. P. Kaelbling. "Learning to achieve goals". In *Proc. of IJCAI-93*, pages 1094–1098, 1993.

[5] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[6] J. H. Connel and S. Mahadevan. "Rapid task learning for real robot". In J. H. Connel and S. Mahadevan, editors, *Robot Learning*, chapter 5. Kluwer Academic Publishers, 1993.

[7] M. Inaba. "Remote-brained robotics: Interfacing ai with real world behaviors". In *Preprints of ISRR'93*, Pitsuburg, 1993.