Reinforcement Learning
When Visual Sensory Signals are Directly Given as Inputs

Katsunari Shibata Yoichi Okabe
Research Center for Advanced Science and Technology (RCAST), Univ. of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 113, JAPAN
shibata@okabe.rcast.u-tokyo.ac.jp okabe@okabe.rcast.u-tokyo.ac.jp

Abstract

It is shown that a neural-network based learning system, which obtains visual signals as inputs directly from visual sensors, can modify its outputs by reinforcement learning. Even if each visual cell covered only a local receptive field, the learning system could integrate these visual signals and obtain a smooth evaluation function. It also represented the spatial information smoothly in the hidden layer through the learning, and the area of the state which seemed important for the system was magnified in the hidden neurons' space. The learning is so adaptive that when different motion characteristic was employed in the system, the representation became different from the previous one, even if the environment is the same.

1. Introduction

Reinforcement learning has been focused for these days. Some learning methods are proposed to evaluate each state from delayed reinforcement signals and system's experience[1][2][3]. Visual sensory signals, from which we can get various pieces of information, has been tried to be used in the learning[4]. However, visual signals were pre-processed and the present state was made correspondence to one of the pre-prepared states in state space. Then the mapping from the state to motions was trained by reinforcement learning. Accordingly, it is difficult for the system to make a continuous mapping from sensory signals to motions, and also difficult to change the configuration of the state space adaptively.

Here it is tried to be examined if a system can obtain appropriate motions directly from visual signals through reinforcement learning. Here Temporal Smoothing Learning Based Reinforcement Learning[3] is employed. That is because Temporal Smoothing Learning has been confirmed to have an ability to integrate local sensory signals into an analog spatial signal[5][6].

2. Temporal Smoothing Learning Based Reinforcement Learning

Temporal Smoothing Learning is a simple learning algorithm to make the output curve smooth along time as shown in Fig. 1. In this learning, the absolute value of the second time derivative of the output is trained to be reduced, and the output curve becomes close to a straight line. Then the output has one-to-one corresponding to the time. This means that the output represents temporal information. This learning algorithm cannot be used only for estimating necessary time to get a reward in reinforcement learning, but also can be used for integrating the signals from many sensory cells, each of which has a only local receptive field like retina[5][6]. For example, when an object simply oscillates in the visual field and the signals from the local visual cells which are arranged in a row, are given as input of the neural network, the output becomes to represent the object location through this learning.

A learning system as shown in Fig. 2, is supposed here to process reinforcement learning. That is composed of motion generator and state evaluator. Since the sensory signals are inputs for both parts, these two components are made as one layered neural network actually. This means that the neural network has two kinds of outputs, that are motion outputs and an evaluation output, and each output neuron connects to all hidden neurons without any discriminations.

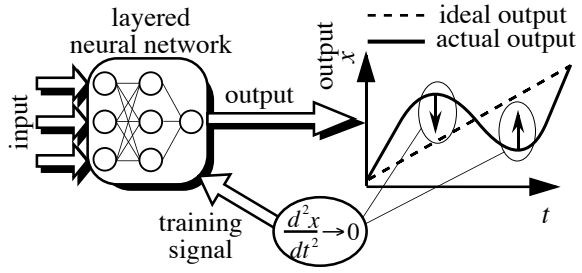


Fig. 1 Temporal Smoothing Learning

Here since one target state (reward) is assumed to be chosen at each time, reinforcement learning can be thought as a learning to minimize the necessary time until a system arrives at the target state. The state evaluator is trained to predict the necessary time to get the reward. For this learning, Constant Evaluation Slope Learning, which is an extension of Temporal Smoothing Learning, is employed here. In this learning, the ideal change of the evaluation value for a time unit, is calculated from the maximum necessary time Nmax as

$$\Delta\Phi ideal = \Phi amp / Nmax \qquad (1)$$

where Φamp : ideal amplitude of evaluation value. Here the value range of a neuron output is from -0.5 to 0.5, Φamp is set to be 0.4-(-0.4)=0.8. For adaptability Nmax is calculated as

$$\begin{aligned} Nmax[i] &= (1-1/\tau)Nmax[i-1] && \text{if } Nmax[i-1]>N[i] \\ &= N[i] && \text{otherwise} \end{aligned} \qquad (2)$$

where $N[i]$: necessary time at the i-th trial, τ : large time constant. Then by comparing the change of the actual evaluation value to this ideal one, the evaluation value at previous time $\Phi(t\text{-}1)$ is trained by the training signal as

$$\Phi s(t\text{-}1) = \Phi(t\text{-}1) - \eta\,(\Delta\Phi ideal - \Delta\Phi(t)) \qquad (3)$$

where Φs : training signal for evaluation value, $\Delta\Phi(t) = \Phi(t)-\Phi(t\text{-}1)$, and η : a training constant. By this learning, evaluation curve along time becomes smooth and the slope of the curve becomes constant. When the system arrives at the target state, the evaluation value is trained to be 0.4. This learning can be thought as the special case of TD type reinforcement learning[1] in which the discount factor γ for calculating a weighted sum of reinforcement signals is 1.0 and the system has always a small penalty. It also can be thought as the TD type reinforcement learning using straight line on behalf of exponential curve for representing the necessary time until a given target state.

The system makes motions according to the sum of the outputs of motion generator $\boldsymbol{m}$, and random numbers $\boldsymbol{rnd}$ as trial and error factors. Motion signals $\boldsymbol{m}$ are trained by the training signals as

$$\boldsymbol{ms} = \boldsymbol{m} + \zeta\, \boldsymbol{rnd}\, \Delta\Phi \qquad (4)$$

where ζ : a training constant. By this learning, motion is trained for the system to get more change of evaluation value. This learning is processed in parallel with the evaluation learning. The neural network is trained by Back Propagation learning[7] according to the training signals as Eq. (3) and (4) at each time step.

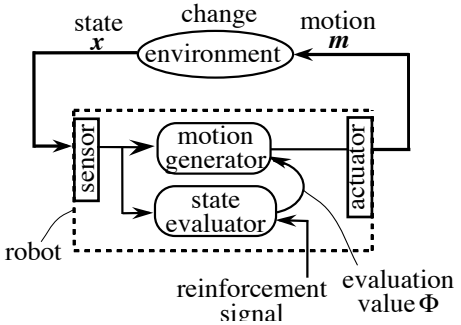


Fig. 2 Structure of reinforcement learning system

3. Simulation

Here the problem that a locomotive robot with two wheels and two visual sensors gets object, was adopted as shown in Fig. 3. Each visual sensor had 24 visual cells, that were arranged in a row, and had a total of 180 degree of visual field. Each visual cell had only a local receptive field without overlapping, and outputed the area ratio occupied by the projected object in the receptive field. This robot coould get a reward only when the robot got the object, in other words, when the center of the object went through the robot. The diameter of the target was 1.0 and length of the robot was 2.0. Figure 4 shows the signal

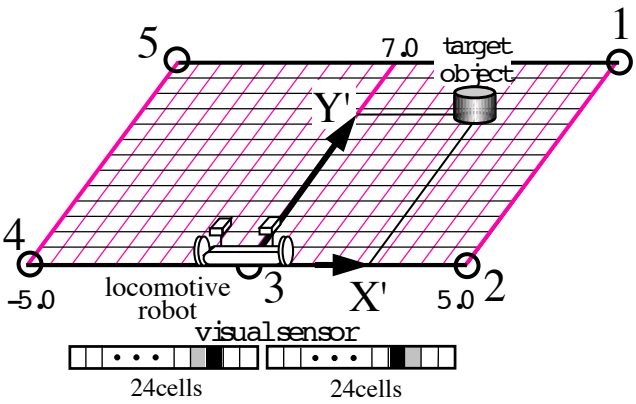


Fig. 3 Simulation environment

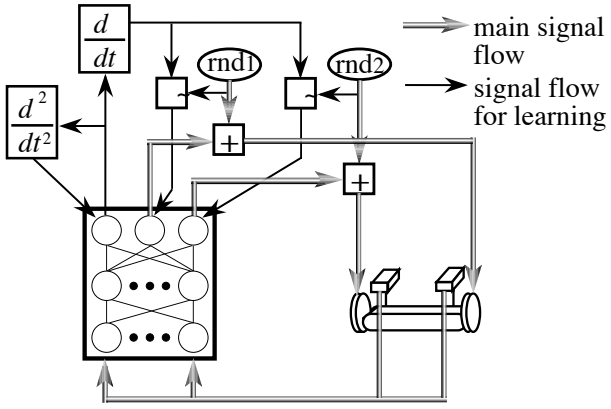


Fig. 4 Signal flow of the simulation

flow of this simulation. Before learning, the input-hidden connection weights were set to be small random numbers, and the all hidden-output connection weights were set to be 0.0. Then all the output values of the neural network were 0.0 before learning. In this case, number of outputs for motion signals were two, and the robot rotateed its wheels according to the outputs. In the early phase, since the robot could move only by random numbers, the object was located within the range that was close to the robot. According to the progress of learning, the range of the initial object location becomed wider gradually. After the robot got the object or missed the object, that means the robot could not catch the object image without getting the object, the object located at another place chosen randomly. If the robot missed the object, the evaluation output was trained to be -0.4. The sequence from the initial state to the object state is defined one trial.

Figure 5 shows the evaluation surface as a function of the object location and the robot locus on the robot-fixed coordinates after 700 trials and Fig. 6 shows those after 30000 trials. Because of robot-fixed coordinates, the object moves relatively in behalf of the robot. In these figures, X' and Y' shows how far the target exists in the lateral and forward direction for the robot respectively. In Fig. 5, the ridge of evaluation surface has tendency to spread radially. That is because, the local receptive field of each visual cell spreads radially. However, in Fig. 6, the contour line becomes smooth and the necessary time for the robot to get the target becomes shorter.

When we use the TD-type reinforcement learning, we can obtain similar result. However, the density of the contour line for the evaluation surface is higher when the object exists close to the robot, and the density is low when the object is far from the robot. In the following section, only the results when Temporal Smoothing type reinforcement learning is applied are mentioned.

4. Examination of spatial information coding in hidden neurons

The coding of spatial information in hidden neurons tries to be examined. The neural network as shown in Fig. 7 is prepared and reinforcement learning is performed using the first three output neurons. After the learning, the last output neuron, which is shown as hatched circle in Fig. 7, is trained by supervised learning. The last output neuron connects to the all hidden neurons with 0 connection weight initially. Visual signals for 6 object locations that are shown as white and black circles in Fig. 8, are put into the neural network in order, and the network is trained by the training signal respectively as shown in Fig. 8 using Back-Propagation learning[7]. The training signal depends only on the object location in X' direction.

Figure 8 (a) shows the output distribution for the object location by color after the learning, and Fig. 8 (b) shows that when the reinforcement learning was not processed. We can see in Fig. 8 (a) that the output

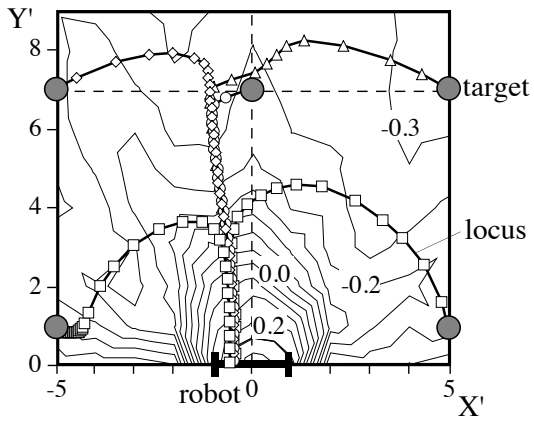


Fig. 5 Evaluation surface (contour line) and robot locus after 700 trials in the robot-fixed coordinates

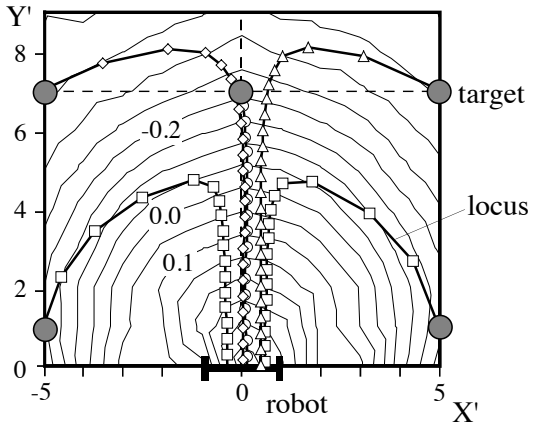


Fig. 6 Evaluation surface (contour line) and robot locus after 30000 trials in the robot-fixed coordinates

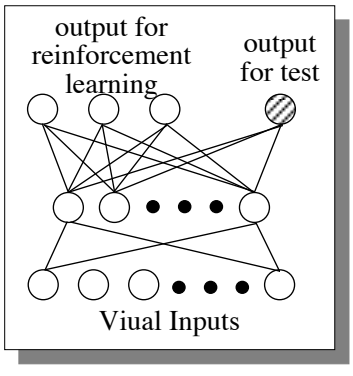


Fig. 7 Neural network to check the learning of hidden neurons

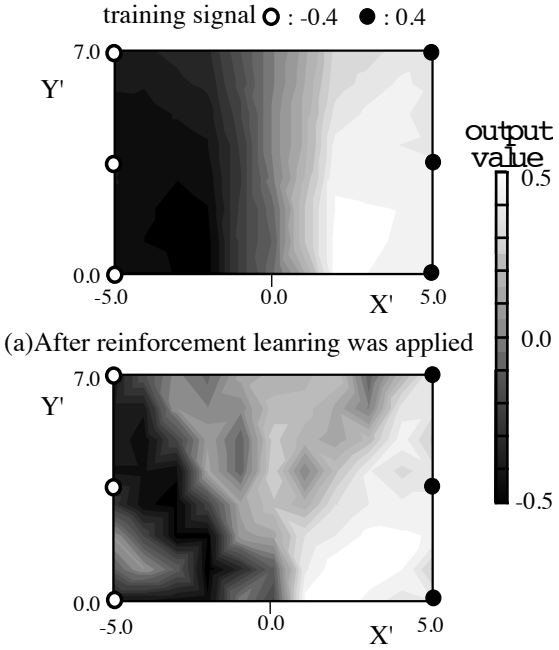


Fig. 8 Comparison of output distribution after supervised learning when reinforcement learning was applied beforehand or not.

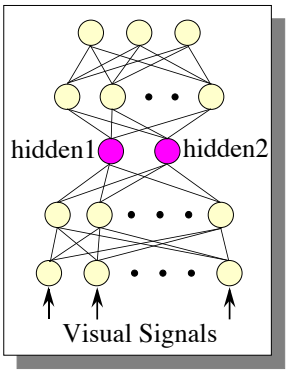


Fig. 9 Neural network to examine the coding of hidden neurons

represents the spatial information smoothly and it looks that the robot can code in hidden neurons if the object is located in the left or in the right. On the other hand, the contour of the output in Fig. 8(b) spread radially. This means that the hidden neurons can code the spatial information smoothly through the reinforcement learning.

Next, the neural network, one of whose hidden layer has only two neurons as shown in Fig. 9, is prepared, and the coding of the two hidden neurons through reinforcement learning are examined. Figure 10 shows the

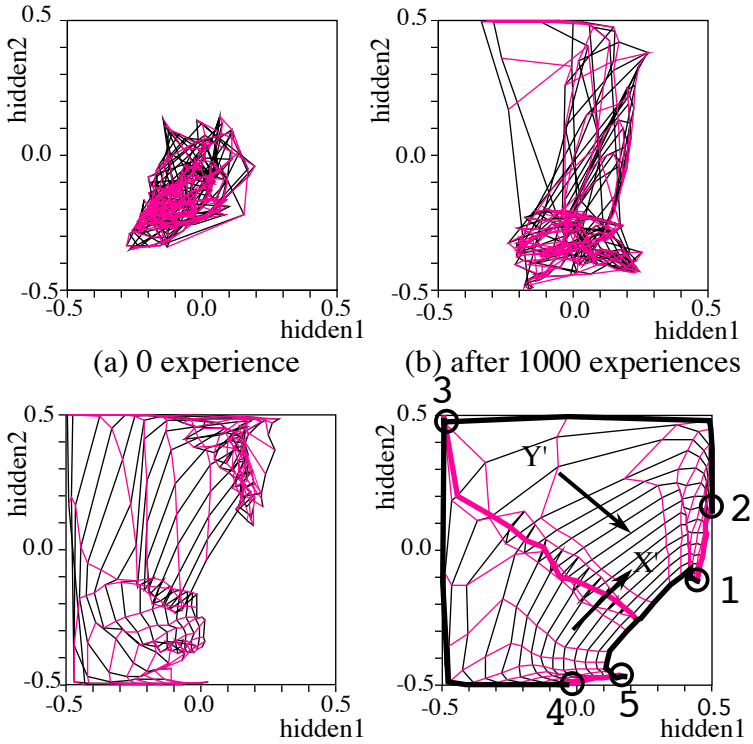


Fig. 10 Coding of object location in hidden neurons' space. Circles in (d) are typical object locations corresponding to the circles in Fig. 3.

change of the coding of spatial information in hidden neurons. Each axis shows the value of each hidden neuron. The state of the hidden neurons are plotted when the object exists on each lattice in Fig. 3. In Fig. 10 (d), the states of the hidden neurons for the 5 typical object location are shown as numbered circles. We can see that the area that the hidden neurons used becomes gradually larger, and finally almost all space of hidden neurons was used. We can also see that the area of state around Y'=0.0 (2-3-4) is larger than that around Y'=7.0 (1-5). Especially, the area of the state around (X', Y')=(-1.0, 0.0) or (X', Y')=(1.0, 0.0) is magnified. That is because such area is very important for the robot because there is a boundary which decides the robot can get the object or not.

6 Change of Spatial Information Coding depending on Motion Characteristics

Here asymmetrical motion characteristics of the robot is employed. As shown in Fig. 11, the robot rotates right wheel according to the product of the corresponding output and 3.0. In this case, the learning using the same neural network, is not so stable. Then, the evaluation surface and the locus are shown in Fig. 12 when the robot can get the object comparably faster. Here

the ridge of the evaluation surface extends slightly to the left, and when the object is located at the right side of the robot initially, it rotates clockwise until it can see the object in the left forward and then goes forward. This motion is good for the robot because it can move right wheel three times more than the left wheel and it goes to left forward direction faster, but the robot goes to the right forward direction slowly. When the object comes close to the robot, it gets the object in the right hand side. That is also because it can rotate right wheel more and get object faster in the right side.

Figure 13 shows the state of hidden neurons. We can see that the range of the state around the ridge of the evaluation surface is magnified. That is the important area for the robot because there is the boundary where the robot have to change the motion signals. We can say that the hidden neurons can code the object location smoothly and effectively for the robot. If many hidden neurons are used, the learning becomes stable. It can be thought that spatial information cannot be represented well in the rectangle space, whose two axes show the values of two hidden neurons, when the robot has an asymmetrical motion characteristic.

5. Conclusion

It is shown that visual signals can be used directly in the reinforcement learning by using Constant Evaluation Slope Learning and a layered neural network. Hidden neurons of the neural network can code the spatial information by integrating visual input signals. It is also shown that the area that is important for the system is magnified in the hidden neurons' space.

References

[1] Barto, A. G., Sutton, R. S. and Anderson C. W., "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems", *IEEE Trans. SMC-13*, pp.835-846, 1983

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

[3] K. Shibata and Y. Okabe : A Robot that Learns an Evaluation Function for Acquiring of Appropriate Motions, *Proc. of WCNN '94 San Diego*, Vol.2, pp. II-29 - II-34, 1994

[4] Asada, M., Noda, S. and Tawaratsumida, S., "Purposive Behavior Acquisition for a Robot by Vision-Based Reinforcement Learning", *Proc. of MLC-COLT Workshop on Robot Learning*, 1994

[5] K. Shibata and Y. Okabe : Unsupervised Learning Method to Extract Object Locations from Local Visual Signals, *Proc. of ICNN '94 Orlando*, Vol. 3, pp.1556-1559, 1994

[6] K. Shibata and Y. Okabe : Integration of Local Sensory Signals and Extraction of Spatial Information Based on Temporal Smoothing Learning, *Jounal. of JNNS*, Vol. 3, No. 3, pp.98-105, 1996 (in Japanese)

[7] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. : Learning representation by back-propagating errors, *Nature*, Vol. 323, No. 9, pp.533-536, 1986

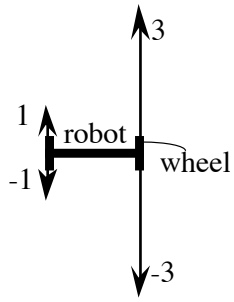


Fig. 11 Asymmetrical motion characteristics of the robot

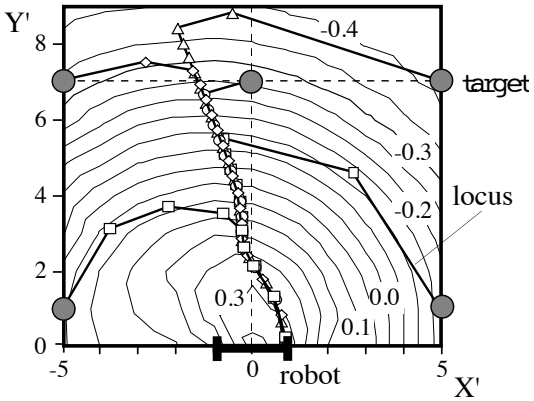
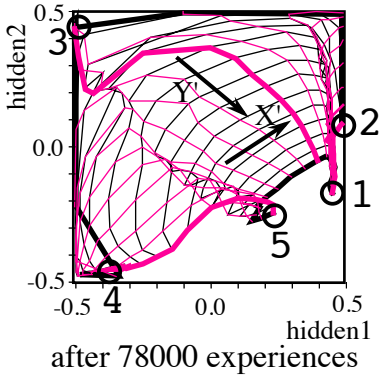


Fig. 12 Evaluation curve and robot locus on robot-fixed coordinates when the robot has asymmetrical motion characteristics



after 78000 experiences

Fig. 13 Coding of the object location in hidden neurons' space when the asymmetrical motion characteristics were employed in the robot.