A Robot that Learns an Evaluation Function for Acquiring of Appropriate Motions

Katsunari Shibata and Yoich Okabe
Research Center for Advanced Science and Technology, Univ. of Tokyo
4-6-1 Komaba, Meguro-ku, 153 Tokyo JAPAN
shibata@okabe.rcast.u-tokyo.ac.jp

Summary

An unsupervised learning method of an evaluation function is proposed. By this function, a robot can learn a series of motion to achieve a given purpose. The evaluation function is calculated by a neural network and time derivative of this function is reduced during motions with random trials. We confirmed by simulation that a robot with asymmetric motion feature could obtain an appropriate asymmetric evaluation function and become to achieve a given purpose along an almost optimal path.

1. Introduction

In famous Skinner's experiment, a mouse in Skinner's box becomes to be able to push the button to get a piece of food after some trials as shown in Fig. 1[1]. However, the mouse could not know that it could get a piece of food by pushing the button before the mouse was put into the box. At early stage, the mouse moves randomly, and when it pushes the button unexpectedly, it gets a piece of food. From the experience, the possibility that the mouse pushes the button, slightly becomes large. Then after many experiences, the mouse has an intention to push the button. This means that the mouse in Skinner's experiment did not know only the way to push the button, but also were not given any evaluation functions about getting the food. From this fact, a robot also can obtain an evaluation function through its experiences. Then we assumed a locomotive robot as shown in Fig. 2. The robot has only a purpose to get a target.

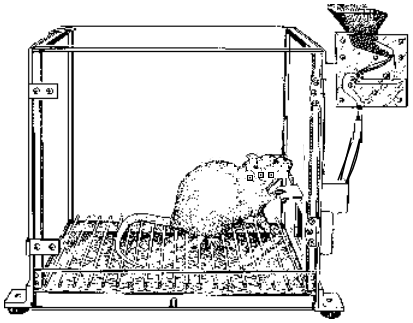


fig.1 Skinner's experiment

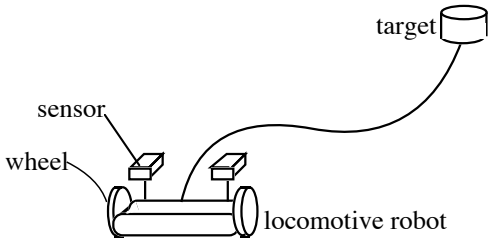


fig. 2 An assumed robot that has a purpose to get a target

The mouse in Skinner's experiment, can learn the evaluation function and motion by the interaction between the mouse and the environment without any supervisors. When we make a robot learn an appropriate series of motion using the interaction, we usually give the robot an evaluation function that we made[2][3]. The evaluation value of the present state is calculated from the sensor signals through the evaluation function. If the robot continues to move towards the gradient direction of the evaluation function, it will be able to achieve the given purpose. For example, in the case of the robot in Fig. 2, an evaluation function can be the inverse of the distance between the robot and the target. As shown in Fig. 3, the robot moves and gets sensor signals. Then the evaluation function makes the supervisor signal for the motion creator from the sensor signals. The motion creator, that is made by a layered type neural network, learns by the supervisor signal, and become to create an appropriate motion signal.

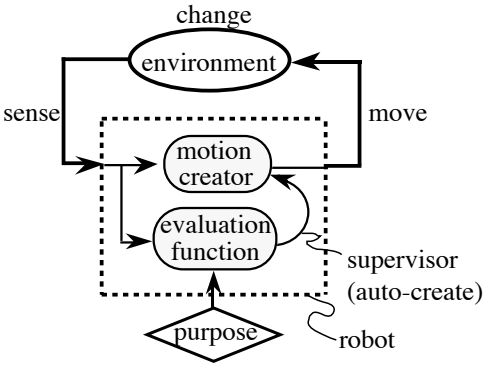


fig. 3 Learning system and interaction to the environment

Then we tried to make a robot learn both an evaluation function and a series of motion to achieve a given purpose. We cannot know if the evaluation function, that we give to a robot, is the best for the robot and the environment. Furthermore, if the environment changes, the evaluation function also has to change. For these reasons, the robot has to learn its evaluation function for a good adaptation to the environment.

In this paper, we describe the way to learn the evaluation

function without any supervisors, and the way to learn the evaluation function and the motion in parallel. Finally we show the result for comparing which series of motion was closer to the optimal route when the robot was given a fixed evaluation function or when the robot learned the evaluation function by itself.

2. Learning Method of a Series of Motion

Some ways to learn a series of motion from an evaluation function has been already proposed[2][3]. We indicate the way that we employed. Here, the motion signal is calculated by a layered type neural network. Let us suppose the 3 dimensions of space as shown Fig. 4. X, Y axes show sensor signals respectively. The other axis shows the evaluation value. An evaluation value is decided for each set of sensor signals and the surface in this figure is an evaluation function surface. The robot moves according to the sum of the output of a neural network and a small uniform random number as follows,

$$\boldsymbol{M} = \boldsymbol{m} + \boldsymbol{r} \qquad (1)$$

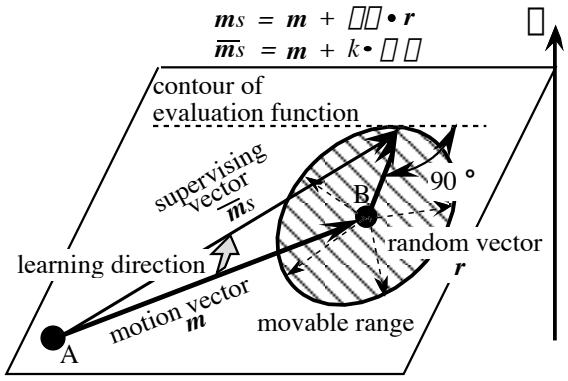


fig.4 Motion learning

where **M** : actual motion vector, ***m*** : motion signal vector and ***r*** : uniform random number vector. If the robot has two or more actuator, the neural network has two or more output neurons and each output is added by a different random number. The circle filled with oblique line in Fig. 4 shows the movable range of the robot by the random vector ***r***. The robot moves and can get a new evaluation value using the evaluation function. Then the robot makes the following supervisor signal,

$$\boldsymbol{m}_S = \boldsymbol{m} + \boldsymbol{r}\,\Delta Q \qquad (2)$$

where $\boldsymbol{m}_S$: supervisor signal vector for motion signal, $Q(t)$: evaluation function and $\Delta Q = Q(t) - Q(t-1)$. The neural network learns at every time unit by this supervisor signal, using a conventional supervised learning algorithm like Back Propagation Method[4]. By the learning through many experiences, the direction of the robot motion vector becomes close to the gradient direction of the evaluation function in average as $\overline{m_s}$ in Fig. 4. Then the robot can learn a series of motion close to the optimal route under the evaluation function.

3. Unsupervised Learning Method of an Evaluation Function

Let us think how should the evaluation function be made. Supposing the 3 dimensions of space in which there is a robot and a target as shown in Fig. 5 (a), we can think of an evaluation method using the distance between the robot and the target. However, how to normalize on each axis cannot be known easily. Furthermore if there is an area that the robot cannot move through, like an obstacle as shown in Fig. 5 (a), it is meaningless to evaluate only by the distance. Then we propose to evaluate by necessary time for arriving at the target. If there is an obstacle, the robot can move beside the area and arrive at the target. If we project the route on the time axis as shown in Fig. 5 (b), we can evaluate under only one criterion and there are no necessity to normalize each axis any longer.

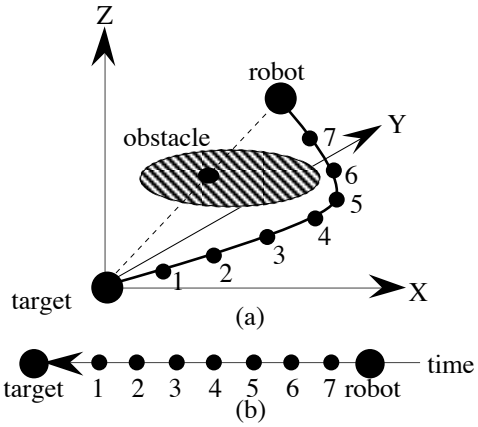


fig.5 State evaluation method
(a) evaluation on space
(b) evaluation on time axis

However, if the robot modifies the evaluation function by tracing back the route to the starting point when the robot arrives at the target, the robot must remember all the states it has existed. In order to avoid wasting memory, we propose a real-time learning method of the evaluation function. The evaluation function should be as follows. As shown in Fig. 6, the output of the function is small in an initial state, the output becomes larger as time goes by, and finally the output becomes the largest value when the robot gets the target. Here, the evaluation function is calculated by a layered type neural network and the output function of each neuron is sigmoid between 0.0 and 1.0. The learning method has three phases. When the robot is in an initial state, to decrease the output of the evaluation function, the

network learns by the supervisor signal as

$$\square_s(x(t)) = \square(x(t)) - \square \quad (3)$$

where $\square_s(x(t))$: supervisor signal for the evaluation function and $\square$: small constant value. Here we employ 0.01 as $\square$. When the robot arrives at the target, the network learns by the supervisor signal as

$$\square_s(x(t)) = \square_{max} \quad (4)$$

where $\square_{max}$: the maximum value of the evaluation function (different from the maximum of the output function). Here we employ 0.9 as the maximum value $\square_{max}$. Since it is the best state when the robot arrives at the target, the supervisor signal can be a constant value. However, since the initial state is not always the furthest state from the purpose, we cannot set up a constant value, and we set up a slightly small value than the actual output as the supervisor signal. Otherwise from the initial state to the target, in order to increase the output smoothly, the supervisor signal is as

$$\square_s(x(t)) = \square(x(t)) + \square \frac{d^2\square(x(t))}{dt^2} \quad (5)$$

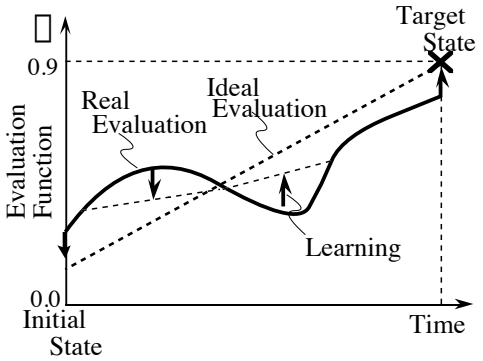


fig.6 Learning method that realizes an evaluation on time

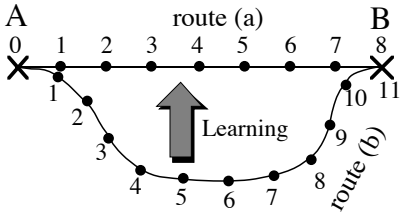


fig.7 An example of optimizing route

where $\square$: a smoothing constant. Here we employ 0.5 as the smoothing constant $\square$. One thing we have to take care is that the learning is on job learning and the network learns only one iteration using conventional supervised learning algorithm like Back Propagation Method. The output does not become to the value of the supervisor signal, but it becomes closer to the supervisor signal slightly.

Let us assume that a robot moves along the route (b) in Fig. 7 when the robot goes from a point A to a point B. It takes 11 time unit, but if it moves along the route (a) it can arrive at the point B in 8 time unit. By the learning of the evaluation function, it becomes smooth around the route (b). On the other hand, $d\square/dt$ along the route (a) is larger than that along the route (b) because the robot can arrive at the point B faster. By the learning of motion, the motion changes to the gradient direction of the evaluation function. The route, that robot moves along, becomes slightly close to the route (a). By repeating the learning of the evaluation function and the learning of motion, the robot becomes to move along the route (a). We can see that the combination of the two learning processes makes the route close to the optimal one.

4. Structure and Flow of the Learning

To realize the combination of the two learning processes, we employ the structure as shown in Fig. 8. The large rectangle shows the robot brain and there are two neural networks. One of them is to calculate the evaluation function and the other is to calculate the motion signal. Each of them calculates the output from the sensor signals, and so we can use only one neural network for both purposes. The robot moves according to the sum of the output of the motion network and a random number. The supervisor signal for the evaluation neural network is usually made from the second order deviation of the output of itself as Eq.(5). The supervisor signal for the motion network is made from the product of the first order deviation of the evaluation output and the random number that was used for creating a motion. Then the each

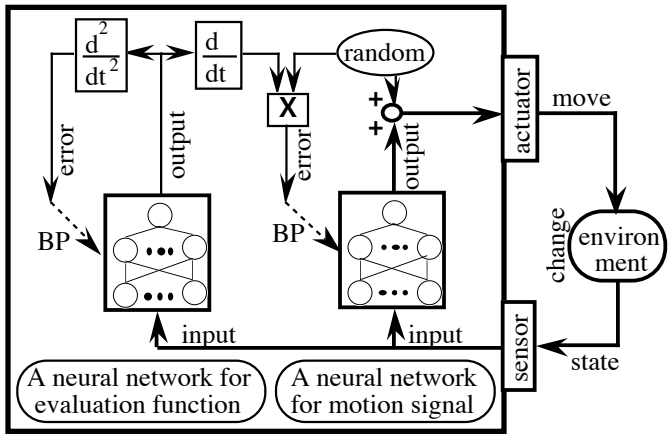


fig.8 Network structure and learning method

network learns only one iteration by the supervisor signal respectively at every time unit. Fig. 9 shows the flow of the learning. First the robot is placed somewhere, and the evaluation network learns by the supervisor signal as Eq.(3). Then the robot repeats a motion as Eq.(1) and learning of both neural networks by the supervisor signals as Eq.(2) and Eq.(5) respectively until it arrives at the target. When it arrives, the estimation network learns by the supervisor signal as Eq.(4). Then the process is repeated many times.

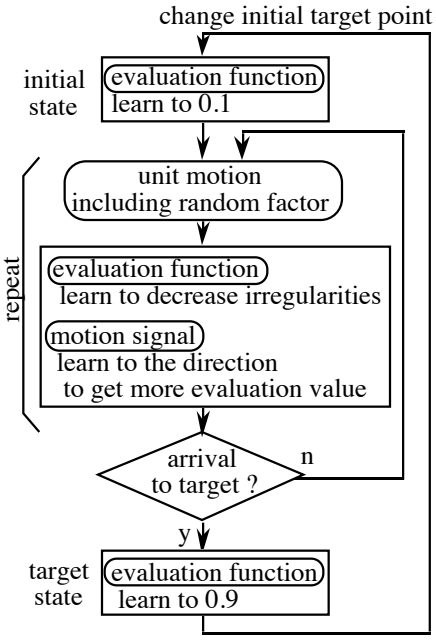


fig.9 Learning procedure

5. Simulation

In this section, we show the simulation result. We assumed the environment as shown in Fig. 10. The locomotive robot has two wheels and can move them independently. The rotate angle of each wheel is decided by the sum of each output of the motion network and a random number. In order to make the environment in which we can make the evaluation function easily, the ratio of the rotation angle against the motion signal is three times larger in the right wheel than in the left wheel. The range of rotating angle of the right wheel is also three times wider that of the left wheel as shown in Fig. 11. The robot also has two sensors. One of them sensed the lateral distance from the robot to the target and the other one sensed the forward distance. Initially a target was placed randomly at the edge of the rectangle except for the edge that the robot existed. It is drawn by a thick line in Fig. 10. The robot moved and if the target went backward passing under the robot, we assumed that the robot could catch the target. If the target went backward passing beside the robot, we assumed that the robot failed to catch the target. In this simulation, when the robot failed to catch the target, the evaluation neural network learned by the supervisor signal 0.1. We defined one experience from placing the target on the initial position to the success or failure to catch the target. However, the robot can move only randomly at first phase and cannot arrive at the target in a finite time. Then we moved the target close to the robot when the robot could not arrive after some time. In order to compare, we do another simulation. We gave the robot an evaluation function that we can think of easily, and make the robot learn only the motion signal.

Then we show the change of the evaluation function in Fig. 12. Each graph in Fig. 12 is drawn on robot centered coordinates. The robot was placed at the center of the coordinates and corresponding evaluation value was plotted on Z axis when the target was placed on each lattice. The contour of the evaluation function is also drawn. Figure 12 (d) shows the evaluation function that we gave the robot in the comparison simulation. Figure 13 shows the contour of the evaluation function and the motion between the robot and the target when the target was placed on each of 5 places. They are also drawn on the robot centered coordinates but on the 2 dimensional surface. Though the robot moved and target did not move actually, the robot was fixed and the target moved in these figures because of the robot centered coordinates. On the locus of the target, the points were plotted at every 10 time unit. In these graphs except for the graph (a), the target moved more when the target was far from the robot and moved to the tangential direction. The reason is that when the robot rotates actually, the target relatively moves according to the product of the rotation angle and the distance on the robot centered coordinates. We also show the loci on the absolute coordinates in Fig. 14. Figure 14(a) shows the comparison between the locus after 5000 experiences and that after 30000 experiences. Figure 14(b) shows the comparison to the case of the fixed evaluation function after 30000 experiences. On the locus of the robot, the points are also plotted at every 10 time unit. After 1000 experiences, the evaluation function did not have an enough expansion of value range and the robot could not get the target as shown in Fig. 13(a). After 5000 experiences, the evaluation function had a sharp shape and the robot could arrive at the target from any initial positions. The robot got the target at the right half of the robot. The reason is that the right wheel could rotate 3 times more than left wheel and the robot could get the target soon when the target was in the right. After 30000 experiences,

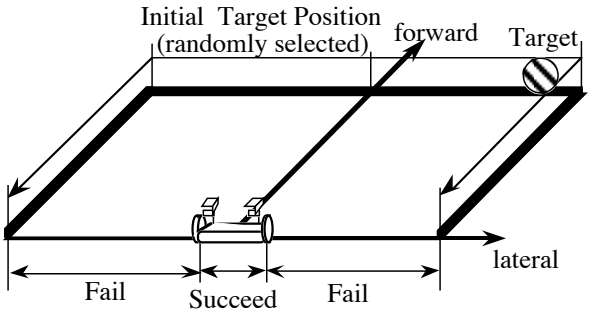


fig.10 Environment in the simulation

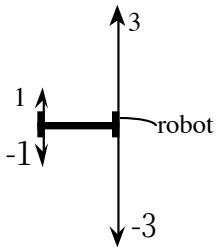
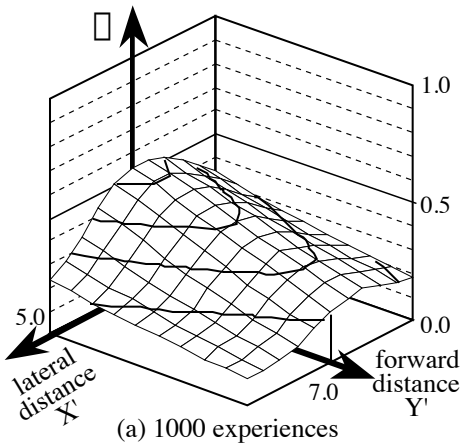
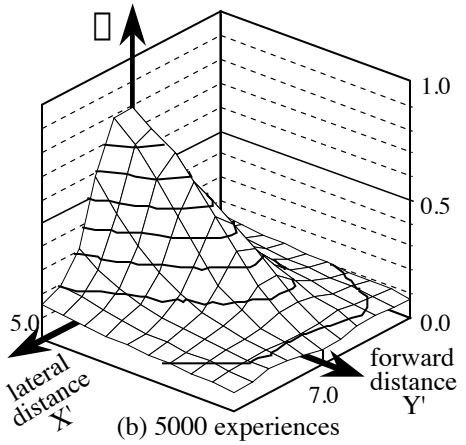


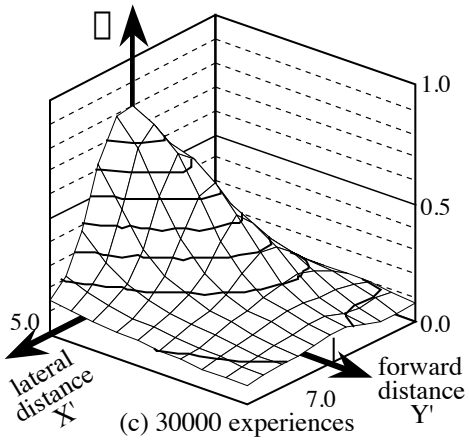
Fig.11 Asymmetric robot motion



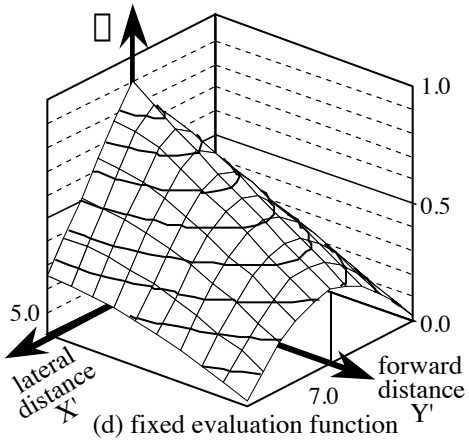
(a) 1000 experiences



(b) 5000 experiences

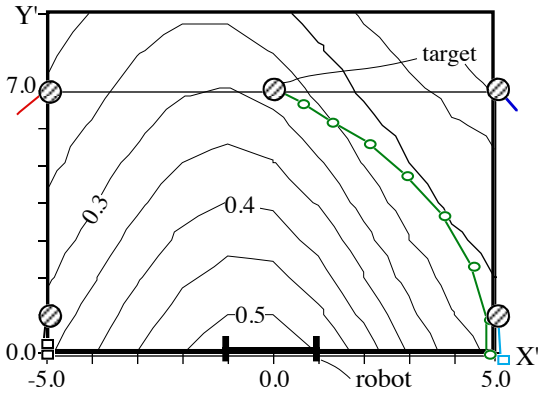


(c) 30000 experiences

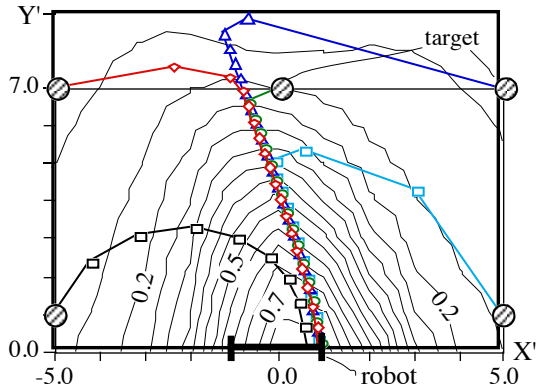


(d) fixed evaluation function
(30000 experiences)

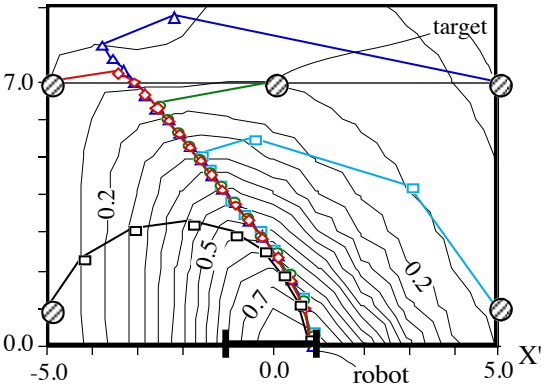
fig.12 Evaluation function (on robot centered coordinates)



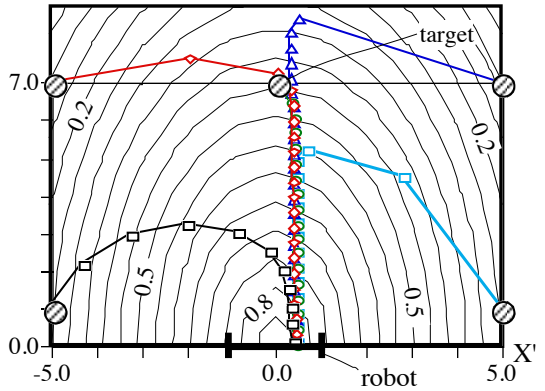
(a) 1000 experiences



(b) 5000 experiences



(c) 30000 experiences



(d) fixed evaluation function
(30000 experiences)

fig.13 Evaluation function and the locus
(on robot centered coordinates)

the

ridge of the evaluation function moved to the left and the target moved along the ridge. It means that the robot approached the target with looking at it in the left. In the absolute coordinates as shown in Fig. 14(a), the locus of the robot became like an arc after 30000 experiences. Furthermore, the robot went back with rotating itself to the right when the target was seen in the right direction. So, the locus after 5000 experiences seems better than that after 30000 experiences at a glance. However, the necessary time was more after 5000 experiences. The evaluation function and the locus could be reflected the characteristic of the robot. On the other hand, the evaluation function in a comparison simulation is symmetric shape as shown in Fig. 12(d), and the locus of the target expanded in front of the robot as shown in Fig. 13(d). When we see the Fig. 14(b), the locus in the case of fixed evaluation function is close to a straight line and it seems better than that in the case of self-organized evaluation function at a glance. Also in this case, the robot could arrive at the target faster in the case of self-organized evaluation function. Then we show the change of the necessary time for the robot to arrive at the target in Fig. 15. The time is the average of the 12 trials in which the target was placed at a regular interval on the edge of the rectangle. If the robot failed in a trial, the time is set 1000. In this figure, we show the results of 4 simulations. We did two simulations in which the initial weight value in the neural network was different from each other in the case of the fixed evaluation function and in the case of the self-organized evaluation function respectively. We can see that the robot, using the fixed evaluation function, could become to get the target faster than in the case of the self-organized evaluation function in an early stage. The reason is that it took some time to form the evaluation function in the case of the self-organized evaluation function. We can also see from the Fig. 15 that after many experiences the evaluation function and the route of the robot were optimized in the case of the self-organized evaluation function and realized the better route than in the case of the fixed evaluation function.

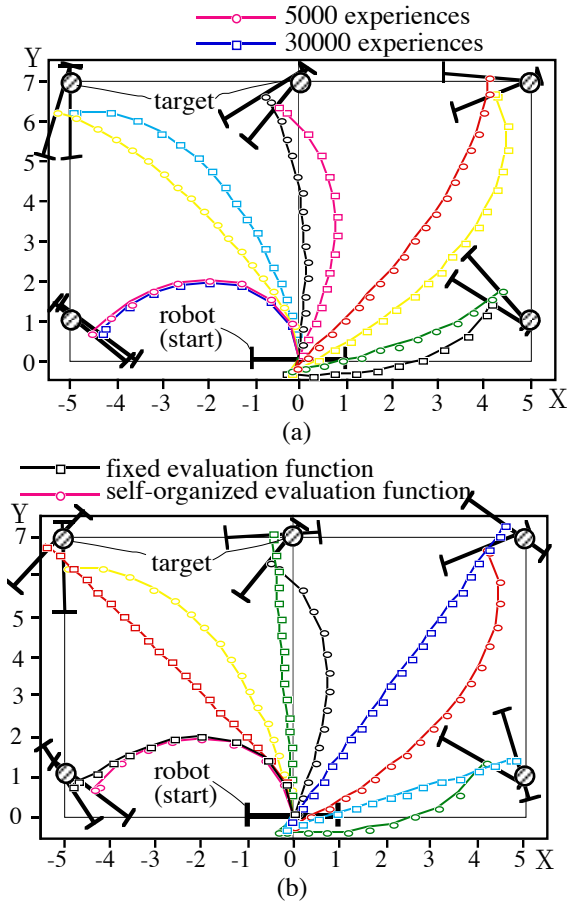


fig.14 Locus of the robot (on absolute coordinates)

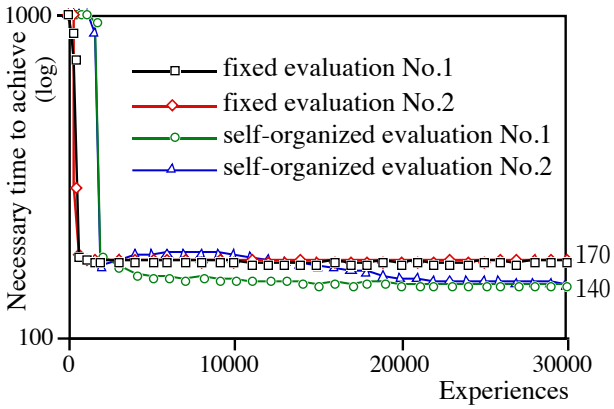


fig. 15 Change of the necessary time to get the target

6. Conclusion

We have proposed an unsupervised learning method by which a machine forms an appropriate evaluation function and a series of motion through its experiences. We confirmed in a simulation that the evaluation function obtained by the learning method was reflected the characteristic of the environment. Then the robot became to be able to move along with the optimal route.

Reference

[1]Skinner B.F., "Cumulative Record", Appleton-Century-Crofts, 1961

[2]Yoda,H., Miyatake,T. and Matsushima, H., : "A Kinematic Model with a Self-Learning Capability of a Series of Motions Based on Instinctive Incentive", *Trans. of IEICE,* **J73-D-II**, 7,1027-1034, 1990

[3]Nakano, K., Doya, K., SICE'84, S1-3, 1984 (in japanese)

[4]Rumelhart D.E., Hinton G.E. and Williams R.J.: "Learning representations by back-propagating errors", *Nature*, **323**, 9, 533-536, 1986