Hindawi Publishing Corporation
Journal of Robotics
Volume 2010, Article ID 437654, 9 pages
doi:10.1155/2010/437654

Research Article

Emergence of Prediction by Reinforcement Learning Using a Recurrent Neural Network

Kenta Goto and Katsunari Shibata

Department of Electrical and Electronic Engineering, Oita University, 700 Dannoharu, Oita 870-1192, Japan

Correspondence should be addressed to Kenta Goto, kenta.0510@gmail.com

Received 6 November 2009; Revised 1 March 2010; Accepted 17 May 2010

Academic Editor: Noriyasu Homma

To develop a robot that behaves flexibly in the real world, it is essential that it learns various necessary functions autonomously without receiving significant information from a human in advance. Among such functions, this paper focuses on learning "prediction" that is attracting attention recently from the viewpoint of autonomous learning. The authors point out that it is important to acquire through learning not only the way of predicting future information, but also the purposive extraction of prediction target from sensor signals. It is suggested that through reinforcement learning using a recurrent neural network, both emerge purposively and simultaneously without testing individually whether or not each piece of information is predictable. In a task where an agent gets a reward when it catches a moving object that can possibly become invisible, it was observed that the agent learned to detect the necessary factors of the object velocity before it disappeared, to relay the information among some hidden neurons, and finally to catch the object at an appropriate position and timing, considering the effects of bounces off a wall after the object became invisible.

1. Introduction

Unlike factories and laboratories, the real world is too complicated and diverse for robots to behave flexibly while following some specific programs. To develop robots that behave flexibly like humans in the real world, the robots must autonomously learn in various environments and acquire necessary knowledge and functions by themselves. The acquired knowledge and functions enable the robots to behave more appropriately, even in unfamiliar environments. Living beings acquire various functions and achieve appropriate purposes by using these skills. "Prediction" is one such function. It is a higher function that estimates a future state from the past and present states considering both dynamics of the environment and actions of the robot.

For developing highly intelligent robots, recent focus has been on autonomous learning of prediction. When we predict some information, we can usually know in the future whether or not the prediction was correct. For a learning system that predicts a future state from the present and past states and actions, this signifies that training signals can be obtained in the future even though they are not provided by humans. In this sense, the learning of prediction is autonomous although supervised learning is actually utilized.

Many studies have investigated learning of prediction. The dynamics or context that appears in a learning system to predict the future states is utilized for state representation [1–6]. In some of these, a recurrent neural network is used as the learning system [1–4]. The learning of prediction has also been used to establish curiosity-driven learning [7–10]. However, typically, the prediction target is given, that is, what information should be predicted at what future timing; in many cases, the sensor signals at the next time step are the prediction target.

Regarding the abstraction process in robots, Brooks [11] pointed out the following: "This abstraction is the essence of intelligence and the hard part of the problems being solved. Under the current scheme the abstraction is done by the researchers leaving little for AI programs to do but search." When prediction is learned, the same holds true. A robot can get many sensor signals, such as visual sensor signals, but it seems difficult and meaningless to predict all the signals in some future. When considering the case of humans, we do not seem to predict all the visual signals at all the steps in the future. Therefore, the process of choosing a prediction target should be considered.

The information about the prediction target should be useful for achieving certain purpose and might be learned through experiences. For example, when humans chase and catch a batted ball, they seem to extract necessary information from numerous sensor signals and then predict the place where the ball will land as a prediction target and go ahead of the ball. We can easily know that "Prediction of the landing site" is useful "for catching the ball", but that must be a very intelligent process actually.

Schmidhuber has identified a very interesting and important point that we do not care either unpredictable or easily predictable information, but seem to "explore the predictable" [9]. He also proposed a learning system to realize this. However, to know whether a piece of information is predictable, the system must first conduct tests to predict the information. A method for discovering the prediction target from linear independence has also been proposed [12], but it only considers the sensor signals as inputs and does not provide a way to consider the purpose of the robot. Accordingly, it is difficult for this approach to realize a purposive prediction.

In reinforcement learning, an agent or robot learns from rewards and punishments based on trial and error. Therefore, it is a highly autonomous and purposive learning although the learning is generally very slow. If a neural network is used to connect the sensors to the motors in parallel and trained by training signals generated based on reinforcement learning, the network is optimized to obtain more rewards and less punishment, that is, to represent the value function more accurately and to generate actions with more gain for the value. Accordingly, with reinforcement learning it can be expected that the functions that contribute toward obtaining more rewards emerge in the neural network [13].

The objective of this paper is to clearly show that the prediction function, including the choice of prediction target, emerges purposively through reinforcement learning using a recurrent neural network when a given task requires prediction. In the learning system, the prediction of only predictable and useful information emerges without individual testing because only those predictions that contribute toward obtaining a reward emerge through reinforcement learning. Therefore, the system does not determine whether each piece of information is predictable or not individually. The curiosity-driven acceleration of learning is beyond the scope of this study and will be addressed in the future work.

To compensate for the missing information in solving a partially observable problem in reinforcement learning, a recurrent neural network or other finite-state controllers are often used [2, 3, 14–17]. Compensation of the missing information from the past series of sensor signals can be considered as a kind of "prediction" in a wide meaning. However, none of these studies has claimed that prediction should be considered in reinforcement learning. Furthermore, we usually call the function prediction when the environment changes dynamically, and the regularity in the dynamics is found. In the multiagent task in [15], the agent needs to predict the other agent's behavior to some extent to accomplish its purpose in a discrete state space. However, it is not shown how the agent predicts the other agent behavior and how the predicted information is represented inside the learning system after learning. Here, emergence of both spatial and temporal prediction in a continuous and dynamic environment through reinforcement learning is examined, and it is analyzed how the internal states represent the predicted information in the recurrent neural network.

2. Learning System

The learning system does not employ any special techniques for prediction, and a recurrent neural network is simply trained by the training signals that are derived autonomously on the basis of reinforcement learning. Therefore, it can be understood that reinforcement learning trains the recurrent network. Since the agent's actions are discrete in the task, while the state space is continuous, Q-learning [18] is used as a reinforcement learning algorithm.

For the recurrent neural network, a popular 3-layer Elman-type network is used, in which the outputs of hidden neurons, 40 in number, are fed back as inputs of the network at the next time step. The network is trained by back propagation through time (BPTT) technique [19]. The output function of each hidden or output neuron is the sigmoid function ranging from -0.5 to 0.5, and 0.4 is added before it can be used as a Q-value to match the value range between the output and Q-value.

The initial weights from input to hidden neurons are chosen randomly from -0.5 to 0.5, and those from hidden neurons to output are all 0.0. The initial feedback connection weights between hidden neurons are 0.0 except that the weights for the self-feedback connections are 4.0. The maximum derivative of the output function is 0.25. Therefore, under this setup, the error signal in BPTT propagates to the past without divergence, because the products of the self-feedback connection weight and the maximum derivative of output function become 1.0. Furthermore, in the forward computation, forming of bistable dynamics is promoted in each hidden neuron.

First, the present state is given as an input to the network. The recurrent network is expected to extract and store the necessary information in its hidden layer without holding the past state in the external memory. The output layer has the same number of neurons as that of the possible actions, and each output is used as the Q-value of the corresponding action.

For the action selection, a two-step stochastic selection is used. A small random number is added to each Q-value derived by the network computation, and then an action is chosen according to ε-greedy [4]. This not only assigns a higher priority to the actions with larger Q-values but also occasionally selects an action with a small Q-value. Both these random factors decrease with the number of episodes and finally become almost 0. We have not used a soft-max type action selection, such as Boltzmann selection, because the actions with very small Q-value are rarely chosen.

In the learning phase, a training signal is given only to the output of the chosen action a_t. The training signal $T_{a_t,t}$ for the action a_t in the present state S_t is generated autonomously using the maximum Q-value at the future state s_{t+1} after the action a_t:

$$T_{a_t,t} = r_{t+1} + \gamma \max_{a'} Q_{a'}(S_{t+1}) - 0.4, \tag{1}$$

where r indicates a reward and γ indicates a discount factor. 0.4 is subtracted to match the value range between the training signal and Q-value. The discount factor γ is set to 0.96 in this task. Further, if the training signal is greater than 0.4 or less than -0.4, the value is set to 0.4 or -0.4, respectively. Equation (1) indicates that at time $t+1$, the state of the network at time t is restored and the network is trained by the training signals generated by the equation, according to BPTT.

3. Task Setting

To examine whether the prediction function emerges, we used a task in which it is impossible to achieve a purpose without a prediction. This task is performed on a field of size 7.5×3.0, as shown in Figure 1. In the task, an agent catches an object. The initial direction of motion and velocity of the object are randomly chosen for each episode, and it cannot be seen moving in some area. The object bounces off walls. The agent gets a reward when the object approaches it and the agent catches the moving object at an appropriate position and timing. Therefore, considering the bounces, the agent has to predict the appropriate position and time for the approaching object before it becomes invisible.

The bottom left corner is defined as the origin, and the initial location of the moving object is fixed at (0.0, 1.5). For each episode, its initial velocity is chosen randomly between 0.50/step and 0.70/step and its direction is chosen between -45° and 45° from the x-axis. The object's direction of motion and velocity is constant during the episode unless it bounces off a wall. When it bounces, the angles of incidence and reflection are equal, and the velocity is reduced to 80% of the previous value. The agent is fixed on the line of $x = 6.0$ and can move only in the y direction. The initial y location is chosen randomly from 0.25 to 2.75 for every episode, enabling the agent to get a reward from any initial location, each time it chooses the optimal action. At every time step, the agent can choose one of the four actions: "catch," "wait," "move up," or "move down". When it chooses move up or move down, it moves 0.25 or -0.25 in the y direction. When the action chosen is wait, the agent does not catch or move.

When the agent chooses the action catch, the episode is complete, and the agent gets a reward when the relative distance between the agent and the moving object is less than 1.0. The reward value r is generated by

$$r = 0.40 \times (2.0 - d), \tag{2}$$

where r varies depending on the relative distance d. When d is 0.0, r becomes 0.8, which is the maximum value, and when d is 1.0, r becomes 0.4, which is the minimum value. When the agent chooses the catch action at $d > 1.0$, or when it does not choose the catch action until the object goes beyond the x limit of the field ($x = 7.5$), the episode is considered as failed and is forced to terminate. In such cases, to decrease the Q-value for the catch action, it is modified by the training signal as

$$T_{a_t,t} = r_{t+1} + Q_a(s_t) - 0.4, \tag{3}$$

and $r_{t+1} = -0.1$ is a punishment.

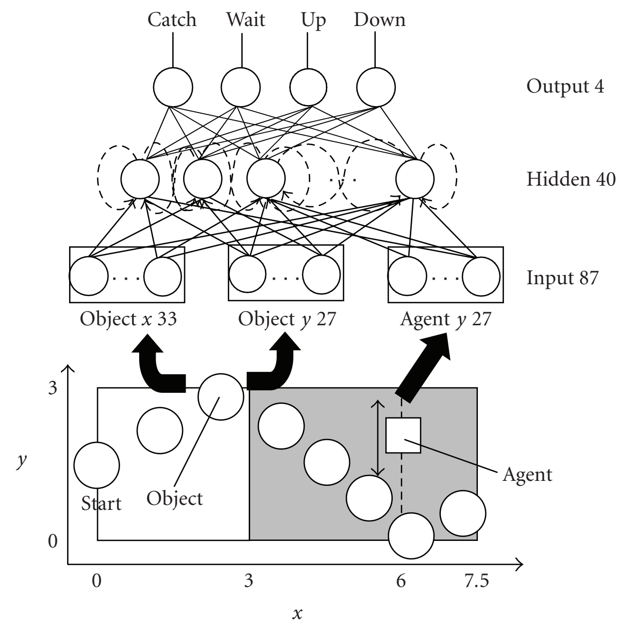

FIGURE 1: Object catching task and a recurrent neural network. An agent moves up or down and catches a moving object. The initial direction of motion and velocity of the object are chosen randomly for every episode. The invisibility area is also chosen randomly in the range of $x > 3.0$. x, y coordinates of the object and y coordinate of the agent are input to an Elman-type recurrent neural network. Each input signal represents local information, as shown in Figure 2.

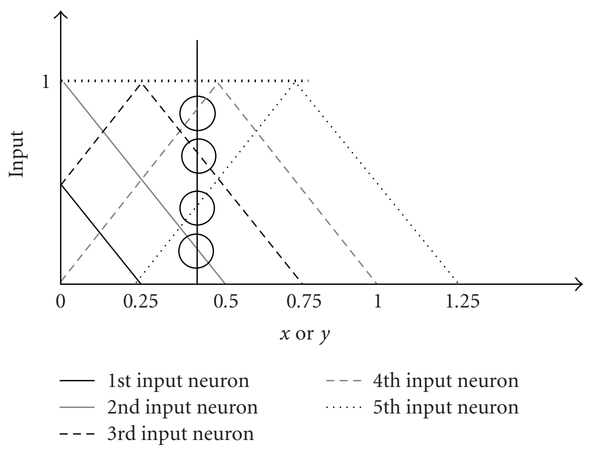

FIGURE 2: Localized input signals. Each input signal responds when the object or agent exists around one specific location in x or y coordinate. This localized representation helps to learn a strong nonlinear mapping.

Three signals, the x, y coordinates of the moving object and y coordinate of the agent, are provided to the agent as inputs. However, each signal is represented by signals, each of which actually responds to a local area of the original signal. With this representation, it becomes easy for the neural network to learn a strong nonlinear input-output relation. The x position of the moving object that ranges from 0.0 to 7.5 is represented by 33 signals. Each signal represents local information as shown in Figure 2. 4 of the 33 neurons have a value other than 0.0, and the others have the value 0.0. The y position of the object or agent is represented by 27 signals. There are a total of 87 signals that are inputs of the neural network.

An invisibility area lies over the region where the x coordinate is more than 3.0. When the object is in the invisibility area, the agent cannot see it and all the input signals representing the object location are all 0.0. Both the beginning and end of the invisibility area are set randomly within a range of 3.0–7.5 for every episode on the condition that the end position is larger than the beginning position. The agent is unaware of the beginning or end of the invisibility area in advance. Therefore, the agent cannot get a reward unless it predicts the position and timing of the moving object when it comes close to the agent. The prediction should be done using information acquired before the moving object enters into the invisibility area.

4. Experimental Results and Investigation

4.1. Learning Results. The learning curve is shown in Figure 3. The horizontal axis shows the number of episodes and the vertical axis shows the average reward.

This figure indicates that the agent can catch the object more accurately through iterative learning. The maximum reward generated by (2) is 0.80, but the object and agent locations are computed on a discrete time scale. Therefore, even though the agent always chooses the optimal action, the average relative distance is 0.144, as shown in Table 1. Table 1 also shows the performance after learning, compared with the case where the agent always chooses the optimal action.

Table 1: The agent's ideal and actual performance after learning for three cases of invisibility area.

	Range of the invisibility area			Ideal
	Random	Nothing	Maximum	
Average reward	0.685	0.685	0.681	0.742
Percentage with which the agent gets the reward	99.0	98.4	99.9	100
Relative distance between the agent and object when the agent chooses catch action	0.270	0.260	0.296	0.144

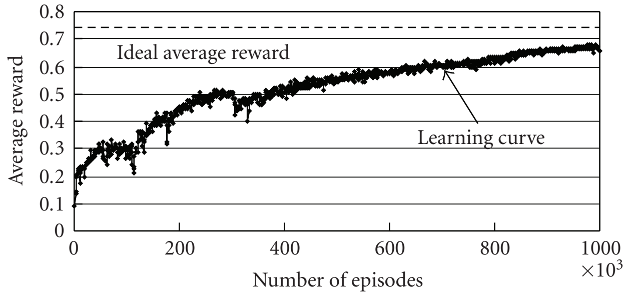

Figure 3: Learning curve. Change in the average reward according to the number of episodes is shown. Broken line shows the ideal average reward under the assumption that the agent always catches the object at the optimal timing and place.

Figure 4 shows an example of agent behavior after learning. In this case, the invisibility area is maximum, that is, from $x = 3.0$ to $x = 7.5$, the velocity is 0.5/step, and the angle of the object's direction of motion from the x axis is 35°. The agent can move only in the y direction along $x = 6.0$, but for easy understanding, Figure 4 is plotted assuming the agent moves together with the object in the x direction. In this case, the initial y location of the agent is 2.0.

According to Figure 5, the changes in Q-values for all actions, except the catch action, are similar to each other; however, for the catch action, the Q-value increases suddenly from the 13th step (around $x = 4.5$), and the agent finally chooses catch action at the 16th step. Therefore, the agent is considered to choose the catch action at an appropriate time step without catching the object at a wrong time before it comes into the reward area.

Figure 5 shows the change of the Q-value for each action in this episode.

The prediction of the catch timing for variable object velocities is observed. While maintaining the angle of the direction of motion constant, as shown in Figure 4, the velocity of the object is varied, and the catch timing is observed. Figure 6 shows the change in the Q-value of the catch action for three velocities. Although the agent cannot see the object in the latter half, the Q-value suddenly increases around the reward area in all the three cases.

Next, the authors examine whether the agent can predict the object location when it comes into the reward area. Figures 7 and 8 show the y coordinate where the agent catches the object for the two cases: with no invisibility area and with the maximum invisibility area. In these figures, the horizontal axis shows the initial angle of the object's motion from the x axis, and the vertical axis shows the y position where the agent catches the object. In these cases, the initial angle is determined from -45° to 45° at an interval of 1°, and the velocity is fixed at 0.6/step. To demonstrate whether the agent can catch the object at appropriate positions, the optimal position where the agent gets the maximum reward for each angle is plotted, denoted by squares.

Figures 7 and 8 show that both position and timing of the agent catching the object are appropriate except for the case around -45° initial angle and no invisibility area. Even in the case where the agent cannot see the object in the latter half of the episode, the agent catches the object at an appropriate position, considering that the object bounces off the wall. However, it seems difficult for the agent to catch the object

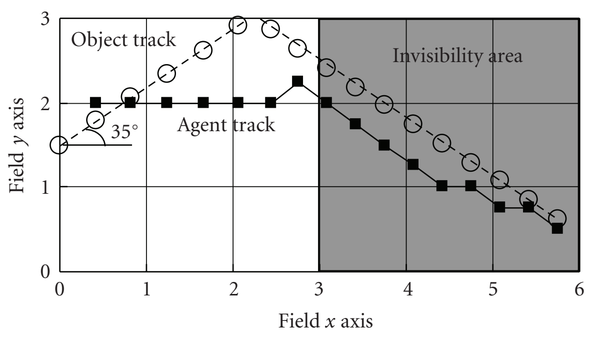

FIGURE 4: Sample object and agent trajectories for the object moving at 35° with a velocity of 0.50/step, and the object cannot be seen at $x > 3.0$. The agent does not move in the x direction actually, but for easy understanding, it is shown to be moving in the x direction along with the object.

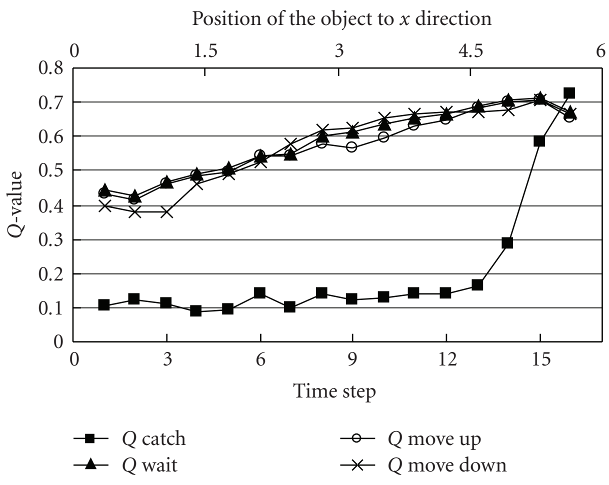

FIGURE 5: Change in Q-values in an episode. Four lines show the change in Q-values for the actions, "catch", "wait", "move up", and "move down". If the action is selected greedily, the action with the maximum Q-value is chosen.

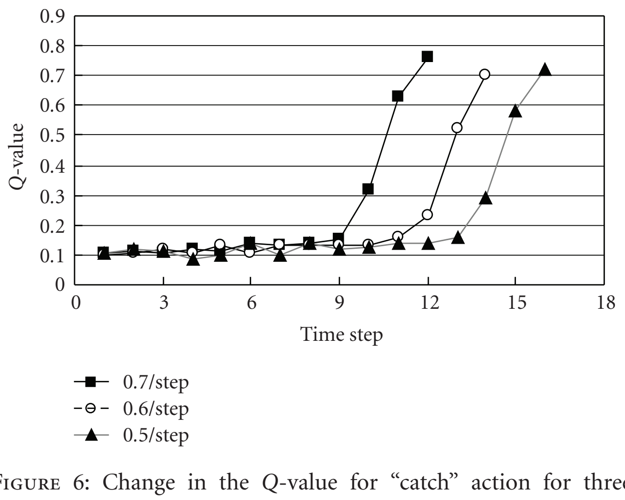

FIGURE 6: Change in the Q-value for "catch" action for three velocities. The timing for the increase in value differs, but the position of the object at that timing is almost the same in all cases.

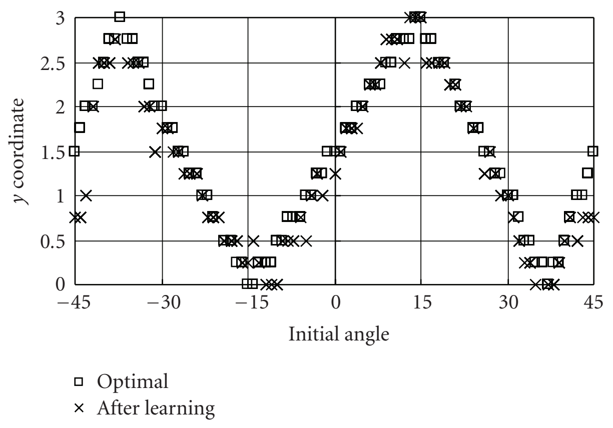

FIGURE 7: Positions where the agent catches the object when the initial object direction of motion varies (with no invisibility area).

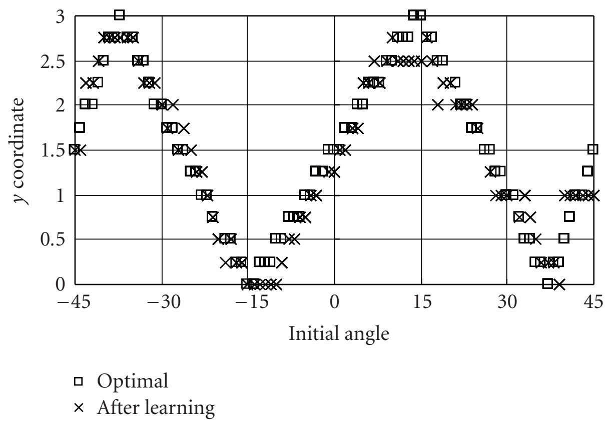

FIGURE 8: Positions where the agent catches the object when the initial object direction of motion varies (with the maximum invisibility area).

at around $y = 3.0$ in the case of the maximum invisibility area.

In this task, when the object bounces off a wall, its velocity is reduced to 80% of the previous value. Even if the object is in the invisibility area, it bounces and its velocity is reduced. If the agent cannot consider this property, it cannot catch the object at an appropriate timing. The authors examine whether the agent can really consider the reduction in the object velocity due to the bounce in the invisibility area, as follows.

If the velocity in the x direction is constant, the optimal time step for each catch action is equal unless the object bounces. The number of bounces increases with the increasing initial angle from the x axis, and this decreases the velocity. Then the initial angle is varied at an interval of 1°, with a constant velocity of 0.5/step in the x direction, and the time step when the agent catches the object is observed. In Figure 9, the optimal time step for catch actions is also plotted together with the observed results. In Figure 10, the optimal time step is plotted assuming that the object velocity is not reduced by the bounce in the invisibility area.

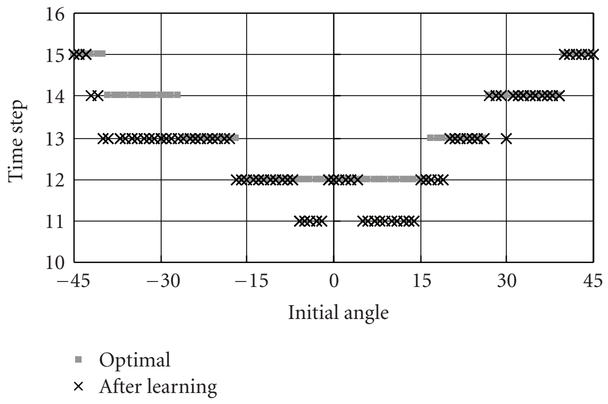

Figure 9: Catch timing when the initial motion of the object varies. The optimal catch timing is also plotted.

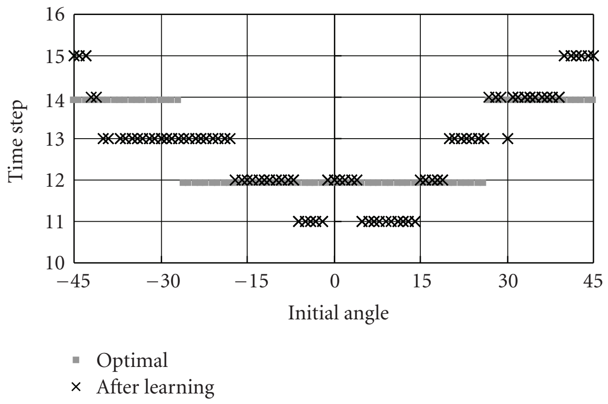

Figure 10: Catch timing when the initial motion of the object varies. The optimal catch timing under the assumption that the velocity is not reduced at the time of bounce in the invisibility area is also plotted to show whether the agent can consider the reduction in velocity due to the bounce by comparing it with Figure 9.

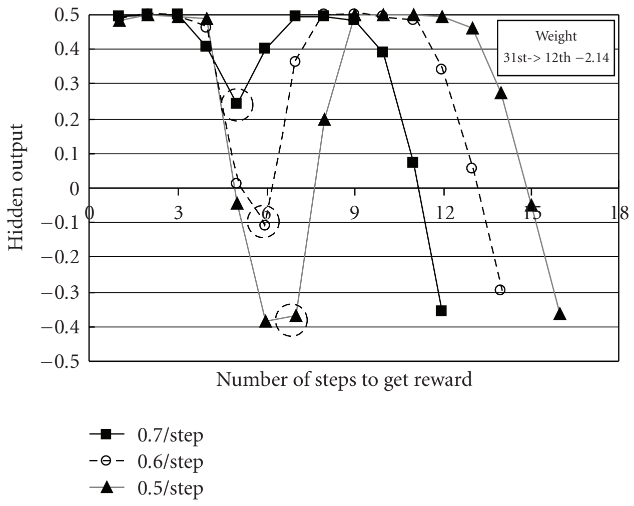

Figure 11: Change in output of the 31st hidden neuron for the three velocities. Broken circles indicate the timing just before the object disappears. Important connection weight value from this neuron is given in a box. In this case, the connection weight from the 31st neuron to the 12th neuron is −2.14.

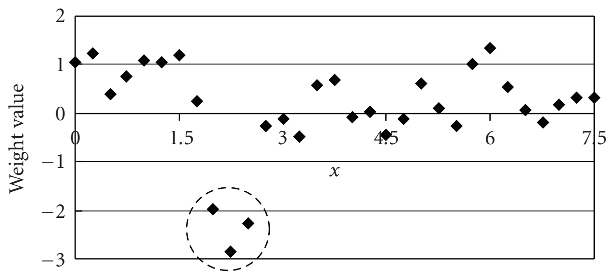

Figure 12: Connection weights to the 31st neuron from the input neurons that are responsible for the x location of the object. Lateral axis indicates the most responsible point of each input neuron. For example, the connection weight from the input neuron that takes the maximum value at $x = 2.25$ is approximately −2.8.

If the agent considers the reduction in velocity due to the bounce, the plots are more similar in Figure 9 than in Figure 10. In Figure 9, the time step is more similar to the optimal. However, the agent does not always catch the object at the optimal time step. The agent tends to catch the object a little earlier than the optimal.

These results indicate that through reinforcement learning alone, the agent can predict the necessary information such as the position or timing of the catch, considering the bounce, by using information before the object disappears.

4.2. Investigation of Hidden Neurons. In this section, the authors examine how the agent predicts the catch timing before the object enters the invisibility area, and the agent catches the object using the predicted information after some time lag. It appears that many neurons influence each other in a complex way, and prediction and catch are realized. In other words, one neuron apparently represents several pieces of information simultaneously and one piece of information is represented by several neurons. That is the same as our brain—a massively parallel system. Although it is difficult to clearly explain the mechanism, the authors try to elucidate it. In most graphs shown in the following figures, three lines represent changes in the output of the hidden neurons for each of the three cases shown in Figure 6. Some important connection weights from the neuron are indicated in the box marked “weight”.

As shown in Figure 11, the output of the 31st hidden neuron decreases at around $2.0 < x < 2.5$, that is, a little before the area where the agent may not see the object. The broken circle is set on the timing just before the object disappears at $x = 3.0$. Figure 12 shows the connection weight of the 31st neuron from the input neurons that are responsible for the x location of the object. It can be seen that the 31st neuron has a large negative connection weight from three input neurons, each of which responds when the object exists at around $x = 2.0$, 2.25, or 2.5. This suggests that the negative connection weights decrease the output of this neuron.

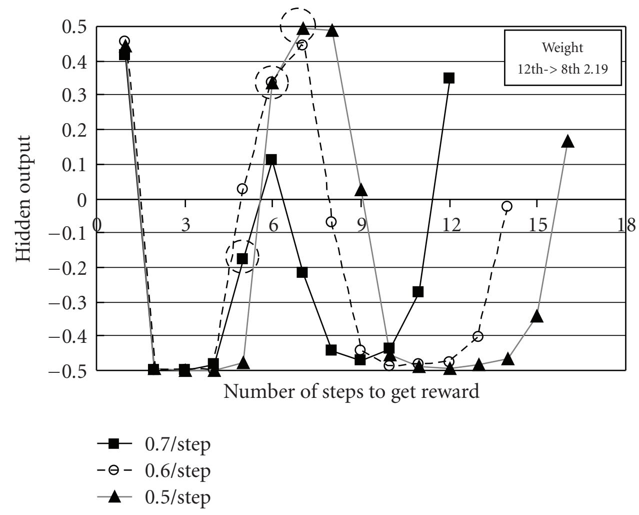

Figure 13: Output change of the 12th hidden neuron for the three velocities. Broken circles indicate the timing just before the object disappears.

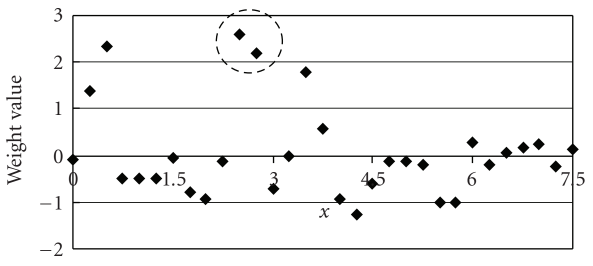

Figure 14: Connection weights to the 12th neuron from the input neurons responsible for the x location of the object. Lateral axis indicates the most responsible point of each input neuron.

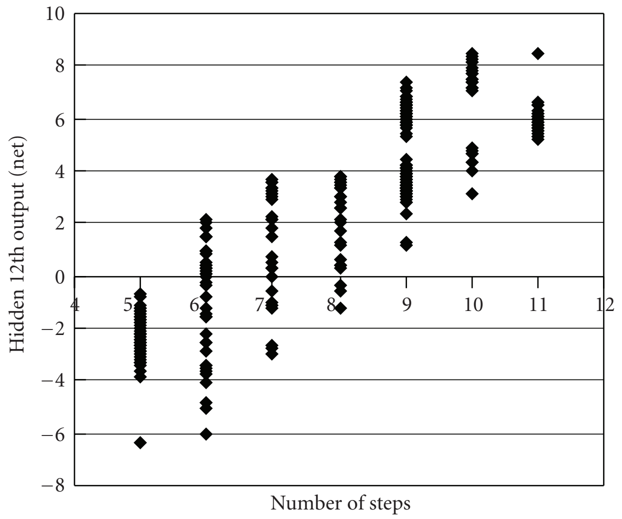

Figure 15: Correlation between the number of steps until catching the object in the invisibility area and the output of the 12th hidden neuron. The output varies according to the direction of motion and velocity of the object, although the number of steps until the catch is the same.

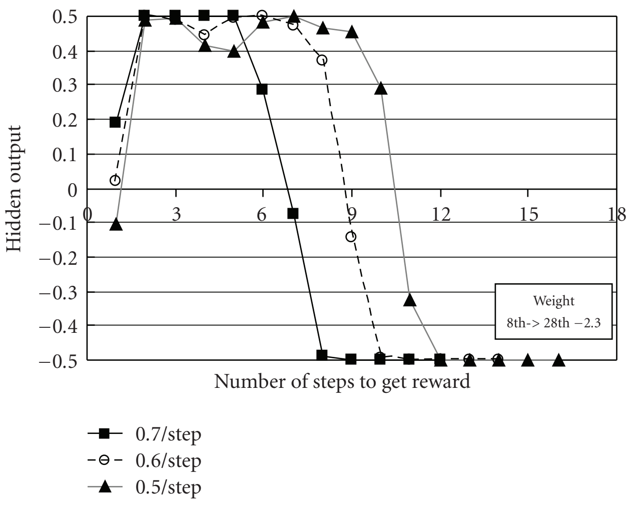

Figure 16: Output change of the 8th hidden neuron for the three velocities.

As shown in Figure 13, the output of the 12th hidden neuron increases just before the object comes into the area where it may be invisible. The timing when the output of 12th neuron increases is slightly later than that of the decrease in output of the 31st neuron. The 12th neuron is connected from the 31st neuron with a large negative weight. Figure 14 shows the connection weights of the 12th hidden neuron from the input neurons contributing to the representation of the x coordinate of the object. This neuron has a large positive connection weight from the input neurons, which respond just before the object enters the area where it is possibly invisible. These two types of connections, from the 31st neuron and from some inputs, might increase the output of the 12th neuron.

When the pattern of increase in the output of the 12th hidden neuron is observed, it is clear that the output increases as the velocity of the object in the x direction decreases. When the x velocity of the object is smaller, it stays in the specific area for a longer period before entering the area where it might be invisible. This suggests that both the 12th and 31st neurons play a role in detecting the x velocity of the object, and the 12th neuron represents the velocity of the object just before it enters the area where it might be invisible. The 12th neuron must represent the predicted catch timing and contribute toward catching the object at the appropriate timing even though the invisible area is wide.

Figure 15 shows the relation between the output of the 12th neuron just before the object passes the line of $x = 3.0$ and the number of steps from $x = 3.0$ to the catching of the object. Although they have some correlation, there is no one-to-one correspondence. As mentioned previously, the representation is distributed and one neuron represents several pieces of information simultaneously. In this case, this neuron represents not only the predicted catch timing but also some other information.

This 12th neuron has a large positive connection weight to the 8th neuron, whose output is shown in Figure 16.

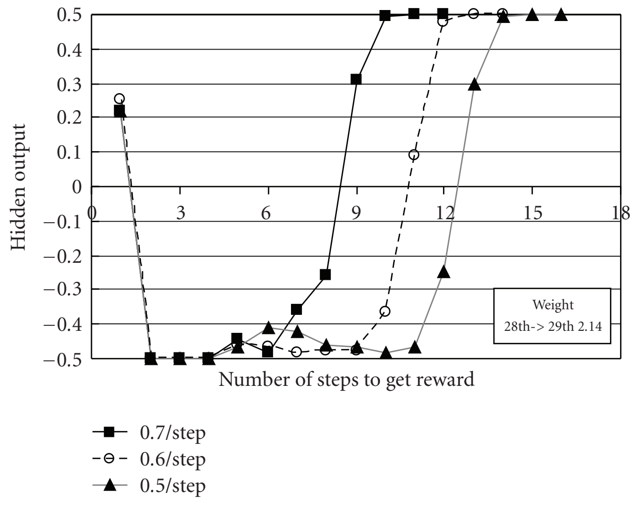

FIGURE 17: Output change of the 28th hidden neuron for the three velocities.

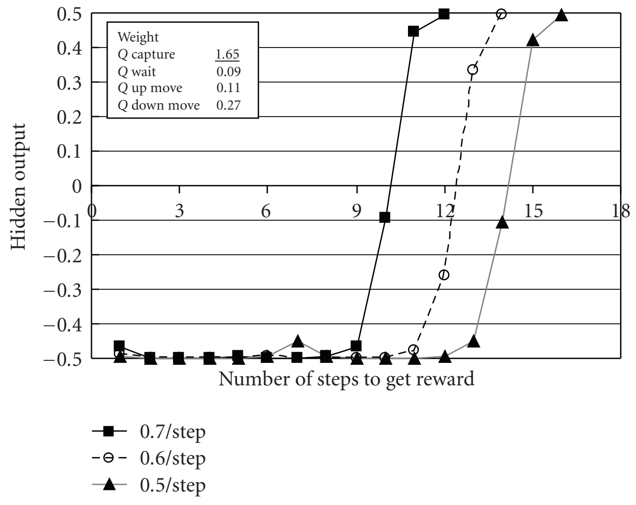

FIGURE 18: Output change of the 29th hidden neuron for the three velocities

Therefore, the delay of the timing of the decrease in the output of the 12th neuron causes a delay in the timing when the 8th neuron's output decreases. It is interesting that the difference of hidden output due to the object velocity in the 31st and 12th hidden neurons causes the delay of the signal in the 8th hidden neuron.

Next, the 28th neuron (Figure 17), which has a negative connection from the 8th neuron, responds later. Furthermore, the 28th neuron has a positive connection to the 29th neuron; the response of the 29th neuron (Figure 18) follows the response of the 28th neuron. The 29th neuron has a large positive connection to the output neuron, which represents the Q-value of the catch action whose response is shown in Figure 6. Thus, the response of the 12th neuron is relayed through some hidden neurons to the output neuron of the Q-value for the catch action. This relay of hidden neurons realizes the sudden increase in the Q-value for the catch action, as shown in Figure 6.

4.3. Consideration. The important point in this study is that the authors have not provided any knowledge in advance about the following items; the agent has learned them autonomously through reinforcement learning alone by reward and punishment.

(1) The velocity in the x direction of the object is useful for performing the catch action at the appropriate timing.

(2) The velocity in the x direction of the object can be detected by the inputs that respond to the existence of the object around a specific x coordinate.

(3) The way in which the detected velocity information can be related to the catch timing: especially, the detected velocity value is transformed to the delay of the signal.

(4) The information can be conveyed through a relay of hidden neurons.

Actually, more information than that discussed is considered in the recurrent network, but due to the parallelism of the processing system, it is difficult to understand its exact mechanism. Thus, it might also be difficult to understand the exact mechanism of the human brain.

5. Conclusion

In this paper, the authors proposed that the prediction function can emerge through reinforcement learning alone, using a recurrent neural network. This prediction function includes not only the way of predicting the target information but also the extraction of prediction target among many pieces of information available and the prediction of an appropriate timing. It was shown that through learning, the prediction function emerged—an agent could achieve a task in which prediction was necessary. Furthermore, the recurrent neural network extracts the necessary information for prediction from many input signals, relays the predicted information among some hidden neurons, and finally, enables the catching of the object at an appropriate timing. However, since the processing system is parallel, one neuron does not represent one piece of information explicitly, thus making it difficult to understand the processing of the network.

Acknowledgment

This research was supported by JSPS Grant-in-Aid for Scientific Research no. 19300070.

References

[1] J. Schmidhuber, "Temporal-difference-driven learning in recurrent networks," in *Parallel Processing in Neural Systems and Computers*, R. Eckmiller, G. Hartmann, and G. Hauske, Eds., pp. 209–212, North-Holland, Amsterdam, The Netherlands, 1990.

[2] J. Schmidhuber, "Reinforcement learning in Markovian and non-Markovian environments," in *Advances in Neural Information Processing Systems 3, (NIPS '91)*, D. S. Lippman, J. E. Moody, and D. S. Touretzky, Eds., pp. 500–506, Morgan Kaufmann, Denver, Colo, USA, 1991.

[3] L.-J. Lin and T. M. Mitchell, "Reinforcement learning with hidden states, from animals to animats 2," in *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*, pp. 271–280, MIT Press, Honolulu, Hawai, USA, 1993.

[4] J. Tani, "Learning to generate articulated behavior through the bottom-up and the top-down interaction processes," *Neural Networks*, vol. 16, no. 1, pp. 11–23, 2003.

[5] M. L. Littman, R. S. Sutton, and S. Singh, "Predictive representations of state," in *Advances in Neural Information Processing Systems*, vol. 14, pp. 1555–1561, MIT Press, 2002.

[6] R. S. Sutton and B. Tanner, "Temporal-difference networks," in *Advances in Neural Information Processing Systems*, vol. 17, pp. 1377–1384, MIT Press, 2005.

[7] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proceedings of the 1st International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, J. A. Meyer and S. W. Wilson, Eds., pp. 222–227, MIT Press/Bradford Books, Paris, France, 1993.

[8] M. B. Ring, "CHILD: a first step towards continual learning," *Machine Learning*, vol. 28, no. 1, pp. 77–105, 1997.

[9] J. Schmidhuber, "Exploring the predictable," in *Advances in Evolutionary Computing*, S. Ghoshr and S. Tsutsui, Eds., pp. 579–612, Springer, London, UK, 2002.

[10] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007.

[11] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, no. 1–3, pp. 139–159, 1991.

[12] P. McCracken, M. Bowling, and S. Tsutsui, "Online discovery and learning of predictive state representations," in *Advances in Neural Information Processing Systems 18*, pp. 875–882, 2006.

[13] K. Shibata and T. Kawano, "Acquisition of flexible image recognition by coupling of reinforcement learning and a neural network," *SICE Journal of Control, Measurement, and System Integration*, vol. 2, no. 2, pp. 122–129, 2009.

[14] A. Onat, H. Kita, and Y. Nishikawa, "Q-Learning with recurrent neural networks as a controller for the inverted Pendulum problem," in *Proceedings of the 5th International Conference on Neural Information Processing (ICONIP '98)*, pp. 837–840, Kitakyushu, Japan, October 1998.

[15] D. Aberdeen and J. Baxter, "Scaling internal-state policy-gradient methods for POMDPs," in *Proceedings of the International Conference on Machine Learning*, pp. 3–10, Las Vegas, Nev, USA, 2002.

[16] B. Bakker, V. Zhumatiy, G. Gruener, and J. Schmidhuber, "A robot that reinforcement-learns to identify and memorize important previous observations," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, pp. 430–435, Las Vegas, Nev, USA, 2003.

[17] H. Utsunomiya and K. Shibata, "Contextual behaviors and internal representations acquired by reinforcement learning with a recurrent neural network in a continuous state and action space task," in *Advances in Neural Information Processing Systems*, vol. 5507 of *Lecture Notes in Computer Science*, pp. 970–978, 2009.

[18] C. J. C. H. Watkins, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[19] D. Rumelhart, G. Hinton, and R.. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, D. Rumelhart, J. McClelland, and R. Williams, Eds., vol. 1, pp. 318–362, MIT Press, Cambridge, Mass, USA, 1986.