Spatial Abstraction and Knowledge Transfer in Reinforcement Learning Using a Multi-Layer Neural Network

Katsunari Shibata
Dept. of Electrical and Electronic Engineering, Oita University
700 Dannoharu, Oita 870-1192, JAPAN
shibata@cc.oita-u.ac.jp

***Abstract*— Abstraction is a very important function for living things. It generalizes the knowledge obtained through the past experiences and accelerates the learning drastically by applying the generalized knowledge to the present state. The most important subject, the author think, is the criterion for the acquisition of useful abstract information. Recently some techniques in which not only the reconstruction of input signals, but also the temporal relation of sensor signal patterns are considered in the abstraction process have been proposed. They seem rational because the information that affects the future seems important. The abstract information space obtained through such techniques is expected to be used as an effective state space in reinforcement learning.**

On the other hand, the author has advocated an idea that whole the necessary functions from sensors to motors, including recognition, memory, planning and so on, can be acquired autonomously by reinforcement learning using a neural network, and higher functions are interpreted as the inner part of the process. In this paper, it is proposed to use the reinforcement signal as a rational criterion for abstraction and the intermediate representation in a neural network trained by reinforcement learning is considered as abstract information. Spatial abstraction acquired through reinforcement learning using a neural network is demonstrated by showing the knowledge transfer in a simple task in which the combination of sensor and motor used in the task is chosen randomly at each episode. The representation of the upper hidden layer acquired through reinforcement learning is dependent on the object location but almost independent on the selected sensor. Such representation cannot be obtained by the above conventional techniques.

***Index Terms*— Spatial Abstraction, Knowledge Transfer, Neural Network, Reinforcement Learning, Generalization**

I. Introduction

If we, humans, had learned every task from scratch, we would not have been intelligent existences anymore. One important factor that makes us intelligent must be the ability of "ABSTRACTION" in the both aspects of space and time. Abstraction can be replaced by "feature selection" or "dimensionality reduction", and there are countless ways to realize the reduction of dimensionality. Reinforcement learning is an autonomous learning that enables an agent or robot to develop and learn by itself. However, slow learning due to the trial-and-error approach seems its fatal problem. Abstraction is also expected to accelerate reinforcement learning drastically.

The most important subject concerning about abstraction, the author think, is "what the criterion for the reduction should be" and "how the abstraction based on the criterion is realized". Some techniques have been proposed to discover a linear manifold in a high dimensional space such as principle component analysis. Recently some other techniques to discover globally non-linear manifold through local fitting have been proposed such as [1]. In both cases, the criterion is how well the sensor signals are reconstructed from the low-dimensional abstract signals. However, all the information that is necessary to reconstruct the input signals is not always important for the real or artificial creatures to generate appropriate actions.

In this aspect, "predictive representation"[2] and also "action representing embedding"[3] might be popular and also promising solutions for the "what the criterion should be" problem at present. They consider not only the reconstruction of sensory signal patterns, but also the temporal relation between sensor signal patterns. They seem rational because the information that affects the future seems important. "Predictive representation" also has a smart implementation method named TD-network[4]. Furthermore, it is easy to introduce the temporal abstraction and has also a good compatibility to reinforcement learning because TD learning is the learning that utilizes the temporal relation, and is widely used also in reinforcement learning.

However, the author is afraid that it has a serious problem in real world applications where a huge number of sensor signals should be processed. It must be impossible to predict all the huge number of sensor signals. If the predicted information must be decided in advance, it is abstraction itself to decide what information should be predicted. In order to solve this problem named "discovery problem", linear independence has been proposed as a criterion for the prediction[5]. However, the discussion seems to be back to the beginning.

J. Schmidhuber has provided a very interesting and important point that we don't seem to care either unpredictable or easily predictable information, and to "explore the predictable". He also proposed a novel and clever system to realize it[6]. Here, "predictable" can be the criterion for abstraction. It seems also rational because it is certain that

prediction of unpredictable is meaningless. However, in order to know whether a piece of information is predictable or not, the system has to try to predict the information at first.

On the other hand, in "action representing embedding"[3], the temporal relation between sensory signal patterns is reflected on the abstract space, and so it is expected that the relation of cause and effect is discovered. However, the same effect can be expected in the way of abstraction proposed in this paper, and an example that appears in the following sections shows that "action representing embedding" is not enough to know that two sensor signal patterns each of which was derived from the same state but using a different sensor came from the same state.

The author has been thinking that in real lives, a rational criterion might be reinforcement signal such as reward and punishment or genetic information, because they can be discussed in the context of the survival that is the most important thing in the beings. If the idea is accepted, it is a natural way that the abstraction is considered "in" reinforcement learning, while the abstraction process is positioned prior to and independent from the reinforcement learning in the conventional approaches.

In this paper, the author proposes to introduce reward and punishment as the criterion for the acquisition of useful abstract information in real or artificial creatures. Reinforcement learning is expected to optimize the abstraction autonomously through learning on the basis of the explicit criterion that the system gets more reward and less punishment. In reinforcement learning, state or action value is trained to change smoothly according time, and so the same effect as the "action representing embedding" described above can be expected. It is concerned that abstraction excludes some potentially useful sensory information, but it has been shown that learning in a layered neural network does not destroy the formerly obtained information so much if the number of neurons in the hidden layer is redundant enough[7].

II. Abstraction "in" Reinforcement Learning

Now the point is how the abstraction is realized as a part of reinforcement learning. Surprisingly the solution is very easy and requires no special technique, but needs just a switch of idea. The system consists of one neural network and it is trained by the supervised signal generated based on reinforcement learning. The hidden representations are interpreted as abstracted representation of the input sensor signals, which is also called observation vector. If a recurrent neural network is introduced, it is expected that the temporal abstraction can be dealt with. However, this paper focuses only on the spatial abstraction. The author's research group already proposed to use a neural network for reinforcement learning, and mentioned that the reinforcement learning can be an autonomous, purposive and harmonious learning for whole the functions we have from sensors to motors such as recognition, memory, conversation and so on[8][9][10].

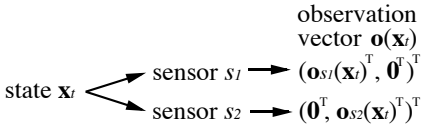


Fig. 1. The difference in the observation vectors for the same state depending on the used sensor.

used sensor	observation				
sensor s_1	$\mathbf{o}_{s1}(\mathbf{x}_1)$	1	0	0	0
	$\mathbf{o}_{s1}(\mathbf{x}_2)$	0	1	0	0
sensor s_2	$\mathbf{o}_{s2}(\mathbf{x}_1)$	0	0	1	0
	$\mathbf{o}_{s2}(\mathbf{x}_2)$	0	0	0	1

(a) observation vector only

used sensor	observation					V	a
sensor s_1	$\mathbf{o}_{s1}(\mathbf{x}_1)$	1	0	0	0	1	1
	$\mathbf{o}_{s1}(\mathbf{x}_2)$	0	1	0	0	2	2
sensor s_2	$\mathbf{o}_{s2}(\mathbf{x}_1)$	0	0	1	0	1	1
	$\mathbf{o}_{s2}(\mathbf{x}_2)$	0	0	0	1	2	2

(b) observation vector with state value V and action a

Fig. 2. An example to show that the state value and appropriate actions help to know that two different observation vectors come from the same state. In this case, the state is the same for the observation vectors $\mathbf{o}_{s1}(\mathbf{x}_1)$ and $\mathbf{o}_{s2}(\mathbf{x}_1)$ and also the same for $\mathbf{o}_{s1}(\mathbf{x}_2)$ and $\mathbf{o}_{s2}(\mathbf{x}_2)$, but there are no way to know that only from the observation vectors as in (a).

It may be difficult to explain the fact by this approach that we can predict some things that seem unrelated to any reward or punishment. However, on the contrary, can we assure that there exist any things that have completely no relation to any reward or punishment?

As mentioned at first, the important role of abstraction is acceleration of learning by knowledge transfer using abstracted knowledge. Here one example is given to demonstrate the advantage of this approach. There are two sensors each of which can observe the present state $\mathbf{x}_t$ independently. The system gets each observation vector $\mathbf{o}s_1$ or $\mathbf{o}s_2$ as a part of its whole observation vector $\mathbf{o}$ as shown in Fig. 1. If only one sensor is used in each episode, the target of the prediction is completely different according to the used sensor as shown in Fig. 2(a). This indicates that even though the state is the same, the system cannot know the fact if the used sensor is different. However, if the abstraction is done "in" reinforcement learning, it is possible for the system to know that the state is the same even though the used sensor is different. That is because the outputs that are trained to represent expected reward and appropriate actions are the same for the same state as shown in Fig. 2(b).

In the following, the advantage of the proposed approach is demonstrated by the simulation of this simple example.

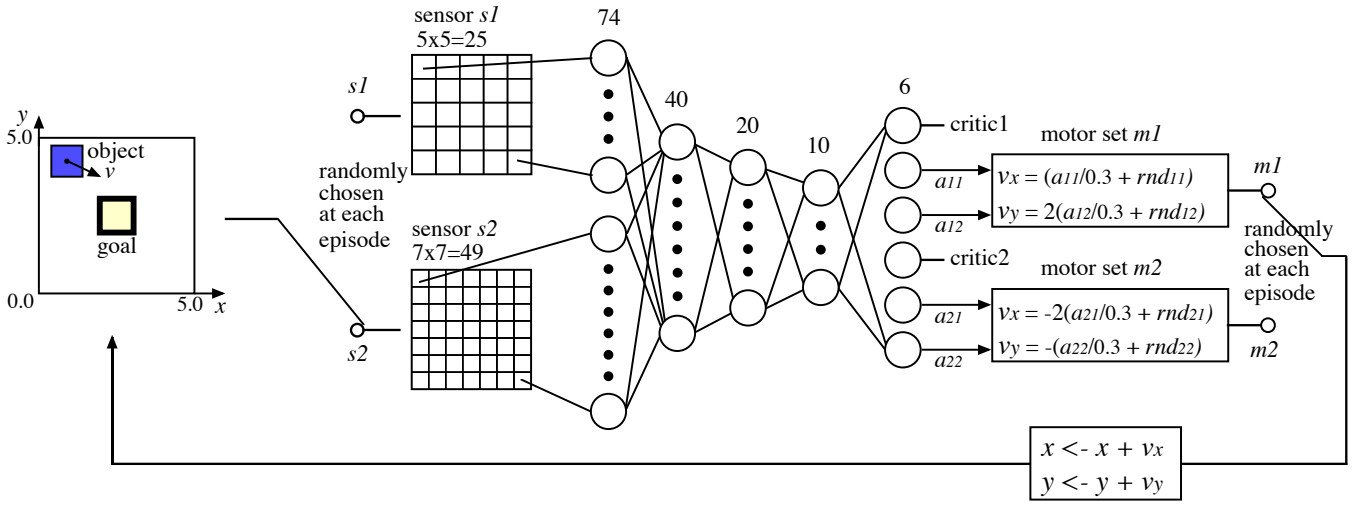


Fig. 3. The system architecture and signal flow. The combination of used sensor and motor set is chosen randomly at every episode.

III. Simulation Setup

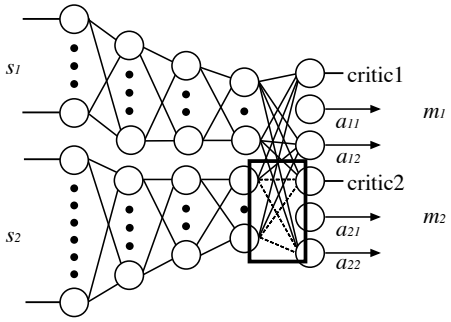
A. Task and Architecture

As mentioned above, there are no special techniques in the system employed here. The system architecture and signal flow are shown in Fig. 3. The system consists of only one regular multi-layer neural network. Each neuron computes weighted sum of the input signals, and then outputs the sigmoid function of the sum. The output value ranges from -0.5 to 0.5. There are no connections beyond one or more layers, and the initial connection weights are small random values. However, only the initial connection weights to the output layer are 0.0 that make the outputs be 0.0 not depending on the input signals before learning.

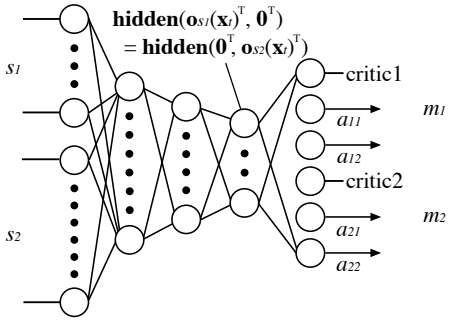
A square-shaped object with 1.0 side length is located randomly on the square-shaped area with 5.0 side length. There are two visual sensors, s_1 and s_2. The visual sensor s_1 consists of $5 \times 5 = 25$ visual cells, while the sensor s_2 consists of $7 \times 7 = 49$ visual cells. The visual field of the two sensors is the same with each other, and is identical to the square-shaped area. It can always catch the object. The size of the receptive field of each sensor cell is different between the two sensors, and the object size is identical to the size of one sensor cell of the sensor s_1. One of the sensors is chosen randomly at every episode. Each sensor cell outputs the area ratio occupied by the projected object comparing with the receptive field. For the sensor that is not chosen, all the sensor signals are 0.0. All the sensor signals from the both sensors are the input of the neural network. Accordingly the network receives $25 + 49 = 74$ input signals totally.

The system also has two sets of motors, m_1 and m_2. One set of the two is chosen randomly at every episode and is used to move an object during the episode. One set of motors consists of two motors, and each motor is responsible for the object motion in one of x and y directions. The outputs of the neural network are divided into two parts, each of which is corresponding to one set of motors. The popular actor-critic architecture[11] is introduced in each part that consists of three outputs. One of the outputs works as critic, and the other two works as actors each of which is corresponding to one of the motors respectively. The actual motor commands are the sum of the actor output vector $\mathbf{a}$ and a random number vector $\mathbf{rnd}$ as trial and error factors. The motor characters are different between the two sets, and they are given by the equations in Fig. 3. The velocity for each axis, v_x or v_y, is limited from -1.0 to 1.0. The object cannot be moved out of the square-shaped area so that the coordinates of the object center always satisfies $0.5 \leq x \leq 4.5$ and $0.5 \leq y \leq 4.5$. When the object center reaches the area that satisfies both $2.0 \leq x \leq 3.0$ and $2.0 \leq y \leq 3.0$, the system can get a reward of 0.7 and the episode finishes. When the object cannot reach the goal area within 100 steps, the episode finishes with no reward. The critic that is used to generate the training signal is also switched according to the selected motor set. Since no penalty is given in this task, the critic output is used after adding 0.4 to the actual output of the network so that the critic value ranges from -0.1 to 0.9. The discount factor is 0.96 here.

There are redundant hidden representations to realize the output. If some hidden neurons are responsible only to the observation vector from one sensor, and the other neurons are responsible only to that from the other sensor as shown in Fig. 4(a), the appropriate mapping for the combination s_2-m_2 indicated by the thick rectangle cannot be obtained by the learning of the other combinations. However, if the hidden representations for each state are the same not depending on the selected sensor as shown in Fig. 4(b), the mapping for s_2-m_2 may be obtained by the learning of the other



(a) completely separated representations



(b) common representation not depending on the selected sensor

Fig. 4. The two extreme but possible hidden representations of the object location for the two different used sensors.

combinations. Then, in this simulation, the neural network does not learn for the s_2-m_2 combination, but just does its forward computation. By observing the performance, it is examined whether the knowledge transfer is done or not. It is also examined whether the representation of the upper hidden layer acquired through reinforcement learning depends on the selected sensor or not.

B. Learning

The training signals for the neural network are computed as follows based on reinforcement learning[11]. The TD-error $\hat{r}_t$ is computed as

$$\hat{r}_{t-1} = r_t + \gamma P(\mathbf{o}_t) - P(\mathbf{o}_{t-1}) \tag{1}$$

where r_t is the reward, P is the critic output, $\mathbf{o}_t$ is the observation vector and γ is a discount factor. The training signal for the critic $P_{s,t-1}$ is computed as

$$P_{s,t-1} = P(\mathbf{o}_{t-1}) + \hat{r}_{t-1} = r_t + \gamma P(\mathbf{o}_t). \tag{2}$$

The training signals for the actor $\mathbf{a}_{s,t-1}$ are computed as

$$\mathbf{a}_{s,t-1} = \mathbf{a}(\mathbf{o}_{t-1}) + \hat{r}_{t-1}\mathbf{rnd}_{t-1} \tag{3}$$

where $\mathbf{rnd}$ is the random number vector that was added to the actor output vector $\mathbf{a}$. The neural network is trained by regular BP (Back Propagation)[12] using the above training signals. As mentioned, the critic value is used in the above equations after 0.4 is added to the output of the network, and the training signal for the critic is used after subtracting 0.4 from the above training signal.

IV. Simulation Result

Each figure in Fig. 5 shows the learning curve for each combination of sensor and motor set. y axis indicates the average steps to the goal. The neural network structure is different among (a) (b) and (c) in Fig. 5. The neural network has three hidden layers in the case of (a) and (c), while it has only one hidden layer in the case of (b). The number of hidden neurons is 40-20-10 in the case of (a), 40 in the case of (b), and 40-40-40 in the case of (c). In the cases from (a-2) to (a-4), one of the combinations of sensor and motor set is not presented during the learning.

In Fig. 5 (a-1), the performance for the combination of s_2-m_2 becomes better even though no learning is done for the combination as mentioned. On the other hand, in Fig. 5(a-2,3,4), the performance for the s_2-m_2 combination does not become better at all even though the performance of the other combinations becomes better. The result of the simulations in [7] and [13] that examined the hidden representation of the neural network through supervised learning suggested that even though two input signal patterns are not similar with each other, the hidden patterns corresponding to the input patterns become close through learning when the corresponding training signal patterns are similar. From this knowledge, the above result can be explained as follows. The difference of hidden representations due to the selected sensor becomes small through the learning for the combinations of s_1-m_1 and s_2-m_1 because the same output neurons are used and trained. Then a common representation of the object location not depending on the selected sensor is obtained. Accordingly, by adding the learning for s_1-m_2 combination, the mapping from the common representation to the output for the motor set m_2 is obtained, and the performance for s_2-m_2 combination becomes better without learning for the combination. In the case of (a-2) and (a-4), the common representation not depending on the selected sensor cannot be obtained. While, in the case of (a-3), although the common representation can be obtained, the mapping from the representation to the outputs for the motor set m_2 cannot be obtained because any learning was not done for the motor set m_2.

The comparison on the neural network structure suggests that the performance for the combination s_2-m_2 is better when the number of layers becomes more and the number of hidden neurons in the higher hidden layer becomes less to the extent that the network can learn. This is also compatible to the simulation results in [13]. One more thing that can be noticed easily is that in the case of the three-layer neural network, the learning is faster than the others. It is general that the learning is slower when the number of layers becomes more especially when the initial weight values are

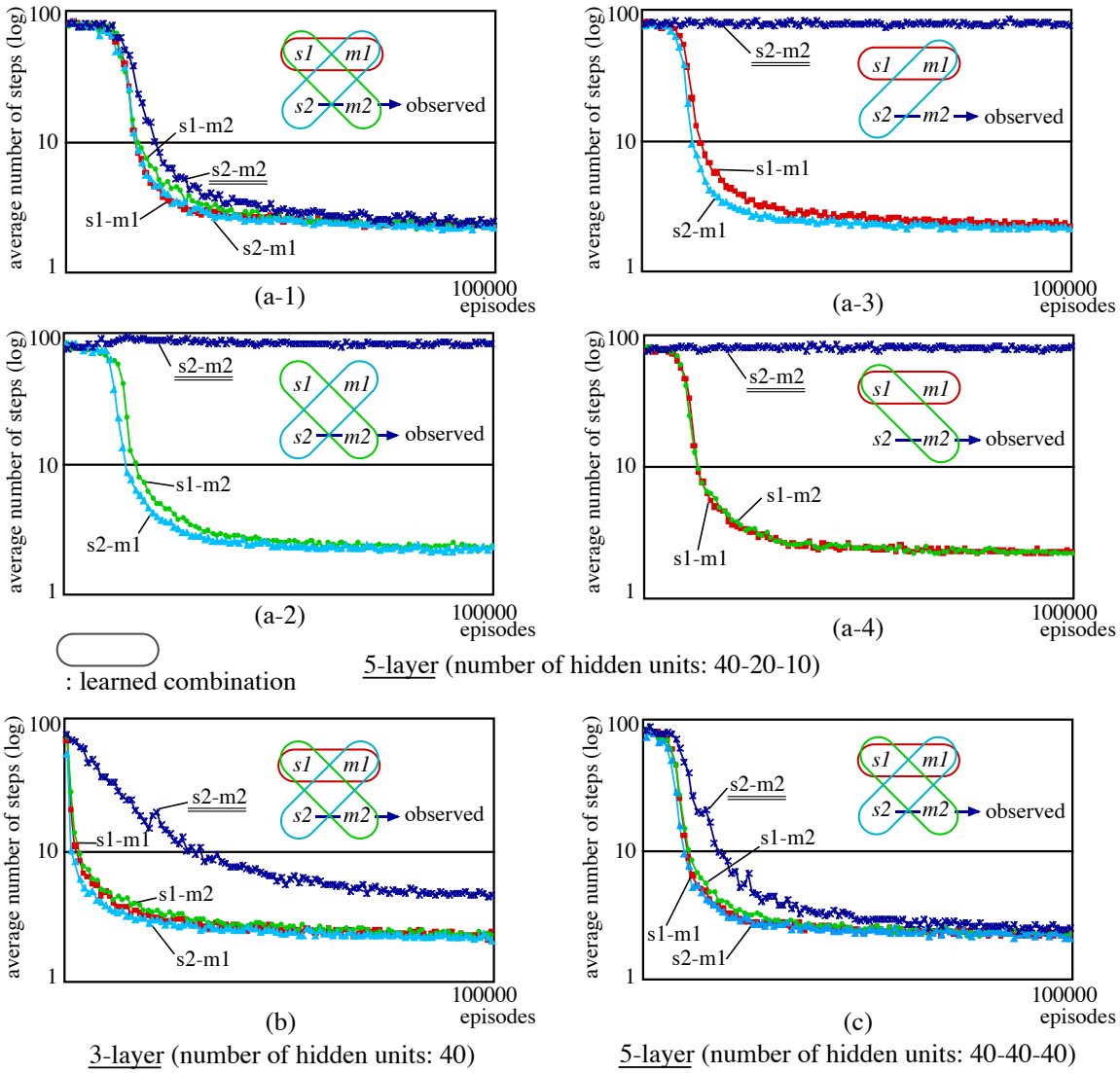


Fig. 5. Learning curve for each combination of sensor and motor set. No learning is done for the combination of the sensor s_2 and the motor set m_2. It can be seen that when the learning is done for the other three combinations, the performance becomes better for the combination s_2-m_2 as the progress of the learning for the other three combinations. It can also be seen that the network structure deeply influences to the performance.

small. The learning is a little bit faster for the case of 40-40-40 hidden neurons. It is also general that the learning becomes faster as the number of neurons becomes more.

In order to confirm the above explanation, the representations of the upper hidden layer are observed. Some examples are shown in Fig. 6 for the cases of (a-1) and (a-2). It can be seen that the hidden patterns are very similar between s_1 and s_2 in the case of (a-1), but in the case of (a-2), the patterns are not so similar. Fig. 7 shows the correlation of hidden patterns between the sensors s_1 and s_2. All the outputs of 10 neurons in the upper hidden layer for 81 object locations are plotted in each figure. The x-axis indicates the output of each neuron when the sensor s_1 is used, and the y-axis indicates the output when the sensor s_2 is used.

The correlation of the hidden patterns is very strong in the case of (a-1) and (a-3), while it is not so strong in the other cases. In the case of (a-2), negative correlation can be observed in some neurons such as the sixth neuron in Fig. 6 (a-2-1) and (a-2-2). In the case of (a-4), some neurons took almost the same value not depending on the object location or on the selected sensor. That can be seen in the three islands of dots at the upper right area in Fig. 7 (a-4). The reason might be that since the hidden layer has to represent the state only for the sensor s_1 in the case of (a-4), the number of hidden neurons is more redundant than the other cases.

V. Conclusion

It was proposed that reward and punishment are used as the criterion for abstraction. Based on this idea, it was also

proposed that abstraction is interpreted as an intermediate representation in the process from sensors to motors that consists of one neural network trained by reinforcement learning. It was shown that in the task in which one combination of sensor and motor set is chosen randomly at every episode, even though one of the four sensor-motor combinations was not learned, the performance for the combination became better through the learning of the other three combinations. However, one of the three combinations was missed, the performance did not become better any more. It was confirmed that the representation of the upper hidden layer was dependent almost only on the object location and independent on the used sensor.

ACKNOWLEDGMENT

A part of this research was supported by JSPS Grant-in-Aid for Scientific Research #1435027 and #15300064. A part of this research was done during my visit at University of Alberta by the fund of Overseas Advanced Educational Research Practice Support Program. I would like to express my gratitude to Prof. R.S. Sutton for fruitful discussions, and also to Mr. Shunta Sato who devoted himself to some preliminary simulations for this research.

REFERENCES

[1] L. Saul and S. Roweis, "Think globally, fit locally: Unsupervised learning of nonlinear manifolds," Journal of Machine Learning Research, Vol. 4, pp. 119–155, 2003.

[2] 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.

[3] M. Bowling, A. Ghodsi and D. Wilkinson, "Action Respecting Embedding," Proc. of the 21st Int'l Conf. on Machine Learning , pp. 65–72, 2005.

[4] R.S. Sutton and B. Tanner, "Temporal-Difference Networks," In Advances in Neural Information Processing Systems, Vol. 17, pp. 1377–1384, 2005

[5] P. McCracken and M. Bowling, "Online discovery and learning of predictive state representations," In Advances in Neural Information Processing Systems 18 , 2006 (to appear)

[6] J. Schmidhuber, "Exploring the Predictable," In Ghosh, S. Tsutsui, eds., Advances in Evolutionary Computing, pp. 579–612, Springer, 2002

[7] K. Shibata and K. Ito, "Adaptive Space Reconstruction and Generalization on Hidden Layer in Neural Networks with Local Inputs," Tech. Report of IEICE, NC2001-152, pp. 151–158, 2002 (in Japanese).

[8] K. Shibata and M. Iida, "Acquisition of Box Pushing by Direct-Vision-Based Reinforcement Learning," Proc. of SICE Annual Conf. 2003, 0324.pdf, pp. 1378–1383, 2003.

[9] K. Shibata and M. Sugisaka, "Dynamics of a Recurrent Neural Network Acquired through the Learning of a Context-based Attention Task," Artificial Life and Robotics, Vol. 7, pp. 145–150, 2004.

[10] K. Shibata and K. Ito, "Emergence of Communication for Negotiation By a Recurrent Neural Network," Proc. of ISADS (Int'l Sympo. on Autonomous Decentralized System) '99, pp.294–301, 1999.

[11] R.S. Sutton and A.G. Barto, "Reinforcement Learning," The MIT Press, Cambridge, MA, 1998

[12] D.E. Rumelhart, G.E. Hinton & R. J. Williams, "Learning Internal Representations by Error Propagation," Parallel Distributed Processing, The MIT Press, Vol. 1, pp. 318–362, 1987

[13] K. Shibata and K. Ito, "Reconstruction of Visual Sensory Space on the Hidden Layer in a Layered Neural Networks" Proc. of ICONIP (Int'l Conf. on Neural Information Processing) '98, Vol. 1, pp.405–408, 1998.

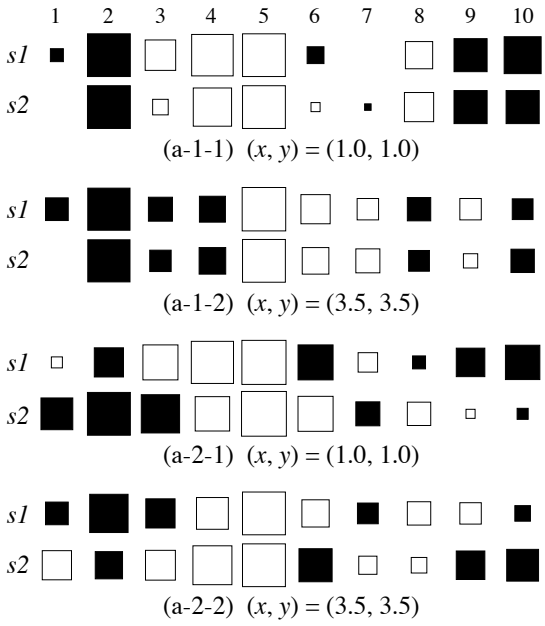


Fig. 6. Two examples of hidden representations for the case of (a-1) and (a-2). There are ten neurons in the upper hidden layer. The area of each square indicates the absolute value of the output, and the color indicates the sign of the output. White indicates that the output is positive, while black indicates that the output is negative.

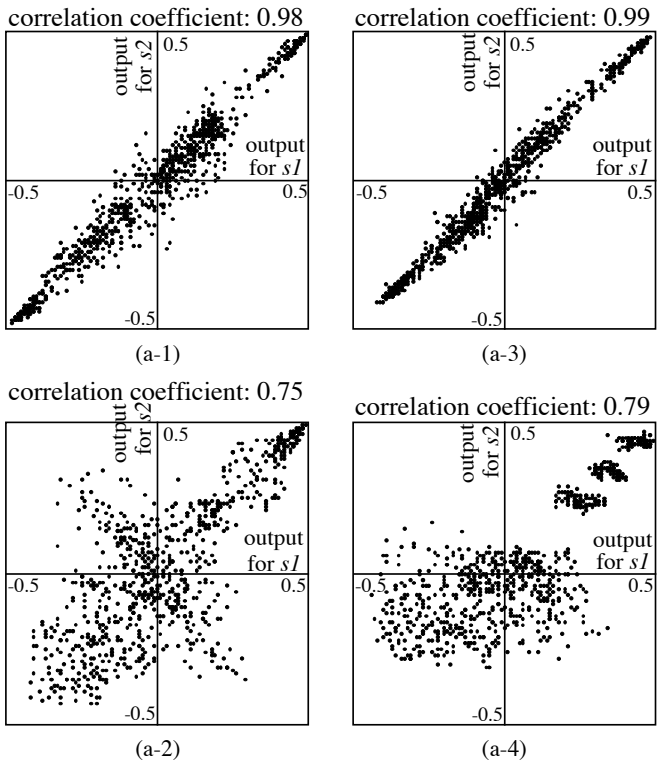


Fig. 7. The correlation of upper hidden patterns between the case using the sensor s_1 and the case using the sensor s_2. All the outputs of 10 hidden neurons in the upper hidden layer for 81 object locations are plotted in each figure.