# Brain-like Computing Based on Distributed Representations and Neurodynamics

Ken YAMANE and Masahiko MORITA

*University of Tsukuba*
*1-1-1 Tennoudai, Tsukuba, Ibaraki 305-8573 JAPAN*

yamane@bcl.esys.tsukuba.ac.jp, mor@bcl.esys.tsukuba.ac.jp

Received 29 May 2009
Revised manuscript received 30 October 2009

***Abstract*** A key to overcoming the limitations of classical artificial intelligence and to deal well with enormous amounts of information might be brain-like computing in which distributed representations of information are processed by dynamical systems without using symbols. We present a method for such computing. We constructed an inference system using a nonmonotone neural network, which is a kind of recurrent neural network with continuous-time dynamics. This system deduces a conclusion according to state transitions of the network in which knowledge is embedded as trajectory attractors. It has the powerful ability of analogical reasoning without special treatment for exceptional knowledge. We also propose a method of linking different neurodynamical systems and show that two mutually interacting systems can process complex spatiotemporal patterns.

**Keywords** Distributed Representations, Neurodynamics, Nonmonotone Neural Network, Selective Desensitization, Brain-like Information Processing.

## §1 Introduction

It has been noticed that classical artificial intelligence (AI) based on symbolic manipulation has the symbol grounding problem[1] and the frame problem[2, 3].

For those reasons, classical AI often does not work well in the real world, which is filled with unlimited amounts of information. In contrast, although animals such as dogs and cats do not seem to have languages or the ability to manipulate symbols, they evidently "think" or even "reason" and can deal better with many real-world problems than AI systems can. In their brains, it is thought that information is represented distributedly by activity patterns of neurons and processed without being symbolized to be other patterns, of which processes can be regarded as pattern dynamics[4, 5, 6].

If information is not represented by symbols as in the animal brain, then the symbol grounding problem does not arise. Distributed representations[7] facilitate analogy based on the similarity between representations, which seems to enable animals to address the frame problem. Accordingly, it is expected that such processing as is done in the animal brain will overcome some limitations of classical AI. We designate this approach as "brain-like computing", which aims to complement classical AI approaches rather than to surpass or expel them.

Although many other "brain-like computing" systems have been developed, few systems can manipulate patterns to reason without using symbols or equivalent representations. Regarding neural network models of inference, all existing models either require local representations that correspond one-to-one to things or concepts[8, 9, 10, 11, 12] or are combined with symbol processing models to form hybrid systems[13, 14, 15] because no scalable neural network has been developed that is based on distributed representations and can simulate an arbitrary finite automaton. For example, the Elman network[16], in reality, cannot learn well to simulate a given large-scale finite automaton.

We previously investigated this problem to find that it arises from an averaging effect of synaptic weights caused by one-to-many correspondence between input and output patterns. We also found that although conventional neural networks in general cannot escape from this averaging effect, it can be avoided by introducing a novel method of contextual modification termed selective desensitization, and developed a neural network model that can simulate any large-scale finite automaton without using symbols or local representations[17, 18].

In this paper, we present a brain-like computing system that can make inferences based solely on distributed representations using pattern dynamics in a kind of neural network. We examine its possibilities using simulation experiments. We also propose a novel method of linking different neural networks to enhance the information-processing capability of neurodynamical systems.

## §2 Basic Principles

### 2.1 Nonmonotone Neural Network

First, we explain the nonmonotone neural network[19)], which is a fully recurrent network of which elements have a nonmonotonic output function. Although it will be modified later for use as an inference engine, the original model is described by the following dynamics:

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} w_{ij} y_j + z_i, \tag{1}$$

$$y_i = f(u_i). \tag{2}$$

Here, $u_i$ is the internal potential of the $i$-th element, $y_i$ is the output, $w_{ij}$ is the connection weight from the $j$-th element, $z_i$ is the external input, $n$ is the number of elements, $\tau$ is a time constant, and $f(u)$ is the output function given as

$$f(u) = \frac{1 - e^{-cu}}{1 + e^{-cu}} \cdot \frac{1 - e^{c'(|u|-h)}}{1 + e^{c'(|u|-h)}}, \tag{3}$$

where $c$, $c'$, and $h$ are positive constants.

The polarity of $u_i$ is important in this model. Therefore, we consider $x_i \equiv \mathrm{sgn}(u_i)(\mathrm{sgn}(u) = 1$ for $u > 0$ and $-1$ for $u \leq 0)$ and refer to the vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ as the current state of the network. The network state $\boldsymbol{x}$ at an instant is represented as a point in state space consisting of $2^n$ possible states. It almost always moves to an adjacent point in the state space because $x_i$ changes asynchronously when $\boldsymbol{x}$ changes. Consequently, a continuous trace of $\boldsymbol{x}$ is drawn with the passage of time, which is called the trajectory of $\boldsymbol{x}$.

### 2.2 Trajectory Attractors

An important feature of the nonmonotone neural network is that it can make stable transitions along a given continuous trajectory in the state space; such a trajectory is called a trajectory attractor[19)]. By forming trajectory attractors from states $S^\mu$ $(\mu = 1, \ldots, m)$ to $T^\mu$, we can associate binary $(\pm 1)$ patterns $S^\mu$ with the corresponding patterns $T^\mu$: if $S^\mu$ is given as the initial state, then the network makes state transitions autonomously and recalls $T^\mu$. We respectively refer to $S^\mu$ and $T^\mu$ as cue and target patterns.

To form the trajectory attractor, we train the network using a spatiotemporal pattern $\boldsymbol{r}(t)$ changing continuously from $S^\mu$ to $T^\mu$ as a teacher

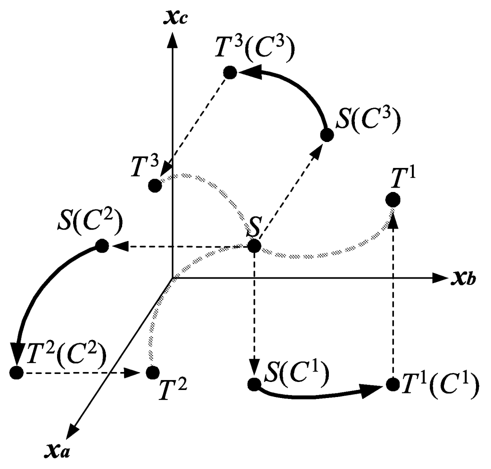

**Fig. 1** Schematic representation of the process of context-dependent recall.

signal. Specifically, we set $\boldsymbol{x} = S^{\mu}$ and feed $\boldsymbol{r}$ to each element in the form of $z_i = \lambda r_i(t)$ ($r_i$ is a component of $\boldsymbol{r}$ and $\lambda$ is the input intensity), and allow the network act according to Eqs. (1)–(3). Simultaneously, we modify connection weights $w_{ij}$ according to

$$\tau' \frac{dw_{ij}}{dt} = -w_{ij} + \alpha r_i y_j, \tag{4}$$

where $\tau'$ is the time constant of learning ($\tau' \gg \tau$) and $\alpha$ is the learning coefficient. Although $\alpha$ can be a constant, we set $\alpha = \alpha' x_i y_i$ in this study ($\alpha'$ is a positive constant) because the learning performance is improved if $\alpha$ decreases concomitantly with increasing $|u_i|$.

Intuitively, this learning decreases the energy of the network in the neighborhood of $\boldsymbol{r}$. Accordingly, when $\boldsymbol{r}$ changes successively from $S^{\mu}$ to $T^{\mu}$, a continuous groove remains in the energy landscape. In addition, because $\boldsymbol{r}$ moves slightly ahead of $\boldsymbol{x}$, a gentle flow from $S^{\mu}$ to $T^{\mu}$ is generated at the bottom of the groove. By repeating several cycles of learning for all $\mu$, gradually decreasing the input intensity $\lambda$ of $\boldsymbol{r}$, the network comes to make autonomous transitions from $S^{\mu}$ to $T^{\mu}$, or trajectory attractors are formed.

## 2.3 Context-dependent State Transition Using Selective Desensitization Method

The original nonmonotone neural network always recalls a fixed target from an identical cue. For the network to recall various targets according to the "context", we modify the network dynamics using the selective desensitization method [17, 18].

This method desensitizes about half of the elements or renders their output as neutral, depending on a given pattern $C$, which represents the context. Specifically, assuming that the neutral value or the average output is 0, we substitute Eq. (2) for

$$y_i = g_i f(u_i). \tag{5}$$

Here $g_i$ denotes a variable gain of the element, which usually takes 1 but takes 0 when the element is desensitized. We consider the simplest case, in which $C$ is an $n$-dimensional binary pattern whose components $c_i$ take $\pm 1$ with equal probability; the gain is given as $g_i = (1 + c_i)/2$.

Through this operation, the modified state of the network is projected onto a subspace comprising active (undesensitized) elements. It also produces transitions according to the dynamics in the subspace. If trajectory attractors are formed in the respective subspaces, then the network state reaches different target patterns according to context patterns, as presented schematically in Fig. 1.

## §3 Inference Based on Distributed Processing

Although our approach aims for non-linguistic thinking resembling that done by animals, describing how the model "thinks" and how it differs from classical AI would be rather difficult if we apply the model straightway to non-linguistic information processing. Instead, here we apply it to simple linguistic reasoning that could presumably be done by some smart animals if we were able to convey the meaning of words to them.

### 3.1 Method of Inference

Assume that a cue pattern $S^1$ is given to the model in a context $C^1$ and that the network makes state transitions from $S^1$ via $S^2$ to $T^1$. If the pattern $S^1$ represents *Sparrow*, $S^2$ *Bird* and $T^1$ *can fly*, then we can regard this transition as a reasoning process "*Sparrow* is a *Bird*, and therefore *can fly*" (Fig. 2). In this case, $C^1$ is regarded as representing the context in which <FLYING ABILITY> is asked. Under this interpretation, we ask a question "Can *Sparrow* fly?" by giving the cue pattern $S^1$ and the context pattern $C^1$ to the model, and the model makes an answer "*Sparrow* can fly." Here, we emphasize that $S^1$, $S^2$, $T^1$, and $C^1$ are not symbols but patterns between which the distance or similarity

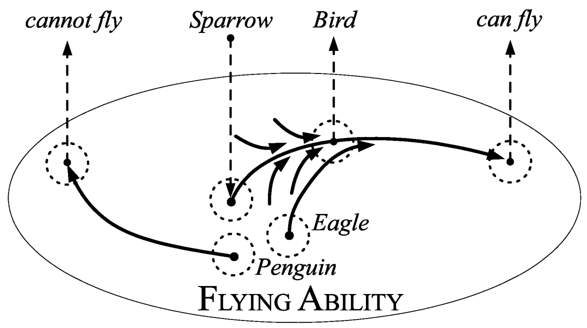

**Fig. 2** Method of inference.

can be defined naturally. Consequently, we can include the relation of similarity between things or contexts in the representations.

We must give knowledge to the model to construct an actual inference system. That knowledge is given by forming a trajectory attractor, for example, from $S^1$ (*Sparrow*) via $S^2$ (*Bird*) to $T^1$ (*can fly*) in the state subspace specified by context pattern $C^1$ (FLYING ABILITY), as presented in Fig. 2. If another trajectory attractor from $S^3$ (*Horse*) via $S^4$ (*Mammal*) to $T^2$ (*cannot fly*) is formed in the same subspace, then the system has gained another piece of knowledge "*Horse* is a *Mammal* and therefore *cannot fly*." Similarly, a trajectory attractor from $S^1$ (*Sparrow*) via $S^5$ (*Animal*) to $T^3$ (*move*) in another subspace specified as $C^2$ (MOBILITY) corresponds to a piece of knowledge "*Sparrow* is an *Animal*, and *moves*."

After acquiring knowledge, the system can infer a conclusion deductively by state transitions along a learned trajectory. Moreover, because of the distributed representation of knowledge and powerful generalization ability of the model, it is expected that the system can infer plausible conclusions even if novel questions are asked.

## 3.2 Encoding

The reasoning ability of the model described above depends largely on encoding: how to represent information using distributed patterns. In the case of the brain, not only concrete objects but also abstract concepts are represented as patterns of neuronal activity. These patterns are considered to be structured, i.e., related things are represented by similar patterns. For example, it is suggested that the categorical structure of visual objects is represented by the pattern of

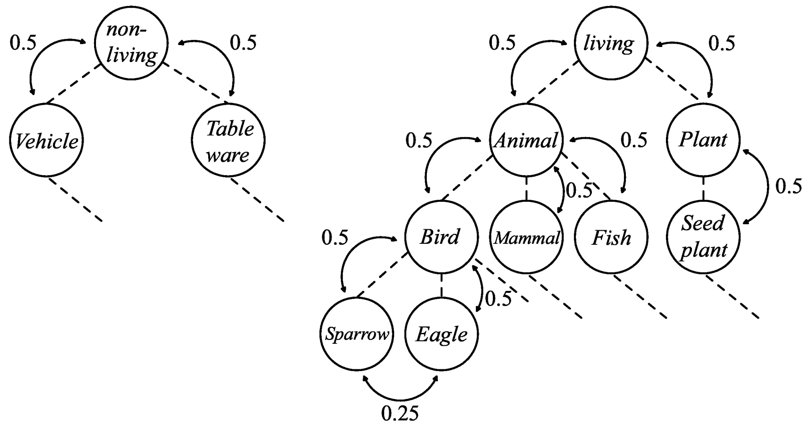

**Fig. 3** Hierarchical structure of similarities among code patterns.

activity distributed over neurons in the monkey's visual cortex and it is organized hierarchically [20]. However, detailed representations and encoding mechanisms in the brain are unclear, so that we cannot apply them directly to this model. In addition, developing a system in which suitable encoding is achieved by self-organization is an interesting but difficult subject. We therefore must reserve that issue for future study. Accordingly, we used a convenient method described below, which would not be the best.

First, we generated context and target patterns randomly under the condition that each component took the value $+1$ or $-1$ with equal probability. Accordingly, similarities (direction cosines) between these patterns were nearly zero, except that context patterns <FLYING ABILITY> and <WING> were set to have a similarity of 0.5, and target patterns <*can fly*> and <*have wings*> were set as identical so that we can investigate the case in which context patterns are similar.

Second, we constructed cue patterns based on categories. Specifically, the code pattern of an object was generated by adding a certain amount of noise to (flipping a certain number of components of) a pattern representing its category. Patterns representing categories were generated to form a tree structure, as shown in Fig. 3, where the numerical values denote similarities between patterns.

This structure was also used for training; that is, each piece of knowledge was represented as generally as possible using a superordinate concept because such generalized knowledge has widely various applications. It is noteworthy

**Table 1** Originally given knowledge.

| Context | Knowledge |
|---|---|
| FLYING ABILITY | *Sparrow⇒Bird⇒can fly*, *Bat⇒can fly*, *Horse⇒Mammal⇒cannot fly*, *Cherry⇒Plant⇒cannot fly*, *Airplane⇒can fly*, *Helicopter⇒can fly*, *Automobile⇒cannot fly*, *Bike⇒cannot fly*, *Boat⇒cannot fly*, *Cup⇒Tableware⇒cannot fly*. |
| BREEDING | *Swallow⇒Bird⇒egg*, *Dog⇒Mammal⇒young*, *Lily⇒Seed plant⇒seed*. |
| BREATHING | *Pigeon⇒Bird⇒lung*, *Bear⇒Mammal⇒lung*. |
| MOBILITY | *Eagle⇒Animal⇒move*, *Dandelion⇒Plant⇒not move*, *Bike⇒Vehicle⇒move*, *Fork⇒nonliving⇒not move*. |
| FEEDING | *Lion⇒Animal⇒feed*, *Azalea⇒Plant⇒not feed*, *Spoon⇒nonliving⇒not feed*. |
| CHLOROPLAST | *Giraffe⇒Animal⇒not have*, *Morning glory⇒Plant⇒have*, *Pot⇒nonliving⇒not have*. |

**Table 2** Added knowledge.

| Context | Knowledge |
|---|---|
| FLYING ABILITY | *Penguin⇒cannot fly*, *Tuna⇒Fish⇒cannot fly*. |
| WING | *Helicopter⇒not have*. |
| BREEDING | *Trout⇒Fish⇒egg*, *Grayfish⇒young*. |
| BREATHING | *Salmon⇒Fish⇒branchi*. |
| DEATH | *Duck⇒living⇒will die*, *Cow⇒living⇒will die*, *Apple⇒living⇒will die*, *Boat⇒nonliving⇒will not die*, *Knife⇒nonliving⇒will not die*. |

that some exceptions can exist (we can give "*Sparrow* is a *Bird* and therefore *can fly*" to the system although some birds cannot fly), and that the superordinate concept can differ in respective contexts (e.g., "*Sparrow* is an *Animal* and therefore *moves*").

## 3.3 Simulation Experiment

We simulated the model with $n = 2000$ elements and constructed the system on a computer to examine the reasoning ability of the model. The parameters were $c = 50, c' = 10, h = 0.5, \tau = 5000\tau'$, and $\alpha' = 2$.

For actual operation of the system, some points were noted. First, it is difficult for the system to learn exceptional knowledge because the corresponding trajectory attractors are affected strongly by those corresponding to common knowledge. They are therefore difficult to form. To cope with this problem, we increased training sessions for each piece of exceptional knowledge by five times so that all trajectory attractors were formed securely.

Second, conventional neural networks must generally relearn acquired knowledge when they learn new knowledge[21]. Such relearning is inefficient, however, if the system requires relearning even for a small addition of knowledge. We tested the system before and after additional learning of new knowledge to investigate this point.

The kind of knowledge that was given originally to the system and that which was added afterward are shown respectively in Tables 1 and 2. Initially, we trained the network 10 times on average for each piece of the original knowledge until the system acquired all. Then we asked the system various questions about both learned and unlearned knowledge. Next, we trained the network for the additional knowledge. The number of training iterations was 10 times on average. Then we asked various questions again.

Table 3 shows results in abbreviated form. In this table, shaded regions indicate the part related to additional learning (the question was newly asked or the answer changed); bold face indicates the answer that was given directly in training. We can say that the system gives a suitable answer to unlearned questions if we consider the limited knowledge given to the system.

## 3.4 Discussion

The inferential system described above has the following brain-like features that most existing systems do not have.

**(a) Analogy based on similarity between cue patterns.** A production system model[22], a typical inference engine of classical AI, basically cannot make an inference unless all the knowledge necessary for leading to the conclusion have been given as "if–then" rules. By contrast, the present system can reason analogically using other knowledge. For example, the system answers "*Eagle* can fly" to a novel question "Can *Eagle* fly?" because *Eagle* is represented using a code pattern similar to <*Sparrow*>. Therefore, the network state is attracted to the trajectory attractor <*Sparrow* $\rightarrow$ *Bird* $\rightarrow$ *can fly*>.

**(b) Analogy based on similarity between context patterns.** The system can perform analogical reasoning when the context is different but similar to a familiar context. For example, if the system knows that $X$ can fly, then it is generally inferred that $X$ has wings, even though it learned nothing about wings. This is because contexts <Flying Ability> and <Wing> are represented by similar patterns and trajectory attractors formed in the former context produce flows parallel to them in the subspace corresponding to the latter context.

**Table 3** Reasoning results.

| | FLYING | WING | BREEDING | BREATHING | MOBILITY | FEEDING | CHLOROPLAST | DEATH |
|---|---|---|---|---|---|---|---|---|
| *Sparrow* | ***fly*** | *have* | *egg* | *lung* | *move* | *feed* | *not* | *die* |
| *Eagle* | *fly* | *have* | *egg* | *lung* | ***move*** | *feed* | *not* | *die* |
| *Swallow* | *fly* | *have* | ***egg*** | *lung* | *move* | *feed* | *not* | *die* |
| *Duck* | *fly* | *have* | *egg* | *lung* | *move* | *feed* | *not* | ***die*** |
| *Penguin* | *fly* | *have* | *egg* | *lung* | *move* | *feed* | *not* | |
| | ***not*** | *not* | | | | | | *die* |
| *Pigeon* | *fly* | *have* | *egg* | ***lung*** | *move* | *feed* | *not* | *die* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *Bat* | ***fly*** | *have* | *young* | *lung* | *move* | *feed* | *not* | *die* |
| *Horse* | ***not*** | *not* | *young* | *lung* | *move* | *feed* | *not* | *die* |
| *Dog* | *not* | *not* | ***young*** | *lung* | *move* | *feed* | *not* | *die* |
| *Cow* | *not* | *not* | *young* | *lung* | *move* | *feed* | *not* | ***die*** |
| *Bear* | *not* | *not* | *young* | ***lung*** | *move* | *feed* | *not* | *die* |
| *Lion* | *not* | *not* | *young* | *lung* | *move* | ***feed*** | *not* | *die* |
| *Giraffe* | *not* | *not* | *young* | *lung* | *move* | *feed* | ***not*** | *die* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *Tuna* | ***not*** | *not* | *egg* | *branchi* | *move* | *feed* | *not* | *die* |
| *Trout* | *not* | *not* | ***egg*** | *branchi* | *move* | *feed* | *not* | *die* |
| *Grayfish* | *not* | *not* | ***young*** | *branchi* | *move* | *feed* | *not* | *die* |
| *Salmon* | *not* | *not* | *egg* | ***branchi*** | *move* | *feed* | *not* | *die* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *Cherry* | ***not*** | *not* | *seed* | | *not* | *not* | *have* | *die* |
| *Dandelion* | *not* | *not* | *seed* | | ***not*** | *not* | *have* | *die* |
| *Lily* | *not* | *not* | ***seed*** | | *not* | *not* | *have* | *die* |
| *Apple* | *not* | *not* | *seed* | | *not* | *not* | *have* | ***die*** |
| *Azalea* | *not* | *not* | *seed* | | *not* | ***not*** | *have* | *die* |
| *Morning glory* | *not* | *not* | *seed* | | *not* | *not* | ***have*** | *die* |
| ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ |
| *Airplane* | ***fly*** | *have* | | | *move* | *not* | *not* | *not* |
| *Helicopter* | ***fly*** | *have* | | | *move* | *not* | *not* | |
| | | ***not*** | | | | | | *not* |
| *Car* | ***not*** | *not* | | | *move* | *not* | *not* | *not* |
| *Bike* | ***not*** | *not* | | | ***move*** | *not* | *not* | *not* |
| *Boat* | ***not*** | *not* | | | *move* | *not* | *not* | ***not*** |
| ⋮ | ⋮ | ⋮ | | | ⋮ | ⋮ | ⋮ | ⋮ |
| *Cup* | ***not*** | *not* | | | *not* | *not* | *not* | *not* |
| *Fork* | *not* | *not* | | | ***not*** | *not* | *not* | *not* |
| *Knife* | *not* | *not* | | | *not* | *not* | *not* | ***not*** |
| *Spoon* | *not* | *not* | | | *not* | ***not*** | *not* | *not* |
| *Pot* | *not* | *not* | | | *not* | *not* | ***not*** | *not* |
| ⋮ | ⋮ | ⋮ | | | ⋮ | ⋮ | ⋮ | ⋮ |

This feature is unique compared with other neural network models such as mixture of local experts models, in which each local network or "expert" is specialized to a particular context and generalization across different contexts does not occur.

**(c) Nonmonotonic reasoning.** In general, inference systems with high ability of analogical reasoning suffer from exceptional knowledge. In stark contrast, systems that deal excellently with exceptional knowledge require numerous detailed rules and have difficulty using analogy. Although various methods have been proposed[23, 24, 25, 26] to cope with this dilemma, it seems impossible to solve it using classical AI approaches only because this problem is deeply related to the frame problem. However, the present system is not troubled by exceptional knowledge that causes a logical inconsistency because logical inferences are not used in this system.

For example, although *Bat* is a *Mammal* and is therefore represented by a similar code pattern to <*Horse*>, <*Dog*>, etc., the system replies "*Bat* can fly" to the question "Can *Bat* fly?" because exceptional knowledge has been given. Nevertheless, it performs analogical reasoning to other questions about *Bat* similarly as it does to questions about *Horse* and *Dog* (e.g. "*Bat* bears young"). In other words, the trajectory attractor <*Bat* $\rightarrow$ *can fly*> in the context <FLYING ABILITY> does not much affect the flow from <*Bat*> toward <*Mammal*> in other contexts because the influence of trajectory attractors toward <*Mammal*> in various contexts, such as <*Horse* $\rightarrow$ *Mammal*> in <FLYING ABILITY> and <*Dog* $\rightarrow$ *Mammal*> in <BREED>, is stronger.

For the same reason, the influence of exceptional knowledge in the same context is limited and does not impair the ability of analogical reasoning as long as the knowledge is exceptional. For example, when a novel cue pattern with equal similarities to <*Bat*> and to <*Dog*> is given in the context <FLYING ABILITY>, the network state moves to <*cannot fly*> through the neighborhood of <*Mammal*>. Such an inference can be regarded as common-sense reasoning of a kind.

**(d) Analogical reasoning using a structure of similarities among code patterns.** A crucial advantage of distributed representations over symbolic representations is that the relation among objects, like that shown in Fig. 3, can be expressed implicitly by similarities or distances between code patterns. The present system can make good use of this advantage for inference.

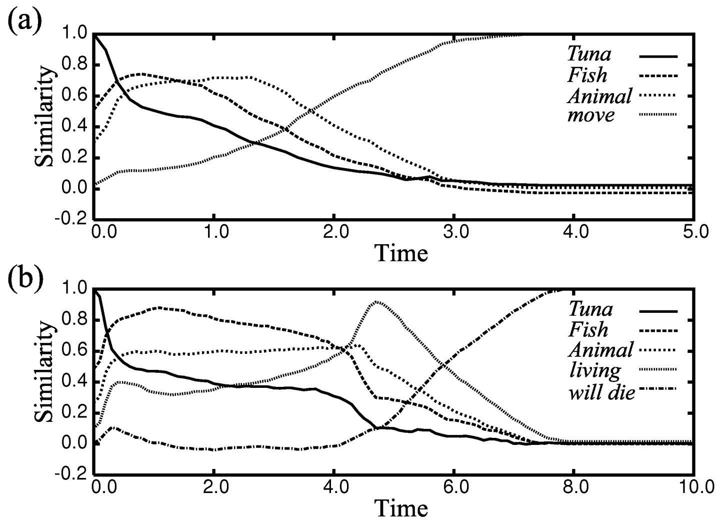

**Fig. 4** Process of reasoning using a structure of code patterns.

For example, although the system has learned nothing about *Fish* in the context <MOBILITY> and has not learned explicitly that they belong to *Animal* in any context, it drew the conclusion that "*Tuna* moves." The process of inference is shown in Fig. 4(a), in which the time course of similarities between the network state $\boldsymbol{x}$ and individual code patterns is shown (the abscissa is scaled by the time constant $\tau$). As this graph shows, the network state moves initially from <*Tuna*> toward <*Fish*> and <*Animal*> as a result of the influence of various trajectory attractors to them. Then it is attracted to the trajectory attractor <*Animal* $\rightarrow$ *move*>, which was formed when the system learned "*Eagle* is an *Animal*; consequently, *moves*."

Similarly, the system infers "*Tuna* will die", as depicted in Fig. 4(b), where the network state is carried by a flow along <*Tuna* $\rightarrow$ *Fish* $\rightarrow$ *Animal*>. It is then attracted to a trajectory attractor formed when the system learned "*Duck*, *Cow* and *Apple* are *living*. Consequently, *will die*."

**(e) Addition of knowledge.** Conventional multilayer neural networks confront the serious problem that previously acquired knowledge is disrupted suddenly in the process of learning a new set of knowledge, which is called catastrophic interference[21]. The present system is, however, robust to additional learning of new knowledge.

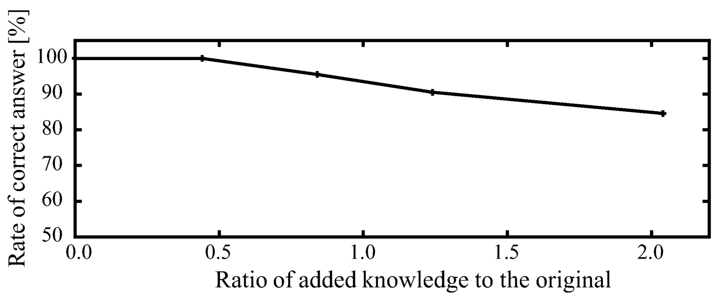

**Fig. 5** Interference to learned knowledge by additional learning.

In the experiment described above, for example, addition of knowledge about *Fish* caused no error in inference as to acquired knowledge because <*Fish*> is distant from other code patterns. Similarly, additional learning in a novel context <DEATH> has little influence on acquired knowledge.

However, if completely conflicting knowledge such as "*Horse* can fly" is added, some previous knowledge ("*Horse* cannot fly") is lost. Furthermore, the addition of exceptional knowledge can interfere with related general knowledge. In particular, addition of much exceptional knowledge of the same kind can change some general knowledge. Some examples are that if very many flightless birds are newly learned, the system will infer "it cannot fly" for an unknown bird. Nevertheless, the interference is actually limited in most cases. For example, even after the system additionally learned "*Helicopter* has no wings," it can infer by analogy that "*Eagle* has wings" and "*Car* has no wings."

To examine the interference effects more specifically, we gradually increased the amount of additional knowledge in the experiment described above while maintaining the ratio of exceptional knowledge around 20%. Figure 5 shows the result in which the percentage of correct inferences as to original knowledge is shown against the ratio of added knowledge to the original. That figure shows that the rate does not decrease rapidly; it remains higher than 80%, even when a double amount of knowledge is added. This result indicates that the system can accumulate knowledge merely through occasional relearning.

## §4 Interaction Between Neurodynamical Systems

The objects and concepts involved in the inference system presented above are represented by static patterns and can be symbolized, but we believe

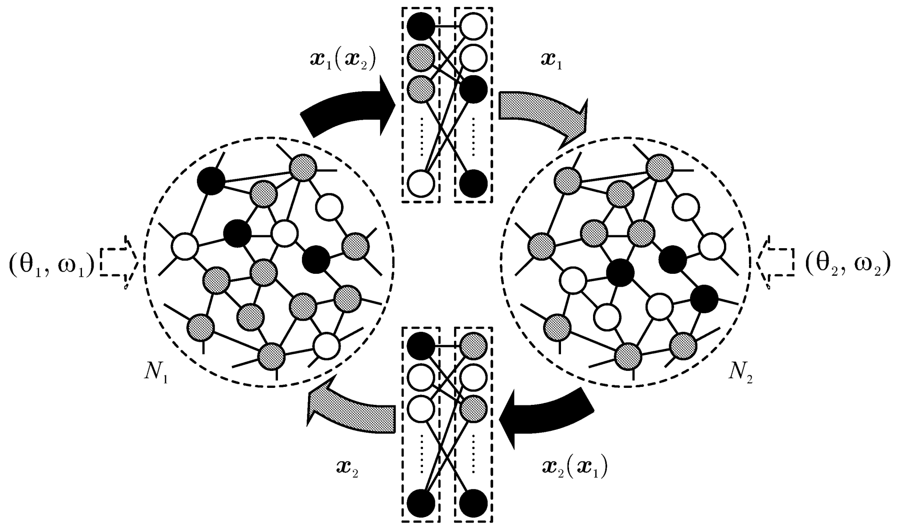

**Fig. 6** Structure of a model of interaction using dynamic modification.

that a critical advantage of brain-like computing will be revealed when we deal with objects that are difficult to represent as symbols such as motions of the body.

However, the system described above is insufficient for dealing with such objects because they should often be represented by spatiotemporal patterns and because a single neurodynamical system cannot process complex spatiotemporal patterns. Considering that the actual brain comprises many areas with specific functions working together, we describe in this section a method of linking multiple neurodynamical systems to construct a modular system.

## 4.1 Dynamic Modification

The basic idea of our method is to modify a neurodynamical system using the state of another system as the context. However, we cannot apply the previous selective desensitization directly because the context pattern has been assumed static so that each trajectory attractor can be formed in a certain subspace.

Nevertheless, if the pattern of desensitized elements varies rather slowly, then it is thought that short trajectory attractors formed in a series of subspaces substantially comprise a long trajectory attractor. Accordingly, to induce interactions between neurodynamical systems, we need only transform the state of a

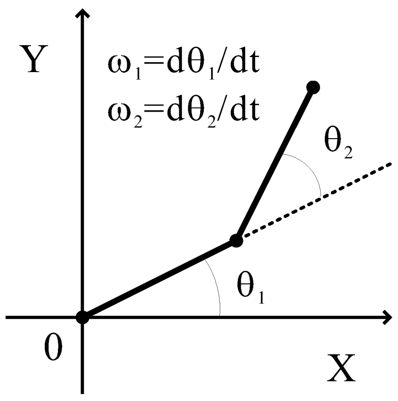

**Fig. 7** Two-joint robot arm.

dynamical system into a pattern that does not change sharply. We refer to such a pattern as the modification pattern instead of the context pattern.

Figure 6 portrays a simple model of mutual interaction, where two nonmonotone neural networks of the same size are mutually linked through a two-layer network consisting of normal binary neurons. Each layered network transforms the output pattern of one neurodynamical system (nonmonotone neural network) into the modification pattern for the other system. In the following experiment, however, it has the sole function of retrieving the state vector (consisting of binary components) from the output pattern with many zero components (corresponding to desensitized elements) because we performed encoding so that the state vector would not change sharply.

## 4.2 Computer Simulation

As a simple example of application, we trained this model to learn the movement of a computer-simulated two-joint robot arm without actuators (Fig. 7). Specifically, each neurodynamical system with 2000 elements represents the normalized state $(\theta_i, \omega_i)$ of one arm joint $(-1 \leq \theta_i, \omega_i \leq 1,\ i = 1, 2)$. It learned 10 trajectories for the corresponding joint, undergoing selective desensitization by the modification pattern that depends on the state of the other system. The initial states of the learned trajectories of the arm are listed in Table 4.

After 10 cycles of training, we gave various initial states to the model. Figure 8 presents an example of the behavior of the model, where the actual trajectory of the arm and the trajectory recalled by the model are displayed,

**Table 4** Initial states of the robot arm given to the model.

| | $\theta_1$ | $\omega_1$ | $\theta_2$ | $\omega_2$ | | $\theta_1$ | $\omega_1$ | $\theta_2$ | $\omega_2$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −0.20 | 0.69 | −0.21 | −0.33 | 6 | 0.00 | 0.00 | 0.00 | 0.90 |
| 2 | −0.15 | 0.70 | 0.28 | 0.55 | 7 | 0.00 | 0.00 | 0.50 | −0.90 |
| 3 | 0.00 | 0.00 | −0.50 | −0.90 | 8 | 0.00 | 0.00 | 0.50 | 0.90 |
| 4 | 0.00 | 0.00 | −0.50 | 0.90 | 9 | 0.47 | −0.68 | 0.41 | 0.95 |
| 5 | 0.00 | 0.00 | 0.00 | −0.90 | 10 | 1.00 | 0.61 | −0.13 | −0.62 |

respectively, in the upper and lower panels. Furthermore, Fig. 9 shows an example in which the initial state is different only for $\theta_2$. The figure shows that the arm trajectory is well reproduced by the model in both cases, and that it is well produced for all learned trajectories. Results show that each neurodynamical system makes state transitions depending on the state of the other system, so that the model can learn and recall complex spatiotemporal patterns.

Although this robot arm task is very simple and although the advantages of the model would be insufficiently demonstrated, we think that the model is important for additional development of brain-like computing because, by linking through modification patterns, many neurodynamical systems can mutually interact while preserving modularity (effective connections can be switched on and off easily). They might learn the dynamics of a more complex system from a few sample trajectories. Exploring this possibility remains as a subject for future study.

## §5 Conclusion

We have presented a neurodynamical system that forms inferences according to context-dependent state transitions. We also demonstrated its many advantages over existing inference systems, such as a powerful capability of analogical reasoning and simple treatment of exceptions. We have also proposed a method of linking multiple neurodynamical systems and showed that two dynamical mutually interacting systems can reproduce complex movements of a two-joint robot arm.

Although these models do not directly model the brain, they are similar to the brain in that information is not represented by symbols or local representations but is represented distributedly by patterns. Furthermore, their basic principles, namely trajectory attractors and selective desensitization, are sug-

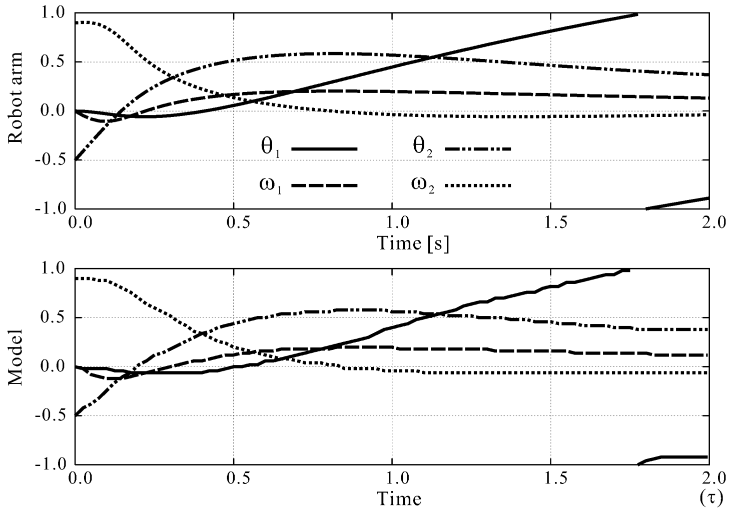

**Fig. 8** State transitions from the initial state $(\theta_1, \omega_1, \theta_2, \omega_2) = (0.0, 0.0, -0.5, 0.9)$.

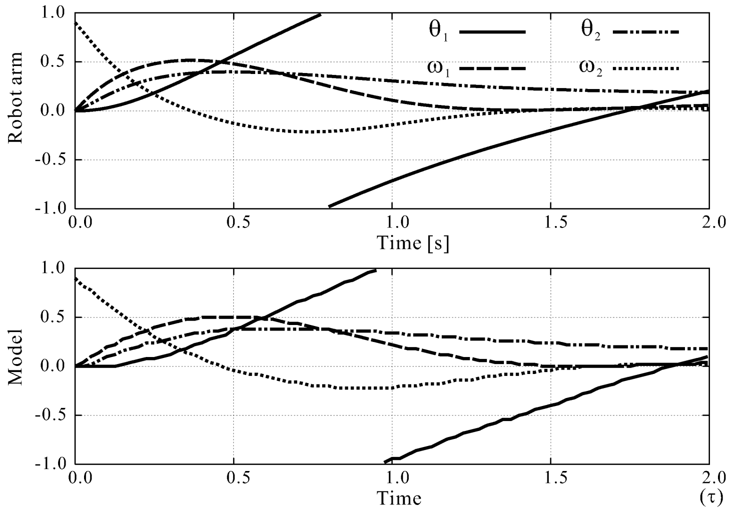

**Fig. 9** State transitions from the initial state $(\theta_1, \omega_1, \theta_2, \omega_2) = (0.0, 0.0, 0.0, 0.9)$.

gested to be used in the brain [5, 6, 27]. This fact, together with the results of this study, suggests the great potential of such brain-like computing.

However, the present models leave much room for improvement and many problems remain. For example, the mode of encoding objects to patterns has not been examined, particularly for abstract concepts of which information is not detected by sensors. Comparison with humans or animals will also be necessary for evaluation and further improvement of the models. Nevertheless, we believe that future study of our brain-like computing models will bring a major breakthrough in intelligent information processing that is required not only to address real-world problems but also to address important problems in the era of information explosion.

***Acknowledgment*** This work was partly supported by a Grant-in-Aid for Scientific Research on Priority Areas (18049008, 19024010, and 21013007) from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

## References

1) Harnad, S., "The symbol grounding problem," *Physica D, 42*, pp. 335–346, 1990.

2) McCarty, J. and Hayes, P. J., "Some philosophical problems from the standpoint of artificial intelligence," *Machine Intelligence, 4*, pp. 463–502, 1969.

3) Sandewall, E., "An approach to the frame problem and its implementation," *Machine Intelligence, 7*, pp. 195–204, 1972.

4) Morita, M. and Suemitsu, A., "Computational modeling of pair-association memory in inferior temporal cortex," *Cogn. Brain Res., 13*, pp. 169–178, 2002.

5) Suemitsu, A., Morokami, S., Murata, K. and Morita, M., "Computational examination on the dynamics of recall activity in the inferior temporal cortex," in Proc. of the 2002 IJCNN, pp. 136-141, 2002.

6) Suemitsu, A. and Morita, M., "A neural network model of context-dependent neuronal activity in the inferotemporal cortex," in *Proc. of the 2006 IJCNN*, pp. 685–690, 2006.

7) Rumelhart, D. E., McClelland, J. L. and the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition volume 1: Foundations*, MIT Press, Cambridge, 1986.

8) Blelloch, G. E., "CIS: A massively concurrent rule-based system," in *Proc. of the AAAI-86*, pp. 735–741, 1986.

9) D'Avila Garcez, Artur S., Lamb, Luis C. and Gabbay, Dov M., *Neural-symbolic cognitive reasoning*, Springer-Verlag, Berlin, 2009.

10) Touretzky, D. S. and Hinton, G. E., "Symbols among the neurons: details of a connectionist inference architecture," in *Proc. of the IJCAI-85*, pp. 238–243, 1985.

11) Touretzky, D. S. and Hinton G. E., "A distributed connectionist production system," *Cogn. Sci., 12*, pp. 423–466, 1988.

12) Samad, T., "Towards connectionist rule-based systems," in *Proc. of the IEEE ICNN-88*, 2, pp.525–532, 1988.

13) Barnden, J. A. and Pollack, J. B., *Advances in connectionist and neural computation theory, 1, High level connectionist models*, Ablex, Norwood, 1991.

14) Holyoak, K. J. and Barnden J. A., *Advances in connectionist and neural computation theory, 2, Analogical connections*, Ablex, Norwood, 1994.

15) Gallant, S. I., *Neural network learning and expert systems*, MIT Press, Cambridge, 1993.

16) Elman, J. L., "Finding structure in time," *Cogn. Sci., 14*, pp. 179–211, 1990.

17) Morita, M., Murata, K. and Morokami, S., "Context-dependent sequential recall by a trajectory attractor network with selective desensitization," in *Proc. of the 3rd ICNNAI*, pp. 235–238, 2003.

18) Morita, M., Matsuzawa, K. and Morokami, S., "A Model of context-dependent association using selective desensitization of nonmonotonic neural elements," *Systems and Computers in Japan, 6*, pp. 73–83, 2005.

19) Morita, M., "Memory and learning of sequential patterns by nonmonotone neural networks," *Neural Netw., 9*, pp. 1477–1489, 1996.

20) Kiani, R., Esteky, H., Mirpour, K. and Tanaka, K., "Object category structure in response patterns of neural population in monkey inferior temporal cortex," *J. Neurophysiol., 97*, pp. 4296–4309, 2007.

21) McCloskey, M. and Cohen, N., "Catastrophic interference in connectionist networks: The sequential learning problem," *The Psychology of Learning and Motivation, 24*, pp. 109–164, 1989.

22) Klar, D., Langley, P. and Neches, R., *Production system models of learning and development*, MIT Press, Cambridge, 1987.

23) McDermott, D. and Doyle, J., "Non-monotonic logic I," *Artif. Intell., 13*, pp. 41–72, 1980.

24) McCarthy, J., "Circumscription; A form of non-monotonic reasoning," *Artif. Intell., 13*, pp. 27–39, 1980.

25) Reiter, R., "A logic for default reasoning," *Artif. Intell., 13*, pp. 81–132, 1980.

26) Hanks, S. and McDermott, D., "Default reasoning, non-monotonic logic, and the frame problem," in *Proc. of the AAAI-86*, pp. 328–333, 1986.

27) Suemitsu, A., Miyazawa, Y. and Morita, M., "A model of the activity of the hippocampal neurons based on the theory of selective desensitization," *Neural Information Processing (Part I), Lecture Notes in Computer Science, 5506*, pp. 383–390, 2009.