Intelligence dynamics: A concept and preliminary experiments for open-ended learning agents

Article *in* Autonomous Agents and Multi-Agent Systems · December 2009

DOI: 10.1007/s10458-009-9076-y · Source: DBLP

Masahiro Fujita
Sony Corporation

Auton Agent Multi-Agent Syst (2009) 19:248–271
DOI 10.1007/s10458-009-9076-y

Intelligence Dynamics: a concept and preliminary experiments for open-ended learning agents

Masahiro Fujita

Published online: 12 February 2009

Abstract We propose a novel approach that aims to realize autonomous developmental intelligence called Intelligence Dynamics. We emphasize two technical features of dynamics and embodiment in comparison with the symbolic approach of the conventional Artificial Intelligence. The essential conceptual idea of this approach is that an embodied agent interacts with the real world to learn and develop its intelligence as attractors of the dynamic interaction. We develop two computational models, one is for self-organizing multi-attractors, and the other provides a motivational system for open-ended learning agents. The former model is realized by recurrent neural networks with a small humanoid body in the real world, and the later is realized by hierarchical support vector machines with inverted pendulum agents in a virtual world. Although they are preliminary experiments, they take important first steps towards demonstrating the feasibility and value of open-ended learning agents with the concept of Intelligence Dynamics.

Keywords Open-ended · Dynamics · Embodiment · Prediction · Intelligence Dynamics · Recurrent neural networks · Intrinsic motivation · Flow theory

1 Introduction

Intelligence Dynamics is a research field for understanding and realization of “intelligence”, as opposed to symbolism, which is the hallmark feature of conventional Artificial Intelligence (AI) [23]. Recently many researchers in various fields have pointed out problems of traditional AI. New research approaches have been proposed to address “Intelligence”. These have been independently developed in Brain Science [14], Cognitive Science [25], Psychology [26], Robotics and AI [2,34]. Our aim in proposing “Intelligence Dynamics” is to make breakthroughs in the study of “intelligence” by further integrating these activities

M. Fujita (✉)
System Technologies Laboratories, Sony Corporation, Shinagawa-ku, Tokyo, Japan
e-mail: masahirof@jp.sony.com

across the various research fields. The common features of these approaches are summarized as (1) embodiment, (2) dynamics, and (3) constructive approach.

- *Embodiment*: Recently many researchers have pointed out that it is necessary for "intelligence" to have a body interacting with an environment [2,20,34]. This approach insists that the cognitive process should be described within both the neural/brain system and the environment. In contrast, traditional AI usually insists that the cognitive process resides only in the brain.
- *Dynamics*: The Dynamics systems approach emphasizes that the cognitive process should be described by both bottom-up sensory-motor dynamics and top-down abstracted pattern dynamics [34]. The competition and cooperation of bottom-up and top-down dynamics naturally lead to the global dynamics for cognitive process. In contrast, traditional AI usually describes the top-down cognitive process by symbols and logic, which make competition and cooperation of sensory-motor signals difficult to handle.
- *Constructive approach*: The constructive approach asserts that in order to understand a function of the brain and its cognitive processes, we should build a body to evaluate the resulting hypotheses, theories, and models. In contrast, conventional brain science is only anatomical and analytical. Conventional cognitive science is also analytical, but recently cognitive researchers begun to explore and adopt a constructive approach [25].

Let us briefly compare Intelligence Dynamics and conventional AI. What kind of intelligence is studied in conventional AI? Originally the goal of conventional AI was to realize human-level thought and human behavior [8]. However, the main approach focused on logical reasoning and logical behavior. Therefore, symbolic representations and their manipulations by logic formula were largely employed. Based on an idea of building knowledge using symbolic representations and logical formulation, researchers tackled the problem by making symbol representations of the problem based on logical formula and manipulating the resulting symbols based on logical formula. Natural language is also addressed in a similar way. A word sequence is represented as symbols and is analyzed based on a representation of grammar rules. The system then generates the correspondence between the words and categories such as subject, predicate, and so on. The meaning of the symbols is given by a set of the frame representations.

However, there are well discussed concerns in the use of the conventional AI approach as described above, some of which include the frame problem [29] and the symbol grounding problem [13]. The frame problem indicates that it is difficult or impossible to describe the real world with logic based symbol representations, because the representational complexity becomes overwhelming when a dynamically changing real world is considered. However, symbolic representation is very powerful in well defined static worlds such as chess even when the number of possible world states in huge. The fact that a computer chess program using conventional AI beat the human world chess champion in 1997 shows the usefulness and power of the conventional AI.

Intelligence Dynamics emphasizes relations among observed, predicted, and recalled time sequences in sensory-motor space, which are the results of interactions between a body and its environment. "Intelligence" must be described by these relations in order to be grounded. Regarding dialogue using natural language, Intelligence Dynamics emphasizes the levels where various kinds of intentions such as "drawing attention" and "shared attention" are carried out in a non-symbolic manner. These levels are acquired or self-organized through interactions between a body and its environment, which can include the presence of other agents such as humans. From self-organization of these levels, we believe that verbal interaction, which was considered as a symbolic process in the conventional AI approach could emerge under certain social constraints, where physical and emotional grounding are achieved. However, we also recognize that this remains an important open challenge.

Another important factor in Intelligence Dynamics is imitation based on embodiment. The importance of imitation is also emphasized in brain science with "mirror neurons" [27] and in cognitive science as "mimesis" [10]. The existence of mirror neurons implies that a human interprets another human's actions as actions by himself, providing an understanding of their meaning. Based on the interpretation of the mirror neurons, embodiment plays an important role in understanding the meaning of behaviors. Mimesis is defined as the activities that action patterns acquired by imitation are reused in the rehearsal of motions and in communication with others. Thus, mimesis emphasizes the importance of imitation, which is a basic function of communication.

In summary, Intelligence Dynamics constitutes a research field for AI by emphasizing the importance of embodiment and dynamics. In Intelligence Dynamics the intelligence is not symbolic representation provided by a system designer or programmer, but it is acquired through interactions between a body and the environment which can include other agents.

The rest of this article will introduce some of our preliminary experiments based on the Intelligence Dynamics approach. First, we review some pioneering work related to the Intelligence Dynamics approach. Then, we introduce a generic common model by abstracting and integrating these models. We emphasize that the prediction function is a key component of the generic model. Using the generic model, we briefly describe the entire architecture of our Intelligence Dynamics Model. Then, we describe two implementation experiments. In the first one we examine properties of dynamics attractors, and in the second we propose a key idea for open-ended systems that can autonomously learn by selecting tasks that are significantly challenging for the system itself.

2 Reviews and generalization

There are many works related to Intelligence Dynamics. Almost all of them point out the importance of some capabilities in the developmental process such as imitation and joint attention. Of course these functions are very important for intelligence. However, we would like to focus more on the system architectural view point for developmental intelligence. There are some pioneering works that deal with this issue. Let us consider the following three works and identify their essentials properties: (1) MOSAIC [14] (Fig. 1a), Mimesis Model [15] (Fig. 1b), and Recurrent Neural Networks with Parametric Bias (RNNPB) [34] (Fig. 1c).

MOSAIC as shown in Fig. 1a is composed of many prediction and control pairs for sensory-motor signals such as the locally weighted projection regression (LWPR) [35] as a prediction function. Using MOSAIC the authors succeed to develop the humanoid named DB, which learns performances of juggling, devil stick handling, and so on [14]. We are not going into the details of MOSAIC, but would like to consider its essential functions and properties. The important difficulty of MOSAIC is how to select a controller that properly works in each situation. It selects a controller based on the goodness of the corresponding prediction function. In a more advanced version of MOSAIC [11], the control signal is generated by adding the controllers' output weighted by the goodness of the corresponding predictors. The weighted values based on the predictors' "goodness" function are used as inputs to the next layer, which is again composed of many prediction and control pairs for the weighted values.

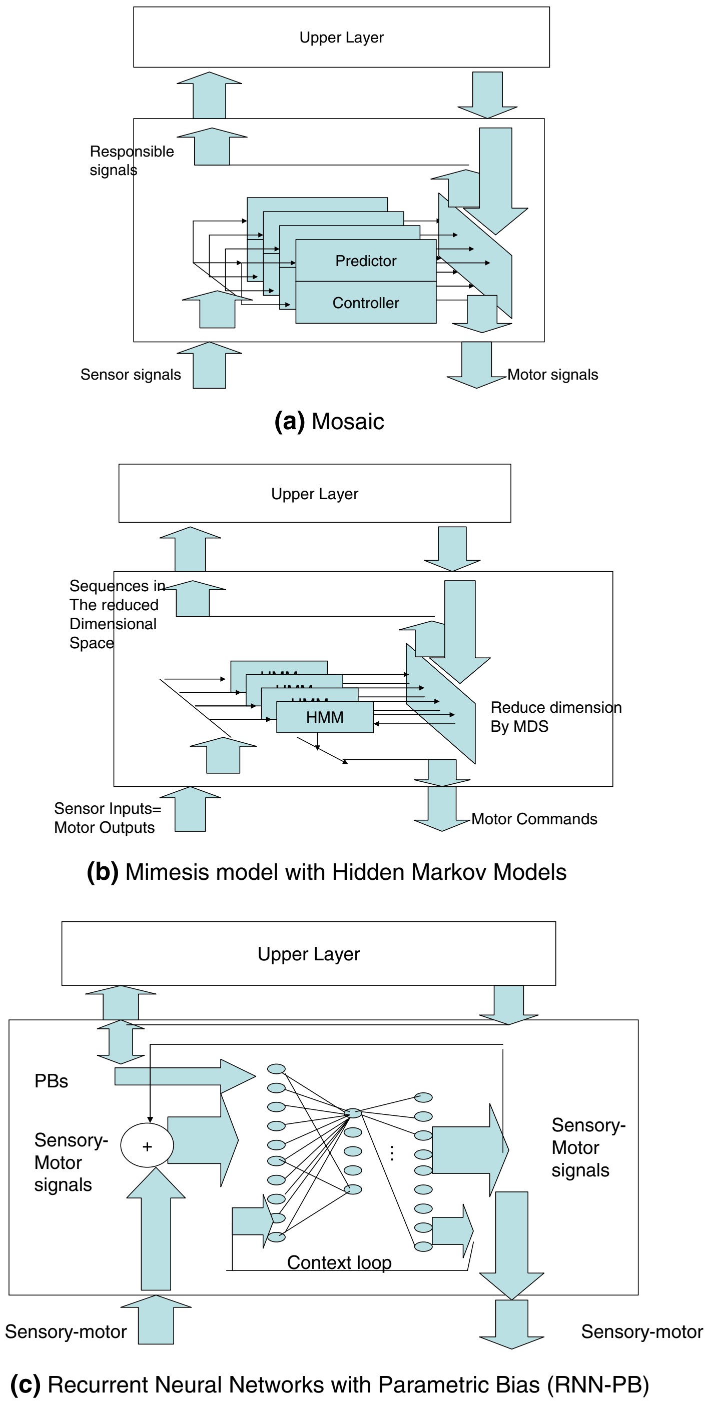


Fig. 1 Some examples of the prediction functions used in **a** Mosaic, **b** Mimesis with HMMs, and **c** RNNPB

In summary the prediction function and a layered architecture are the important points of MOSAIC.

The Mimesis Model as shown in Fig. 1b is composed of many hidden Markov models (HMMs), which are trained with motion data of human actions such as walking, kicking, jumping and so on [5,4,15]. Then, distance metrics are introduced among the HMMs. Based on the metrics the HMMs are set in a low dimensional space named Proto-Symbol space with multi-dimensional scaling (MDS) algorithm [7]. Then, a point in Proto-Symbol space represents a motion sequence. When a new motion sequence is input, a Mimesis system can recognize the sequence as a point in Proto-Symbol space. Multiple motion sequences such as walking and then kicking can be recognized as a sequence in Proto-Symbol space. If we can consider that the sequence in Proto-Symbol space is input to the upper layer, then the Mimesis system can be considered as the layered architecture.

RNNPB as shown in Fig. 1c is trained by a sequence of sensory-motor signals [35]. The input of RNNPB is a sample of the sensory-motor signals at time t and the output is a sample at time $t + 1$. Thus, it can be considered as a learnable predictor. Actually RNNPB can learn multiple attractors of dynamics, which can be selected by parametric bias (PB) signals. Then, we can consider that PB signals are input to the upper layer, which is again composed of RNNPB. Thus, the RNNPB system can also be considered to be a predictor with layered architecture.

All these models can be considered as encoding dynamics attractors using predictors and in some cases controllers. The implementations of the predictors are done using models such as LWPR, HMMs, or RNNPB. Furthermore, all these models have a hierarchical architecture. The upper layer is organized to encode the attractors in the lower layer. In order to achieve a generalization property the dimension of the output signals from lower layer is compressed.

We generalize all the models in Fig. 1a–c to separate the computational model of intelligence from its implementation. Figure 2 shows the generalized model for a hierarchical architecture of prediction and control functions, where embodied interactions are memorized and stored as attractors. The attractors are embedded into a low dimensional space, which provides an input to an upper layer. In the upper layer, time sequences in the input

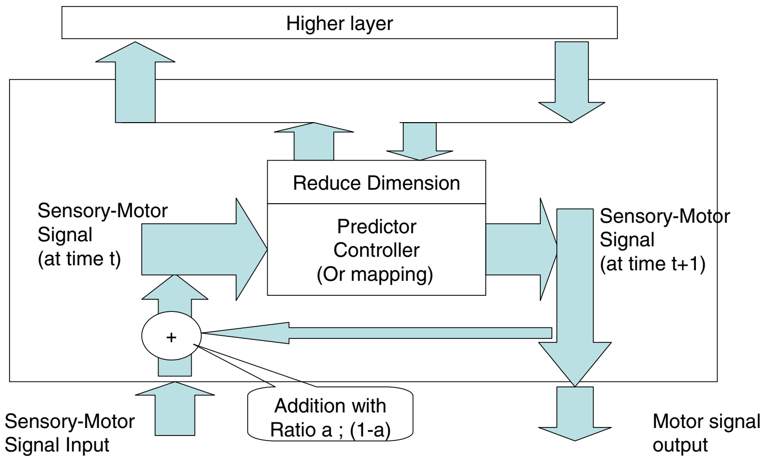

Higher layer
Reduce Dimension
Sensory-Motor Signal (at time t)
Predictor Controller (Or mapping)
Sensory-Motor Signal (at time t+1)
+
Sensory-Motor Signal Input
Addition with Ratio a ; (1-a)
Motor signal output

Fig. 2 Generic prediction function model for Intelligence Dynamics

space again can be considered as attractors, which are again embedded into a lower dimension space.

Regarding the learnable prediction or control functions, many models are proposed such as RNNPB, LWPR SVR [32], HMM, and so on. Regarding the method to encode the attractors into a low dimensional space, there are a range of method that can be used such as self-organizing map (SOM) [19] and MDS [7]. In RNNPB, errors are also back propagated not only through weights but also through PBs, so that attractors are encoded into a low dimensional PB space.

Thus, there are many ways to implement the concept model of Fig. 2. Further studies have to be done to investigate convergence speed, scalability, and stability/plasticity,

3 Entire architecture for Intelligence Dynamics

As we described in the previous section, in the literature of Intelligence Dynamics references, many researchers pursue a range of different goals such as behavior emergence and imitation learning. The generic architecture described in the previous section is a core component of an intelligent system, it can be considered as a part of an open-ended system, which is our ultimate objective. In order to achieve this objective, we should not over-focus on any individual part of the system, but instead consider the entire autonomous system so that various key behaviors emerge as a continuous evolving system. In this section, keeping in mind the suggestions in the previous section let us speculate about the entire architecture for Intelligence Dynamics.

In order to build a fully autonomous agent from scratch the emergence of functions need to simulate the evolution of intelligence life which is a major challenge yet to be realized. Therefore, it is better to breakdown the problem into two phases as a means to reduce the complexity of the problem, and we consider two kinds of functions, first those acquired by evolution and second those which are developed through experience.

3.1 Functions associated with evolution

The following two aspects are addressed for the functions associated with evolution.

1. Functions associated with evolution, such as reflex, instinct behaviors, and behavior motivations
2. Development scenario acquired by evolution

The first item is necessary for living things to survive in the real world. Without reflex and instinct behaviors, many individuals would die with much higher probability. In addition, the reflex and instinct behaviors help spontaneous learning. In reinforcement learning [33], spontaneous search in its action-space is necessary. However, in general, the dimension of the action-space is huge for searching appropriate actions. Reflex and instinct behaviors provide important and relevant constraints to behavior generation, and thus reduces the size of the action-space. Imitation can be considered as a kind of reflex and instinct behaviors.

In addition, basic behavior motivations, which have strong relations to instinct behaviors, are very important. Especially, intrinsic motivations such as curiosity, manipulation, and achievement motivations which are necessary for the development of open-ended systems. Later we will introduce our system with intrinsic motivations, which is also related to Flow theory [9].

In the development process, a living thing grows physically. The muscle power, weight, and the length of arms and feet change, and in parallel the living thing acquires skills to control its body while growth continues. Similarly, sensors increase in their sensitivities and resolution over time as well. Moreover, targets of interest change according to growth. Namely, development scenarios such as the growth of the body, the performance of motors and sensors, and related behavior tendencies are encoded in the genes. The development scenario helps the efficiency and convergence of learning. In particular the development scenario helps to specify the complexity of tasks in terms of proper ordering and timing. According to this scenario, after the system achieves a goal, it then has an interest in a new target, and moves on. This is similar to the idea of controlling the complexity of the environment [2]. The development scenario is considered a result of evolution. In our system we allow the setting of the development scenario to be done manually.

According to the development scenario, while changing its targets of interest, freezing and defreezing the degrees of freedom [6] and other factors, the system implicitly learns the models of its body and the environment in a self-development fashion through its interactions with the environment. If learning is based on evaluation functions set by the development scenario, some primitive motions and motion sequences that satisfy the evaluation function will develop over time. It should be noted that the motivation of the learning is essential for an open-ended system to continue developing. Thus, how to design a mechanism by which proper motivations emerge is one of the main topics of this line of research.

As addressed in the studies of reinforcement learning, a simple probabilistic behavior selection algorithm such as *Softmax* tends to fail in the real world because the size of the state and action spaces becomes too large for appropriate solution to be found. To help direct the convergence of learning, reflexive and instinct behaviors play an important role, because they limit the possible actions in a particular situation, and usually leading to a good selection yielding a reward. In addition, the existence of a caregiver and the use of imitation as an instinct behavior are also important for the convergence of learning. Imitating a caregiver's behaviors helps in organizing the behavioral primitives that are used in social interactions. Imitation and teaching by a caregiver assist in the learning of the implicit models of its body and environment as well.

Thus, the functions associated with evolution such as reflex and instinct behaviors, behavior motivations, the development scenario, and imitation are important for efficient development, and in our approach these functions are allowed to be set manually.

3.2 Functions associated with development

Then, the embodied general prediction model with a proper developmental scenario and intrinsic motivations enables a robot to learn the body, objects, and environment model. The general prediction model described in the previous section can handle relations between observed and predicted states of sensory-motor signals. In order to learn the relations, intrinsic motivation tries to select appropriate actions for the predictor or the controller, so that its learning progresses well. In this model, top-down signal flows try to give a target state, and bottom-up signal flows try to give a current state in the real world. The target states are memorized by reward signals as innate functions. Thus, in a particular situation, a target state is associatively recalled and top-down signals are generated.

Then, if the target state is set, the system can search the action sequence derived directly from the real world. If the prediction and control functions are well trained in the situation, the system can easily reach the target state. However, if it is not well trained the system has to explore the action sequence by trying some actions and deciding which action will allow it to reach the target.

Once the system finds the action-sensor sequence via exploration, it is important to memorize the sequence for reuse in the future. In the reinforcement learning framework it is memorized by learning the value function, which gives the value of the action to be selected. A table look-up memory is often used to memorize the action with maximum value at each state. In our framework, it is memorized by learning the dynamics attractors, which are mapping functions on continuous action-sensor state spaces. In order to do this, it is necessary to record the action sequence and the corresponding sensor sequence from the outcome of the exploration.

There are two main ways of getting the sequence as training data for the controller and the predictor to form the dynamics attractor. The first one is done using the exploration results, and the second through human teaching. Here we focus on the use of the exploration result. Again, there are two key ways to do the exploration. One is to explore in the real world, which is a typical reinforcement learning strategy. Another is exploring in a simulated world, which can be done using a planning and a rehearsing function. The planning function explores the proper actions to get close to the target state by using the predictor, which can predict the state in the next time step. The rehearsing function simulates the sensory-motor signal sequence to get close to the target by using the predictor and the controller. The action sequence to reach the target state can be used as teaching signal pairs for the controller. This trained controller is more accurate than the previous controller used for the rehearsing function. We consider that it could be considered as cortical–cerebellum collaboration [1].

The curiosity and the manipulation motives can be considered as the learning motivations of the prediction and the control function. In addition to these intrinsic motivations, achievement motive, which is one of other intrinsic motivations, can be considered as the learning motivation for planning and rehearse functions.

Considering the features described above, now we can describe our approach which realizes a continuously developing open-ended system hat has been implemented on a real state-of-the-art robot which we describe in later sections. Adding some other features, we describe the entire architecture of autonomous agent for an open-ended system as shown in Fig. 3. The main features of the system are as follows:

(1) Prediction and control functions for not only external sensory-motor signals but also internal state signals for intrinsic and external motivations. The generic architecture as shown in Fig. 2 can be used here.
(2) Planner and rehearse functions using the predictor and controller.
(3) Intrinsic motivations of curiosity, manipulation, and achievement, which enable the predictor, the controller, and planner to develop continuously.
(4) The target state associated with evaluation functions.
(5) Model of cortical–cerebellum collaboration. The controllers can learn suitable action sequences in a supervised fashion by sensor-action sequences of exploration results by the Planning/Rehearse Function.

It should be noted here that the architecture is still under development. The complete Intelligent Dynamics design is still somewhat speculative, and we illustrate the components of the design in subsequent sections. In particular, Sect. 4 features and describes (1) above, and Sect. 5 focuses on (1), (2) and (3).

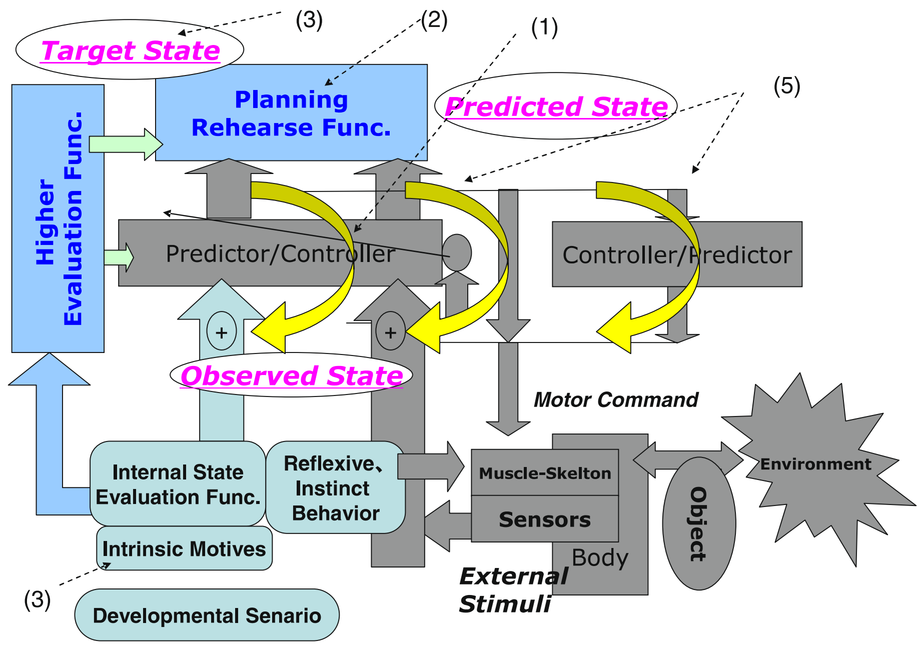


Fig. 3 The entire architecture for open-ended system

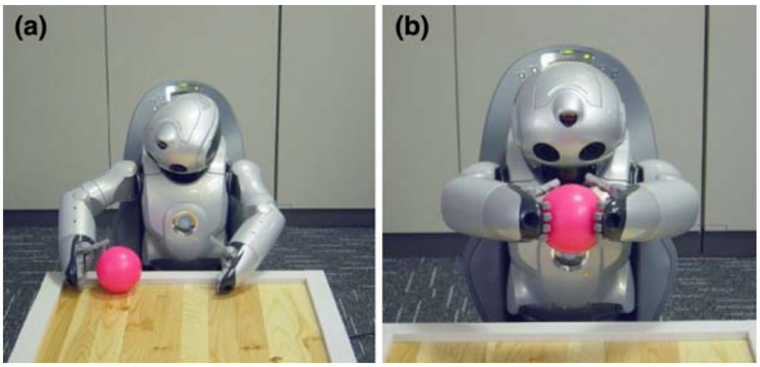


Fig. 4 Ball handling behavior task. **a** Ball rolling behavior. **b** Ball picking up behavior

4 Dynamical systems approach with RNNPB

Based on the entire architecture described in the previous section, in this section let us report the examination and evaluation of the prediction and control functions, which were addressed in Sect. 2. We thoroughly examined RNNPB using a small state-of-the-art humanoid robot named QRIO [12] designed and built at Sony. We examined two different tasks, which are ball handling (Fig. 4) and block handling behavior task (Fig. 5). For details of the experiments and the analysis, please refer to [16,24]. In this article, we would like to provide a summary of the experiments as a means to describe and illustrate the Intelligence Dynamics design.

The main points are (1) we demonstrate that multiple attractors can be encoded in RNNPB, (2) inter-attractor dynamics emerges and generates natural transition behavior patterns, and

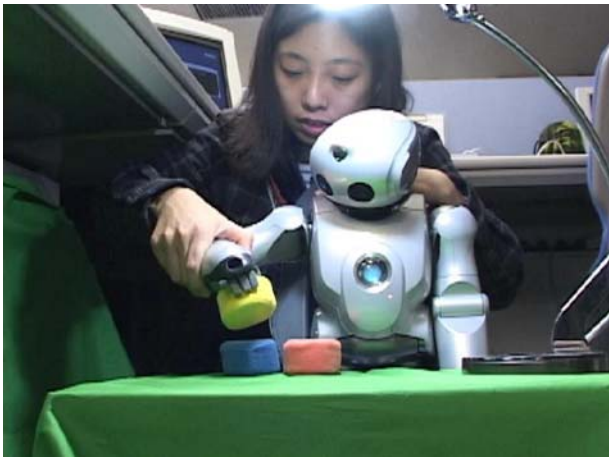


Fig. 5 Block handling behavior task

(3) human guidance can be used to assist the generation of inter-attractor dynamics that eventually transform into the learned attractor dynamics. In Fig. 4a QRIO passes the ball across the table from one hand to another, in Fig. 4b QRIO picks up the ball, and in Fig. 5, a human assistant holds QRIO's hand and "shows" him how to stack blocks in a specific order.

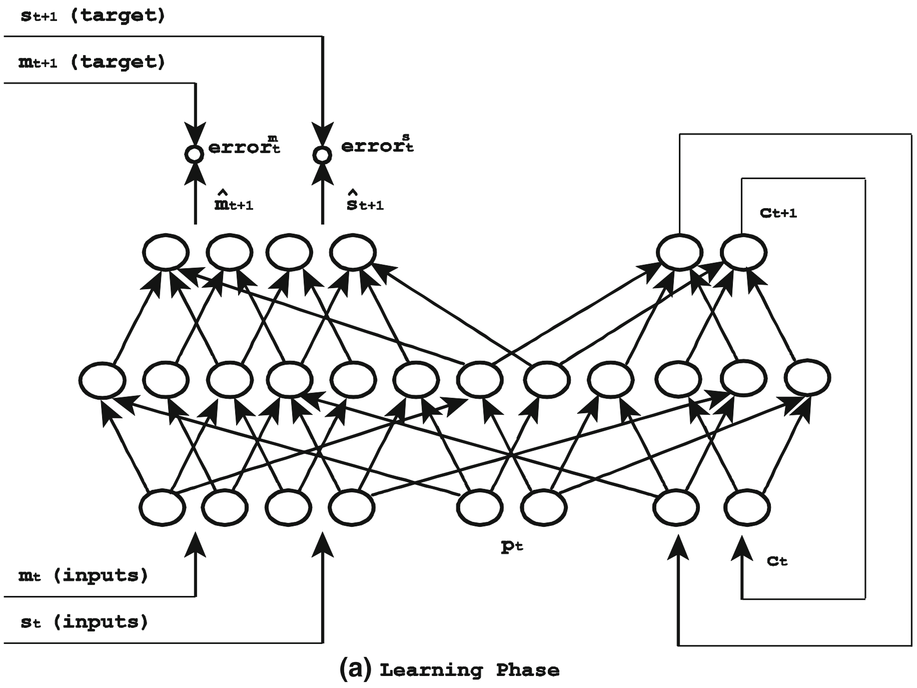
4.1 Model description

The RNNPB model that QRIO uses to complete his tasks has the same architecture as the conventional Jordan-type RNN model [17] except for the PB nodes in the input layer. Figure 6 shows the configuration of the RNNPB model in the learning phase (a) where the RNNPB is trained with sensory-motor flow sequences, and in the interaction phase (b) where the trained RNNPB generates situated motor outputs according to incoming sensory inputs. For the normal input and output nodes, two types of operations are performed at the same time: open-loop and closed-loop operations. In an open-loop operation outputs of the network $(\hat{s}_{t+1}, \hat{m}_{t+1})$ are calculated as a result of a prediction from the current inputs (s_t, m_t). In a closed-loop operation the previous prediction outputs are copied to the current inputs, and outputs are calculated using the feedback information. This feedback enables look-ahead prediction (rehearsal process) for an arbitrary number of future steps without perceiving the actual inputs.

There are context nodes (c_t, c_{t+1}) in both the input and output layers. The output of the context nodes is copied to the context nodes in the input layer. The internal state is recursively computed for future steps utilizing the recurrent feedback loop for the context nodes. There are PB nodes p_t in the input layer. These PB nodes are the additional network variables that can be manipulated to learn and generate diverse behavioral patterns.

The common structural properties of the training data sequences are acquired as connection weights by using the backpropagation through time (BPTT) algorithm [28]. On the other hand, the specific properties of each individual time sequence are simultaneously encoded as PB values. Therefore, the modulation of the PB values shifts the modes of the behavior pattern. In the processes of learning and recognition, the PB values are iteratively computed utilizing the error between the target sensory-motor sequence and the predicted sequence.

In this ball handling task, the sensor values are joint angles $(\theta_1, \theta_2, \ldots, \theta_8)$ and ball position (x, y, z), and the motor command values are joint angle commands.



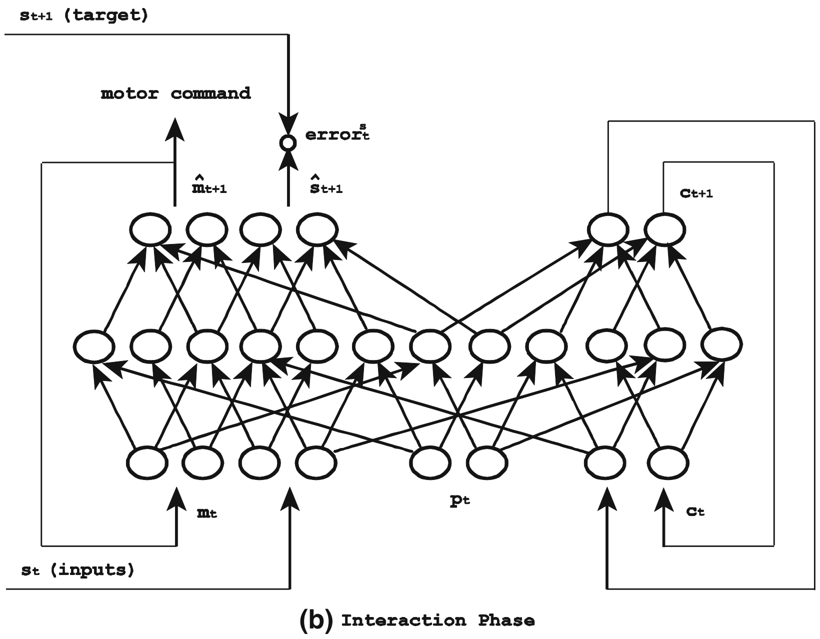


Fig. 6 RNNPB architecture and its input and output signals in **a** learning phase and **b** interaction phase

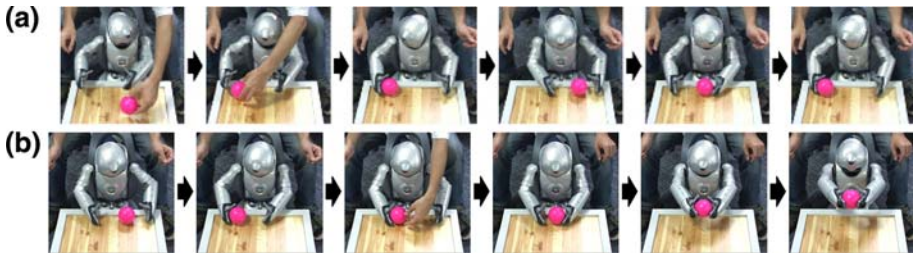


Fig. 7 Snapshots of **a** Ball rolling behavior and **b** Ball picking up behavior

4.2 Ball handling behavior experiment

In the first experiment, the robot learns two different types of ball handling behaviors, as shown in Fig. 7. One (a) is “rolling a ball” in which the robot swings both arms alternately to roll a ball on a table from left to right and vice versa. The other (b) is “lift up a ball”, which is to put the robot’s hands together to lift up a ball on a table vertically and then release its hands to drop the ball.

In the learning phase, the robot learns two different ball handling behaviors directly from human teaching. In the teaching process, a human user grabs the robot’s arms and guides them to perform the target ball handling behaviors using an actual ball while the servo gain of the robot arms is almost set to zero. In this study, the reference trajectory is simply obtained as a copy of the measured arm movement in the direct teaching by human users. It is important to note that during the teaching process these two behaviors are given as separate sequences. Thus the robot never learns the transition between them.

After the learning, we examined how the robot with the trained RNNPB could generate the two different learned ball handling behaviors. We also observed how the ongoing behavior could alternate between them depending on the situational differences between the robot and the ball.

Figure 7a shows a sequence of snapshots for the ball rolling behavior generated by the robot. When the ball was rolling from the front to the left side, the robot hit it by the right hand. Then the ball rolled to the opposite side and the robot hit it with its left hand. This rolling a ball behavior was stably repeated several times. Figure 7b shows the sequence of snapshots for the ball lifting up behavior generated by the robot after the ball rolling behavior. When the human user stopped the ball in front of the robot, after a short while the robot started to hold it with both arms without any irregular movements and then lifted it up to a specified height. After this, the robot released the ball and then the ball was dropped in front of the robot. The robot started to hold it again. This ball lifting up behavior was also autonomously repeated for several times.

Figure 8 shows the time course of the whole interaction and the parametric bias values of the RNNPB. In Fig. 8, the plot at the top and the second row show the actual ball positions and the ones predicted by the RNNPB respectively. The plot at the third row shows the robot joint angles generated by the RNNPB (only 2 DOF are plotted among a total of 8 DOF). The plot at the bottom shows the parametric bias of the RNNPB.

We observed a transient status where two different behaviors were switched between (from rolling to lifting up) as a result of PB online adaptation according to change in the sensory sequence pattern. In this case, the ball position was changed by the human intentionally. Then, the ball’s motion was stopped in front of the robot body. This resulted in one of the

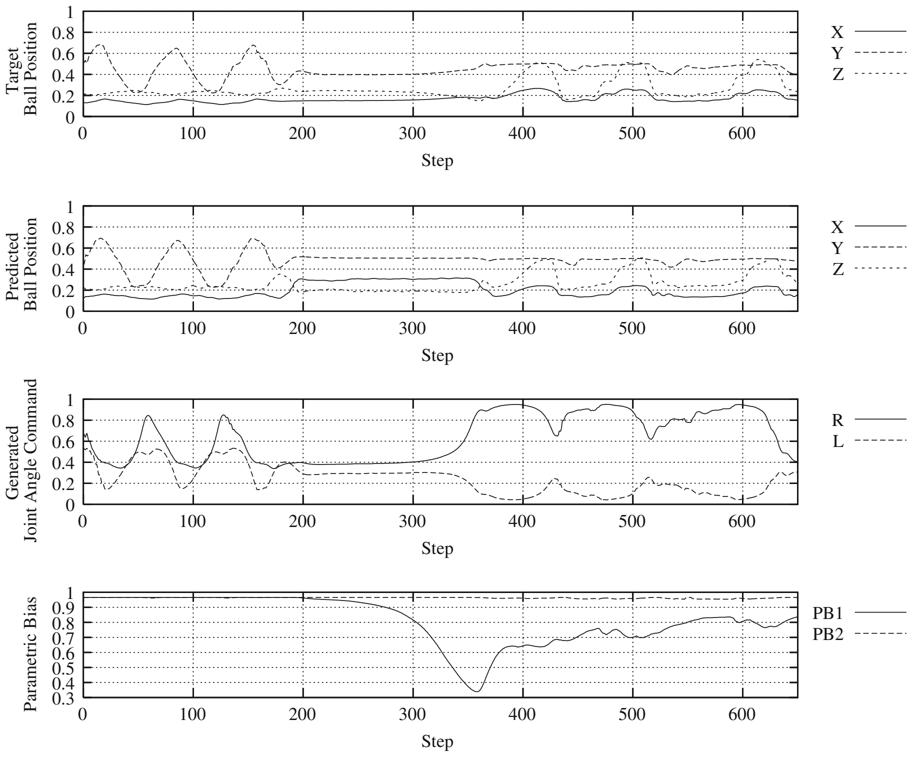


Fig. 8 Dynamic generation and switching of two learned behaviors in the ball handling task. The top row represents the measured position of the ball. The second and the third row represent the ball position predicted and the robot joint angles generated by the RNNPB. The bottom row represents the parametric bias of the RNNPB

PB values being decreased. After a while, ball lifting up behavior was generated according to the PB values.

In this experiment, we observed that the learned ball handling behaviors were well generated through interaction between the dynamics in RNNPB and the dynamics of the ball movement. Remember that the actual ball movement does not necessarily repeat exactly the same as the learned one; it is nondeterministic. Even under such noisy conditions, the learned behaviors were stably generated. The system seems to maintain certain robustness against unknown irregularities. We speculate that such robustness originates from the characteristics of attractor dynamics that emerges in the coupling, indeed grounding, between the RNNPB dynamics and the ball movement dynamics. It was also observed that the behavior switching could be performed smoothly despite the fact that the robot had not learned it explicitly; instead it learned it all by itself. (Recall that in the learning process, these two behaviors were just trained as separate patterns.) What is important is that genuinely novel behavior in terms of behavior transitions was generated spontaneously and autonomously utilizing emergent dynamical structures self-organized in the system.

4.3 Block handling behavior task

In the second experiment, the robot learned to put one block on top of another one from among three color blocks, using the right hand arm. The teaching process was the same as that of the first experiment except for the additional hand action for grabbing and releasing a block. The action of grabbing a block is activated with the tactile sensor on the palm of the robot's hand. Then, when the palm make compact with a block, the grabbing action is reflexively generated. On the other hand, the action of releasing a block is activated by a touch sensor on the robot's shoulder. When human users want the robot to release a block, they touch the robot's shoulder and then the robot releases it.

In this task, the robot perceives its arm movement and the movement of each of the three colored blocks and generates the arm movement including the hand action. In the current experiment, the tactile sensor on the palm and touch sensor on the shoulder were only used in the teaching phase and not used in the interaction phase. The sensory input vector consists of the current step encode values for joint angles of the robot arms and the center of 3D positions of the three color blocks. The motor output is the next step reference values of the joint angles of the robot arms and the binary hand action such as grabbing or releasing. In this experiment the robot learns two alternative behaviors to be selected in the same block layout situation where the yellow, the blue and the red block are located on the left-hand side, in the center and on the right-hand side respectively, in front of the robot. One behavior is to put the yellow block onto the blue one (B1). The other behavior is to put the red block on top of the blue one (B2).

After the robot learns these behaviors, we examined if the robot's behavior could be switched alternatively by means of partial human guidance. In this experiment, the control gains of the robot's arm were set to relatively low values where the robot can be moved by the motor command and by a human assistant. Figure 9 shows the snapshots of behavior switching guided by the human user. In this case, at first the robot autonomously tried to grab the yellow block for B1 due to the initial PB values and succeeded it. After the human user restored the yellow block to the initial position, when the robot tried to generate B1 again, the human user guided the robot by grabbing its arm to switch to B2. The human user continued the guidance to switch it while she or he senses the robot's "resistance-force" (resistance) back to her or his hand. After a while the robot starts to follow the user's guidance and the user feels "collaborative force" instead of the resistance-force by their hand.

Although the human user stopped providing guidance when the robot's hand was approaching the red block for B2, the robot's arm kept on moving to it instead of moving back to the yellow block for B1.

This robot experiment showed that human users can convey their intention to the robot through the force-based bodily interactions while the internal neuronal dynamics and the body dynamics of the robot continue without stopping.

4.4 Discussion

We demonstrated that the RNNPB can learn multiple dynamics attractors. Moreover, its intra- and inter-dynamics generate robust interactive behaviors with the environment and humans. One of the unsolved problems is scalability of memory capacity. How many attractors can be learnt in RNNPB in practice? Based on our experiments and personal communication

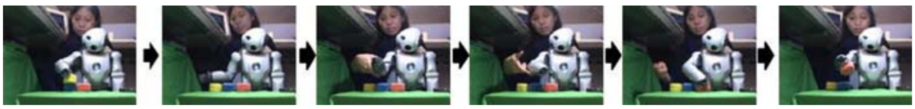


Fig. 9 Snapshot of block handling behavior

[35], it is difficult for RNNPB to learn more than 10 attractors, even if we use a large size of the RNNPB network. We need a layered architecture or other mechanism to make a large number of attractors or a meaningful way to organize and ground them.

There are two major types of memories in neural networks, which are the distributed type and the module type of memory. For example, RNNPB is distributed type of memory because the weights of RNNPB are shared for multiple attractors. In such neural networks learning a new attractor causes modification to the memories of other attractors. Thus, the learning becomes as difficult as the number of learnt attractors increase.

On the other hand, in modular types of memories the learning of each module doesn't cause the modification of memories of other attractors. This means it is possible to increase the number of attractors by increasing the modules and in this sense modular memories are naturally scalable.

We examined the module architecture using RNNs. Each module is implemented by a simple RNN, and by a self-organizing map (SOM) learning. We trained QRIO to have over 20 behavior patterns. As we described in the previous section, this accomplishment can be considered to be an implementation of the generic predictor-controller architecture described in Fig. 4, where a learnable predictor-controller is implemented by RNN, and the reduction of the dimensions is achieved using SOM. Consequently, it is named the RNN–SOM architecture.

The RNN–SOM can learn about 20 behaviors with 144 RNN modules. Thus, we can increase the memory capacity by using module type architecture. Please refer to [22] for more details. However, it is still difficult to increase further. A layered architecture or a networked architecture of modular networks is needed to increase the memory capacity. This is one of future challenges for Intelligence Dynamics.

5 MINDY: toward an open-ended system

In the previous experiments of RNNPB and RNN–SOM, we have to teach the basic or atomic behavioral patterns directly to the robot. The robot doesn't spontaneously learn all by itself. As we described in Sect. 3, one of the key issues in Intelligence Dynamics is to build an open-ended agent, which continues to learn spontaneously. Then, in the Sect. 3 we introduced the intrinsic motivations as shown in Fig. 3. In this section, inspired by Flow theory [9], we propose a computational model for an open-ended system named MINDY, which includes not only the intrinsic motivations but also the important features described in Sect. 3, such as the prediction and control function, and the planner and rehearse function. In this article, we provide a general summary of MINDY for the purpose of developing a model of Intelligence Dynamics, MINDY. Please refer to [31,30] for more details.

Flow theory insists that a human is highly motivated when he/she tries to learn a task of suitable difficulty for his/her capabilities and skills (Fig. 10). The balance between skills and difficulty for a challenging task is very important for machine learning. This balance can be evaluated by the progress of learning. For example, if the task is too easy, the robot can predict states correctly and achieve a goal easily. On the other hand, if the task is too difficult, the robot can not predict states correctly neither can it achieve the goal even if it tries many times. However, if the task is a suitably challenging one, at the beginning of the learning, the robot can not predict states correctly and cannot achieve the goal. However, after some attempts, the robot can often predict the states correctly and can achieve the goal eventually.

MINDY employs the concept of Flow theory. Namely, the system evaluates the progress of the learning by assessing the error of both prediction and control functions. In this section,

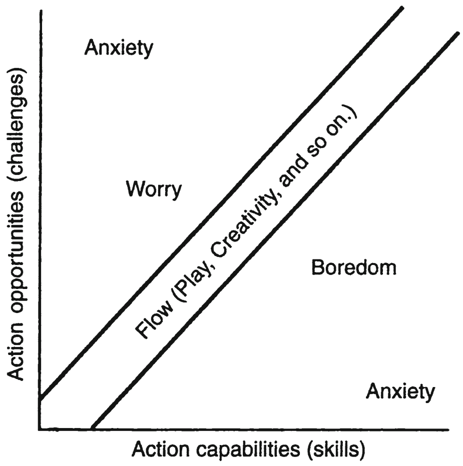

Anxiety
Worry
Flow (Play, Creativity, and so on.)
Boredom
Anxiety
Action opportunities (challenges)
Action capabilities (skills)

Fig. 10 A concept of Flow theory

we describe MINDY with a simple implementation example; a pendulum agent experiment. In this experiment, we implemented the predictor, the controller, and the planner using specific algorithms. However, it is easy to modify and extend the implementation based on the MINDY concept.

5.1 Pendulum agent simulation experiment

Figure 11 describes the appearance of a simulation environment of the pendulum agent. It has one joint that can be controlled by torque commands τ. The joint angle θ and its angle velocity ω are observable. The maximum value of the torque is limited so that it is necessary for the pendulum to accelerate by swinging several times to reach the top position. The energy variable is observable; it decreases as the torque command is applied and increases as the pendulum touches an energy charging station whose position can not be sensed. The lactic acid variable is also observable; it increases as the torque command is generated, and decreases as the torque command is not generated.

In order to continue actions, it is necessary for the pendulum to reach the charge station before the energy goes zero. However, at the beginning, the pendulum can not control itself to reach the charging station. Therefore it is necessary to learn to control its position to reach target positions, and then it can control itself so that it can touch the charge station when the energy variable becomes lower than the threshold.

Thus, selecting the target variable and its value, the pendulum system continues to learn as an open-ended system.

5.2 Prediction and control functions

The key question is what kind of learnable predictors and controllers should be used to build this open-ended system? Whatever learning algorithms are used for the purpose, in general, to learn the prediction function, it is necessary to neglect irrelevant variables, or to select only relevant variables. Otherwise the irrelevant variables will interfere with the learning of the prediction function. Therefore, if the system could identify the cause of the target variable, then it should be the only relevant variables selected as the inputs of the prediction function. However, in general it is difficult to analyze the cause and the effect relations for

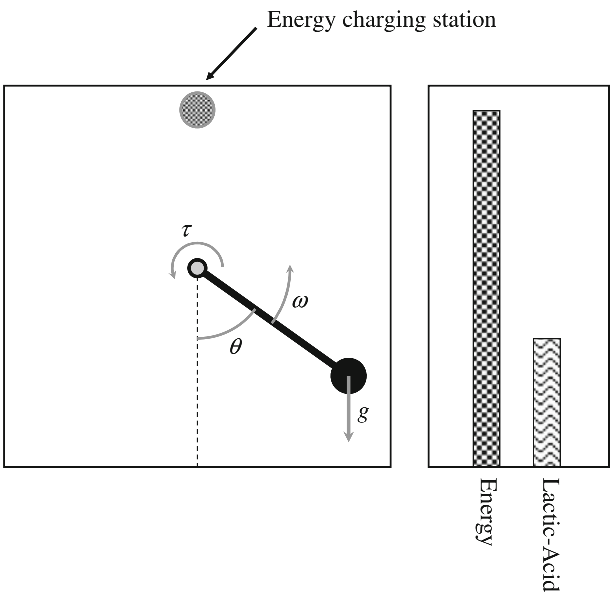


Fig. 11 Pendulum agent simulation setup

this purpose. Therefore, in this experiment, the system constructs a number of predictors for a selected target variable that has different inputs variables of possible combinations. While training each of the predictors, the system evaluates and selects the one that performs the best prediction result at the time. The inputs to the best predictors are assumed to be the causal variables of the selected target. From this analysis, the input to the corresponding controller is determined.

The controller has two kinds of inputs. One is the target variable with a target value and another is the causal variable with a current value. The controller is trained to generate an appropriate motor command given these two inputs: minima or maxima respectively.

Figure 12 shows the basic module with the predictor and the controller. We use accurate online support vector regression (AOSVR) [21] for the predictors and the controllers. AOSVR is an incremental learning algorithm for support vector regression (SVR). The learning algorithm of SVR is based on quadratic programming, which guarantees a global minima or maxima.

5.3 Planning function

Figure 13 shows the entire architecture of MINDY for the pendulum agent experiment. There is a planner on top of the network of the basic modules. The planner explores actions towards the target value of the target variable with predictors. We use a simple A* planner for the experiment, but other modern planners can be used as well. The planner searches the action sequence which can reach the target value. If it failed to find a perfect solution within the limited amount of time, it selects the plan that achieved the closest distance to the target. Then, the actions in the plan are applied as motor commands. The predictor and the controller are both trained using the motor and the sensor signals. Note that the final state is not the same as the target state used by the planner. In such a case the final state should be used as the

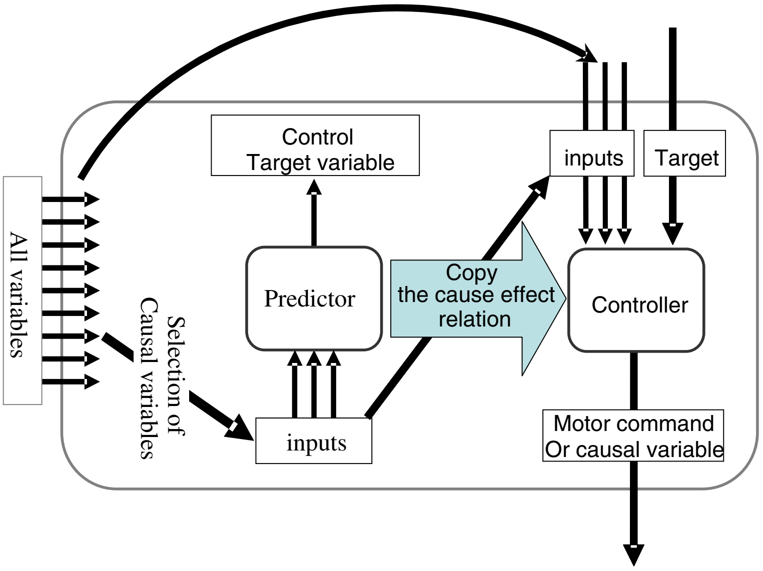

All variables
Selection of Causal variables
Control Target variable
Predictor
inputs
Copy the cause effect relation
inputs
Target
Controller
Motor command Or causal variable

Fig. 12 Basic module of MINDY with predictor and controller

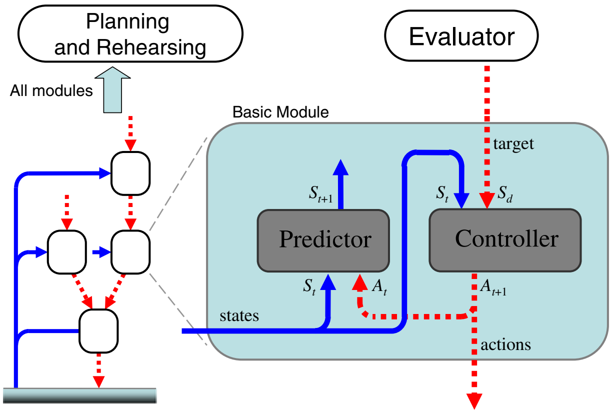


Fig. 13 Entire architecture of MINDY for the pendulum agent experiment

target state for the controller. The target variable and value selection, planning and training of the predictor and the controller are executed several times to evaluate the progress of the learning.

If the learning is in progress successfully, the system continues to learn the same target variable and value. If the task has been learnt and the learning progress stops, the system changes the target variable and value. However, if the task is too difficult and the learning doesn't progress, the system changes the target variable and value too.

5.4 Evaluation of the progress of learning

Figure 14 shows examples of the state (value) sequence of variables of the simulation. There are three sequences, which are a sequence to a target explored by the planner (X_{plan}),

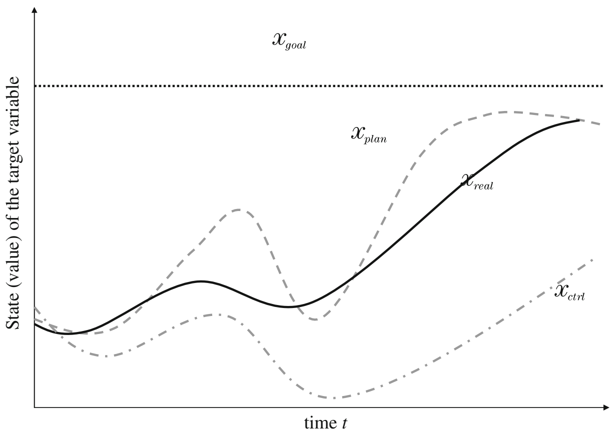


Fig. 14 State sequence of the pendulum agent simulation. Here X_{goal} is a goal state of a target variable, X_{plan} is a state sequence generated by the planner, X_{ctrl} is a state sequence generated by rehearsal with the predictor and the controller, and X_{real} is a state sequence generated by the real motor commands by the planner or the controller

rehearsed by the predictor and the controller (X_{cntl}), and observed by the real execution (X_{real}). Here, the rehearsal by the predictor and the controller means that to generate the motor command sequence and the prediction variables by the internal loop that is made up by connecting the output of the predictor to the input of the controller, and the output of the controller to the input of the predictor.

The system evaluates the progresses of the learning of the predictor, the controller, the planner, and, with the following functions.

- $E_{\mathrm{pred}} = \frac{\sum_t |X_{\mathrm{plan}}(t) - X_{\mathrm{real}}(t)|}{\Delta t}$
- $E_{\mathrm{cntl}} = \frac{\sum_t |X_{\mathrm{plan}}(t) - X_{\mathrm{cntl}}(t)|}{\Delta t}$
- $D_{\mathrm{plan}} = \min(|X_{\mathrm{real}}(t) - X_{\mathrm{goal}}|)$

Because X_{plan} and X_{real} use the same motor commands, E_{pred} represents the performance of the predictor. Similarly, because X_{plan} is explored by using only the predictor, E_{cntl} is a measure of the difference in the sensor domain between the motor commands selected by the planner and the one generated by the controller. Thus, E_{cntl} represents the performance of the controller, if we can assume that the predictor's performance is almost good. Finally, D_{plan} can be considered as representation of the achievement degree to the goal. The smaller the better for the three evaluation functions.

5.5 Intrinsic and extrinsic motivations

Basically the behavior of the pendulum is determined by the selection of the target variable. We use the softmax strategy for the selection. Namely, the system selects the target variable based on the probability defined by

$$P(i) = \frac{\exp(v(i)/\tau)}{\sum_k \exp(v(k)/\tau)}$$

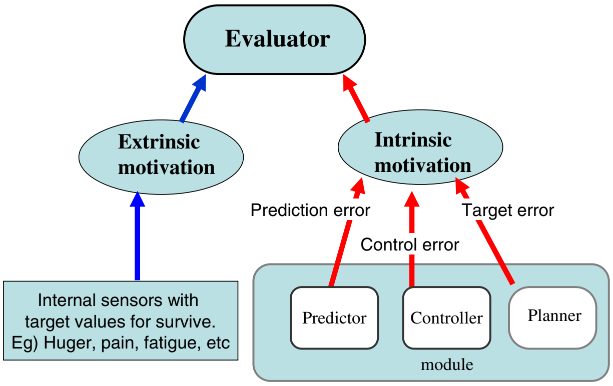


Fig. 15 Interpretation of errors and target values as the intrinsic motivation and the extrinsic motivation

where i is the index of the target variables, τ is a temperature, $v(i)$ is the priority of the ith variable. For example, we define the priority of the energy variable as $v(i) = 1 - Energy$, so that the smaller the Energy value the higher the selection probability. For the Lactic acid, we use $v(i) = Lactic\ acid$, so that the larger Lactic acid value the higher the selection priority. Energy and Lactic acid can therefore be considered to create the extrinsic motivations because the system tries to keep their value within a certain range for survival. For the other variables that give rise to the intrinsic motivations, we set $v(i) = 0.5$. This setting of $v(i)$s can realize the appropriate selection of the variables.

As shown in Fig. 15, both of the extrinsic and the intrinsic motivations are suitably combined to continue the development of the pendulum. It should be noted that the motivation for the learning of the predictor, the controller, and the planner can be considered as generating human-like motives for curiosity, manipulation and achievement respectively.

5.6 Summary of the algorithm

We summarize the algorithm into the following key steps.

1. Select a target variable and value according to the probability defined in the previous section. If the variable is for the extrinsic motives, the value is set as a defined value. If the variable is for the intrinsic motives, the value is set randomly.
2. Select the best predictor among the predictor modules. Set the input and target variables of the predictor as the inputs of the controller.
3. Make a plan towards the target value with the predictor. If the planner can not reach the target value, the action sequence that results in the closest to the target is selected, and the closest value is considered as the target value instead.
4. Execute the action sequence. Then, teach the controller with the actions and the observed sensor variables. Note that the final value of the execution should be used for the training of the controller as the real target value.
5. Evaluate the progress of the predictor, the planner, and the controller in this order. If the learning progress is proper, continue learning using the same target variable and value. Usually the learning of the predictor becomes in progress first. Then, the learning of the planner and the controller become in progress in this order.
6. If the learning doesn't progress, go to step 1. If it progresses, go to step 2.

5.7 Summary of the results

The pendulum simulation can be summarised as follows:

1. At the beginning of the learning, the joint angle and the angle velocity are often selected as the target variables. In the pendulum agent the dynamics of the joint angle and the angle velocity are nonlinear and complex. So, it is difficult for the system to learn the predictors and controllers. However, after some trials, the progress of the learning improves, and it often selects the angle and the angle velocity variable to learn based on the intrinsic motivations.
2. When Lactic acid increases, the system tries to decrease its value based on the corresponding extrinsic motivation. It is relatively easy to learn the dynamics of Lactic acid, the system can decrease the lactic acid when it is necessary from the beginning of the learning.
3. When Energy decreases, the system tries to increase its value based on the corresponding extrinsic motivation. However, in order to increase its value, it is necessary to swing up to the top position. This means that it is necessary to use the controller of the angle variable with the top position value. In the early phase of the learning, the causal estimation of Energy to angle is not well trained. Moreover, the angle controller is immature to reach the target angle value. Therefore, the learning is terminated with a small number of trials, because the learning doesn't progress suitably. After the angle controller becomes well trained to reach the target position, the progress of learning to control the energy variable using angle controller becomes better. Finally, the system can change for more energy when it is necessary.

Figure 16 shows the self-organized network structure of the basic modules. It exactly represents the relations used in the simulation model. The lactic acid variable is caused by only the torque and the lactic acid itself. The angle variable is caused by the torque, the angle velocity, and the angle itself. The angle velocity variable is cause by the torque, the angle, and the angle velocity itself. The energy variable is caused by the angle and the energy itself. Note that the energy is not directly caused by torque, but indirectly caused through the angle.

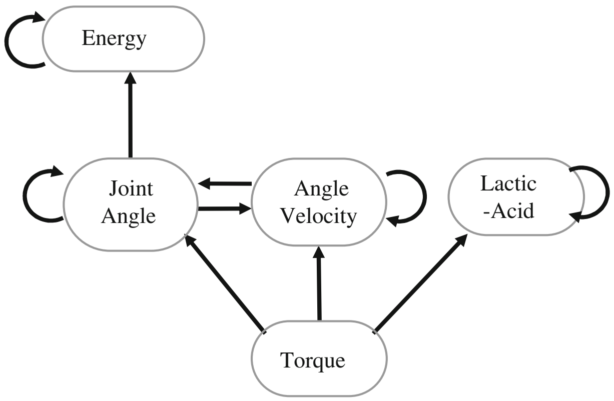


Fig. 16 Self-organized network of the basic modules

5.8 Discussion

There are some related works that emphasize the intrinsic motivation for developmental autonomous agents. Kaplan and Oudeyer [18] introduce curiosity drives for developmental robots, which insists that the progress of learning is important rather than decreasing the error measure. Our idea for the intrinsic motivation is also the progress of learning not only for the predictor but also for the controller and the planner. Barto et al. [3] proposes to integrate the idea with reinforcement learning to extend the computational model of reinforcement learning. They introduce an internal environment that gives internal reward, which can be considered as intrinsic motivation. They have not pointed out or used the importance of the progress of learning.

One of our key ideas to realize the open-ended system described in this section using the intrinsic motivation for learning based on Flow theory. However, in order to realize an open-ended system, we have to solve the variable selection problem or the cause-effect problem for machine learning. Because we conducted a small experiment using a single joint pendulum agent, we were able to solve the selection problem by making a large number of predictors with a limited number of combinations of the inputs. However, for a task with a larger number of variables, this approach becomes unrealistic and robust heuristics need to be developed. In fact we may need alternative approaches to solve such problems.

Regarding the self-organized network of modules, our approach is novel and highly effective for open-ended system. However, we considered only observable variables in this experiment. It is important to extend the idea to hidden variables. The simple example of a hidden variable is a variable for context information; if there are two different situations which generate the same observation vectors, the hidden variable plays an important role distinguishing the two situations. The hidden variable usually represents the history of the sensors, or in general the context. This hidden variable has to be generated in a self-organized manner. Handling self-organization with the hidden variables is one of the future challenges.

6 Summary and conclusion

In this paper we introduced and described Intelligence Dynamics as a new approach to machine intelligence. Intelligence Dynamics has three key elements embodiment, dynamical system properties and a constructive modeling approach.

We provided the entire architecture for an open-ended autonomous agent, which is able to develop and learn all by itself. We illustrated the architecture using our state-of-the-art robot QRIO and a simulation. We reported two examples of our implementations based on the entire architecture which provides a proof of concept and also highlights the next steps that need to be taken in order to design and build truly intelligent agents that can develop and learn all by themselves.

The first example examined the dynamical systems approach using RNNPB where the dynamics attractors play an important role in developing memory of behaviors, and the intra-inter-dynamics of the attractors give rise to novel and interesting characteristics such as smooth transitions from one behavior to another. The second example demonstrated how to build an open-ended system that continues learning throughout its development. Inspired by Flow theory we defined intrinsic motivations for learning as the predictor, the controller, and the target setting. In order to realize cognitive development, we proposed a novel layered architecture that is self-organized using a causal-effect analysis to guide the learning progress.

As we discussed in each section, there are many outstanding problems that need to be addressed which we highlighted using the Intelligence Dynamics approach. We argue that the Intelligence Dynamics approach is a new and powerful way to take machine intelligence to a new level. There are many related multi-disciplinary approaches in biology, psychology and philosophy and computer science which are beyond the scope of this article. Even after 50 years of AI research, we need to continue seeking novel holistic approaches that may take new strides in studying intelligence and build truly intelligent autonomous agents.

Acknowledgements The author would like thank the former members of Sony Intelligence Dynamics Inc. (SIDL) for their research efforts on the concept and experiments described here. Especially the author would like to thank Dr. Toshi. T. Doi, the former president of SIDL, for directing Intelligence Dynamics research, and to Hideki Shimomura, Kohtaro Sabe, and Masato Ito for their discussions on this article. The author would also like to thank Akira Iga, the former president of Information Technologies laboratories, Sony, for his assist to continue research on Intelligence Dynamics.

References

1. Allen, G. I., & Tsukahara, N. (1974). Cerebrocerebellar communication system. *Physical Review, 54*, 957–1006.
2. Asada, M., MacDorman, K. F., Ishiguro, H., & Kuniyoshi, Y. (2001). Cognitive developmental robotics as a new paradigm for designing humanoid robots. *Robotics and Autonomous Systems, 37*, 185–193. doi:10.1016/S0921-8890(01)00157-9.
3. Barto, A. G., Singh, S., & Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collection of skills. In *Proceedings of the 3rd international conference on developmental learning (ICDL)*, San Diego, CA (pp. 112–119).
4. Bentivegna, D. C., Atkeson, C. G., & Cheng, G. (2004). Learning tasks from observation and practice. *Robotics and Autonomous Systems, 47*(2–3), 163–169.
5. Bentivegna, D. C., Ude, A., Atkeson, C. G., & Cheng, G. (2002). Humanoid robot learning and game playing using PC-based vision. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, Lausanne, Switzerland (pp. 2449–2454).
6. Bernstein, N. (1967). *The coordination and regulation of movements*. Oxford: Pergamon Press.
7. Borg, I., & Groenen, P. (1997). *Modern multidimensional scaling. Theory and applications*. New York: Springer.
8. Charniak, E., & McDermott, D. (1985). *Introduction to artificial intelligence*. Reading, MA: Addison Wesley.
9. Csikszentmihalyi, M. (1990). *Flow: The psychology of optimal experience.* New York: Harper and Row.
10. Donald, M. (1991). *Origin of the modern mind*. Cambridge, MA: Harvard University Press.
11. Doya, K., Samejima, K., Katagiri, K., & Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Computation, 14*(6), 1347–1369. doi:10.1162/089976602753712972.
12. Fujita, M., Kuroki, Y., Ishida, T., & Doi, T. T. (2003). Autonomous behavior control architecture of entertainment humanoid robot SDR-4X. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, Las Vegas, NV (pp. 960–967).
13. Harnad, S. (1990). The symbol grounding problem. *Physica D. Nonlinear Phenomena, 42*, 335–346. doi:10.1016/0167-2789(90)90087-6.
14. Haruno, M., Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Computation, 13*, 2201–2220. doi:10.1162/089976601750541778.
15. Inamura, T., Nakamura, F., & Toshima, I. (2004). Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research, 23*(4), 363–377. doi:10.1177/0278364904042199.
16. Ito, M., Noda, K., Hoshino, Y., & Tani, J. (2006). Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks, 19*(3), 323–337.

17. Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science, 16*, 307–354.
18. Kaplan, F., & Oudeyer, P.-Y. (2003). Motivational principles for visual know-how development. In *Proceedings of the 3rd international workshop on epigenetic robotics*, Edinburgh, Scotland (pp. 73–80).
19. Kohonen, T. (1997). *Self-organizing maps*. New York: Springer-Verlag.
20. Kuniyoshi, Y., Ohmura, Y., Terada, K., Nagakubo, A., Eitoku, S., & Yamamoto, T. (2004). Embodied basis of invariant features in execution and perception of whole body dynamic actions—knacks and focuses of roll-and-rise motion. *Robotics and Autonomous Systems, 48*(4), 189–201. doi:10.1016/j.robot.2004.07.004.
21. Ma, J., Theiler, J., & Perkins, S. (2003). Accurate on-line support vector regression. *Neural Computation, 15*(11), 2683–2703. doi:10.1162/089976603322385117.
22. Minamino, K. (2008). Intelligence model organized by rich experience (in Japanese). In *Intelligence dynamics* (Vol. 3). Japan: Springer.
23. Newell, A., & Simon, H. A. (1976). Computer science as empirical enquiry: Symbols and search. *Communications of the ACM, 19*(3), 113–126. doi:10.1145/360018.360022.
24. Noda, K., Ito, M., Hoshino, Y., & Tani, J. (2006). Dynamic generation and switching of object handling behaviors by a humanoid robot using a recurrent neural network model. In *Proceedings of simulation of adaptive behavior (SAB'06)*, Rome, Italy. Lecture Notes in Artificial Intelligence (Vol. 4095, pp. 185–196).
25. Pfeifer, R., & Scheier, C. (1999). *Understanding intelligence*. Cambridge, MA: MIT Press.
26. Reed, E. S. (1997). *From soul to mind: The emergence of psychology, from Erasmus Darwin to William James*. New Haven, CT: Yale University Press.
27. Rizzolattie, G., Fadiga, L., Gallese, V., & Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Brain Research. Cognitive Brain Research, 3*, 131–141. doi:10.1016/0926-6410(95)00038-0.
28. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing*. Cambridge, MA: MIT Press.
29. Russell, R., & Norvig, P. (2002). *Artificial intelligence: A modern approach*. Englewood Cliffs, NJ: Prentice Hall.
30. Sabe, K. (2005). A proposal of intelligence model: MINDY (in Japanese). In *Intelligence dynamics* (Vol. 2). Japan: Springer.
31. Sabe, K., Hidai, K., Kawamoto, K., & Suzuki, H. (2006). A proposal for intelligence model, MINDY for open ended learning system. In *Proceedings of the international workshop on intelligence dynamics at IEEE/RSJ Humanoids*, Geneva, Italy.
32. Scholkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization and beyond*. Cambridge, MA: MIT Press.
33. Sutton, R. S., & Bart, A. G. (1998). *Reinforcement learning*. Cambridge, MA: MIT Press.
34. Tani, J. (2001). Learning to generate articulated behavior through the bottom-up and the top-down interaction process. *Neural Networks, 16*(1), 11–23. doi:10.1016/S0893-6080(02)00214-9.
35. Vijayakumar, S., & Schaal, S. (2000). LWPR: An $O(n)$ algorithm for incremental real time learning in high dimensional space. In *Proceedings of the seventeenth international conference on machine learning (ICML2000)*, Stanford, CA (pp. 1079–1086).