Cognitive Learning for Practical Solution of the Frame Problem

Atsushi Ueno, Hideaki Takeda and Toyoaki Nishida

Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-0101, JAPAN
E-mail: ueno@is.aist-nara.ac.jp, Phone: +81-743-72-5263, Fax: +81-743-72-5269

Abstract. The main problem for agents in real environments is how to abstract useful information from a large amount of environmental data. This is called *the frame problem.* Learning how to perform abstraction is a key function in a practical solution to the frame problem. As such a learning system, we developed *Situation Transition Network System (STNS).* The system extracts *situations* and maintains them dynamically in a continuous state space on the basis of rewards from the environment. These situations can be regarded as empirically obtained symbols. In this way, the system learns how to perform abstraction in a dynamic environment. The results of computer simulations are given.

key words: cognitive learning, articulation, the frame problem, reinforcement learning

1 Introduction

The real world is a dynamic environment and has a huge amount of information. So autonomous systems in the real world are required both to react quickly and to process such a huge amount of information. There is a conflict between these two functions, so many problems occur, such as explosion of the processing time, explosion of the amount of the description, inconsistency between the real world and the inner model and so on. These problems are called *the frame problem.* In this paper, the frame problem is defined as "the problem of how to deal with partiality of information" [3].

The frame problem cannot be solved intrinsically because no agent can have all information in the world. But animals seem not to be troubled with the frame problem in the daily life. It can be considered as a practical solution of the frame problem. We can categorize this solution into the following three methods. The first method is "designing appropriate information frames in advance" that is adopted by industrial robots and vending machines. This method is very efficient in familiar environments. But only by this method, agents cannot select appropriate actions in environments that the designers did not take into account. The second is "making many feedback loops between the system and the environment ". This method is realized in behavior-based systems. Agents with this method can correct small mistakes before they fail into serious situation, so the agents can act in unfamiliar environments without fatal faults. But only by this method, the agents repeat the same mistakes in the same situations. The last is "obtaining appropriate information frames empirically". This method can be regarded as cognitive learning or learning how to perform abstraction. By this method, agents can gradually get acclimated to unfamiliar environments. In this paper, we focus on the last method.

2 Cognitive Learning

2.1 Cognitive Learning in Agents

Many intelligent agents are based on symbol processing because a symbol system can abstract an essence from a huge amount of information. Usually, learning in such an agent is executed on the fixed symbol system. In this case, information processing is efficient, but it is very difficult to design a symbol system appropriate for all environments in advance. So, in unfamiliar environments, such an agent is apt to fail to manage a huge amount of information and get into the frame problem.

On the other hand, agents can also learn the symbol system itself. This type of learning can be regarded as cognitive learning. By this learning, it is expected that an agent can obtain a necessary and sufficient symbol system for the current task and environment. So an agent with both cognitive learning and symbolic learning can have flexibility necessary for adapting to unfamiliar environments. This ability is essential to the practical solution of the frame problem. The demerits of this method are inefficiency and complexity of information processing, and it is difficult to decide appropriate basis of articulation.

Such cognitive agents are realized only in the domain of reinforcement learning. In the agent, reinforcement learning of behavior policy is executed in the symbol space, and learning of the state representation is executed in the continuous input space. The latter is a kind of cognitive learning. In reinforcement learning, the task is represented in terms of rewards. So, rewards can be used reasonably as the basis of articulation. In this respect, reinforcement learning is suitable for cooperating with cognitive learning. But reinforcement learning system can deal with symbols only for representing states or behaviors. So such systems can execute only very simple symbol processing.

2.2 Cognitive Learning for Practical Solution of the Frame Problem

The following two properties are important for cognitive learning to solve the frame problem practically. First, articulation should be based on rewards. By this property, highly abstracted state representation that is appropriate for the task is expected as mentioned above. Second, cognitive learning and behavior learning should be executed simultaneously while executing the task. In this case, good state recognition can make good behavior selection, and good behavior selections can maintain well-shaped states. Thus it can be expected that specialization to the task and flexibility to changes are realized together by interdependence between states and behaviors. This flexibility is especially important for adapting to unfamiliar environments.

3 Situation Representation

In this section, we explain a new state representation that realize the above two properties.

3.1 Articulation Based on Similarity of Rewards

By the words, "similarity of rewards", we consider two types of similarity. One means that "the similar reward will be gotten if the same behavior is executed". By this type of similarity, a state space is articulated into pathway-shape areas. The other means "the volume of discounted reward is similar". By this type of similarity, a state space is articulated into contour-shape areas.

The former similarity is adopted in the new representation that we developed. And we call the articulated areas *situations*. A situation can also be regarded as a highly abstracted state that has a specific meaning. The meaning of a situation is "the system get the specific result by the specific behavior". The specific behavior is called the *condition behavior* of the situation. The specific results are divided into two types. In a situation based on immediate rewards, *R-Situation*, the result is to get a specific big reward. In a situation based on situation transitions, *T-Situation*, the result is to transit to a specific situation. If every chain of T-Situations is anchored to an R-Situation, every situation is guaranteed to lead to a specific reward by the same series of behaviors.

3.2 Bitten Hyper-Ellipsoid Representation

In on-line learning, the system should be able to decide rough shapes of situations from a limited amount of data, and to decide finer shapes as data increase. For this purpose, we propose the *bitten hyper-ellipsoid* representation. In this representation, each situation is shaped by the positive instances and the negative instances that are decided based on the meaning of the situation.

As shown in Fig. 1, this representation is a mixture of macroscopic cognition and microscopic cognition. In macroscopic cognition, the boundary of a situation is a contour of Mahalanobis' distance from the population of the positive instances. This boundary forms a hyper-ellipsoid. This cognition is quick and rough that can make up for the lack of data. Microscopic cognition is realized by the Nearest Neighbor methods and grows finer as data increase. By mixing these two types of cognition, a fine and flexible cognition is realized.

4 Situation Transition Network System

In this section, we explain *Situation Transition Network System (STNS)* that we developed.

4.1 Outline of STNS

Fig. 2 shows the structure of STNS. This system consists of a situation classifier, a situation transition network (STN), and several behavior modules. In each behavior step, the system decides the current situation where the current input

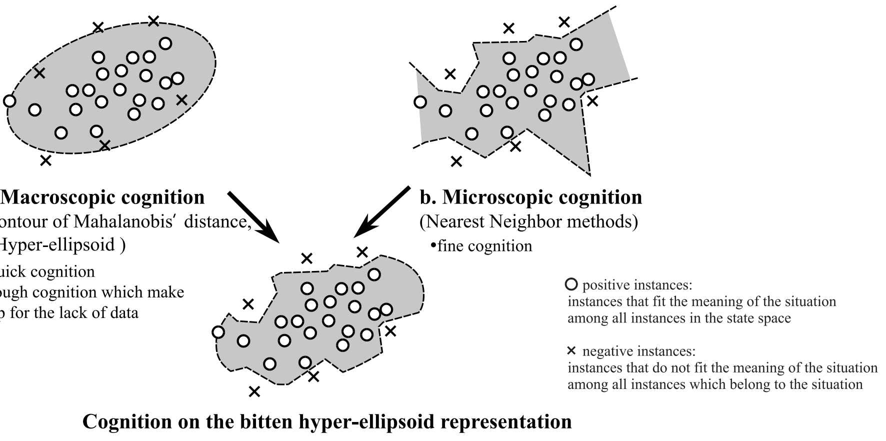

a. Macroscopic cognition
(Contour of Mahalanobis′ distance, Hyper-ellipsoid)
•quick cognition
•rough cognition which make up for the lack of data
b. Microscopic cognition
(Nearest Neighbor methods)
•fine cognition
○ positive instances:
instances that fit the meaning of the situation among all instances in the state space
× negative instances:
instances that do not fit the meaning of the situation among all instances which belong to the situation
Cognition on the bitten hyper-ellipsoid representation

Fig. 1. The bitten hyper-ellipsoid representation

is included, makes a partial plan on the STN, and activates a behavior module according to the plan. And a list of the input, the corresponding situation, the selected behavior, and the gained reward are put into a history database. Cognitive learning and behavior learning are executed on the data in the history database. The history database always has fixed numbers of data. When a new data comes in, the oldest data is eliminated.

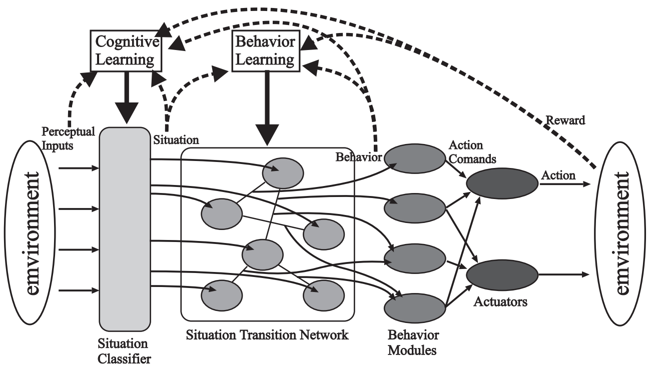


Fig. 2. The structure of STNS

In cognitive learning, the system extracts situations and maintains them on the basis of rewards from the environment. This is regarded as learning of situation representation. In behavior learning, the system learns the Markov Decision Problem (MDP) model of the environment on the learned situation representation. The next behavior is decided by the partial planning on the model. This process is regarded as a kind of reinforcement learning of behavior policy. These two learning processes are performed simultaneously while executing the task.

Next, we explain the cognitive learning in more detail. Space did not permit us to explain the details of the behavior learning.

4.2 Cognitive Learning in STNS

In STNS, the perceptual input space (state space) is divided situations as shown in Fig. 3 (a case of 2-dimensional input). Situation 1-7 are represented by bitten hyper-ellipsoids explained in the section 3. Situation 0 is the margin space. Situations are overlapped, and a new perceptual input is ascertained whether it belongs to each situation from the top of the discrimination tree in Fig. 3.

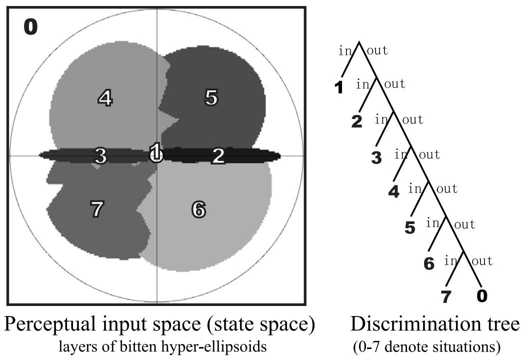

0
4
5
3
1
2
7
6
in
out
1
2
3
4
5
6
7
0
Perceptual input space (state space)
layers of bitten hyper-ellipsoids
Discrimination tree
(0-7 denote situations)

Fig. 3. The perceptual input space and the discrimination tree

Cognitive learning is realized by extraction and maintenance of situations. There are two conditions for situation extraction.

1. **R-Situation** In the whole state space, there are enough (more than N^R_{init}) data in which the system got a specific reward larger than a threshold (r^R_{min}) by a specific behavior.
   T-Situation In Situation 0, there are enough (more than N^T_{init}) data in which the system transited to a specific situation other than Situation 0 by a specific behavior and did not get a large (more than r^R_{min}) reward.
2. The given data are not positive instances of any situations.

When both of these conditions are fulfilled, a new situation is extracted. In extraction, the meaning of the new situation is defined referring to the condition 1, and positive instances are collected from the history database.

The extracted situations are maintained by these five methods.

– Renewal of the population of the positive instances and the population of the negative instances of each situation
– Renewal of the boundary of each hyper-ellipsoid (The boundary is settled just on the farthest positive instance.)
– Renewal of the order of situations in the discrimination tree (The nearer situation to rewards is put upper.)
– Changing the meaning of deformed situations
– Removal of deformed situations

5 Experiments

5.1 Experiment 1: Navigation on 2-Dimensional Input

To show the validity of STNS, we show a simple experiment on computer simulation. Fig. 4 shows the navigation problem on 2-dimensional input. Every trial starts after the goal is settled at an arbitrary position and the rover is set at an arbitrary position outside of the goal in an arbitrary direction. The trial ends when the rover arrives at the goal, and the next trial starts immediately. At the beginning of learning, there are only the goal area and Situation 0 (the margin space) in the state space.

Task:
To navigate a 16x12 rectangle rover from an arbitrary position to the only goal (a small circle whose radius is 5) in a 100x100 square continuous plane.
The goal is settled at an arbitrary position in the 72x72 square area at the center of the room.
There is no obstacle but the wall around the room.

Perceptual Input (2-dimensinal):
The real-valued (x, y) coordinates of the goal on the coordinate system fixed to the rover.

Rewards:
1. Arrival at the goal: +10
2. Collision with the wall: -1
3. Trying to rotate over 90 degrees: -1
(The rover can look any direction by at most 90 degrees rotation because of the symmetry of the behaviors.)

Behaviors:
forward movement, backward movement, clockwise rotation, counterclockwise rotation.
(The rover is assumed to be able to make collision-free rotation.)

Stopping Conditions of Behaviors:
1. Getting some reward.
2. Arrival at the target situation in the present plan.

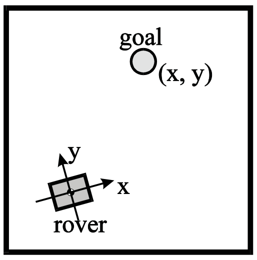


Fig. 4. The navigation problem on 2-dimensional input

Learning has converged after about 1200 behaviors are executed on average. Fig. 5 shows a typical state space after leaning has converged and the optimum state space. The small circle at the center of each space denotes states in which the rover arrives at the goal. As shown in this figure, good situation representation and good behavior policy were obtained.

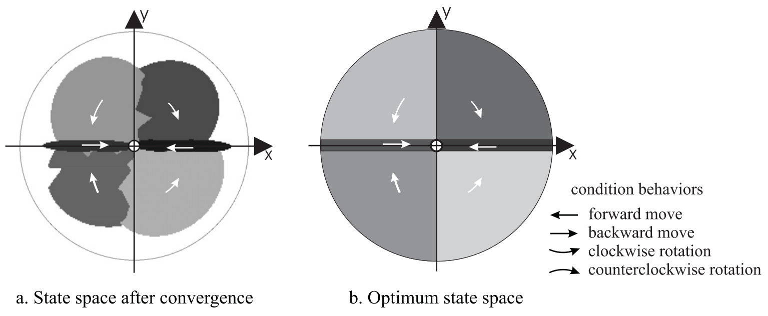
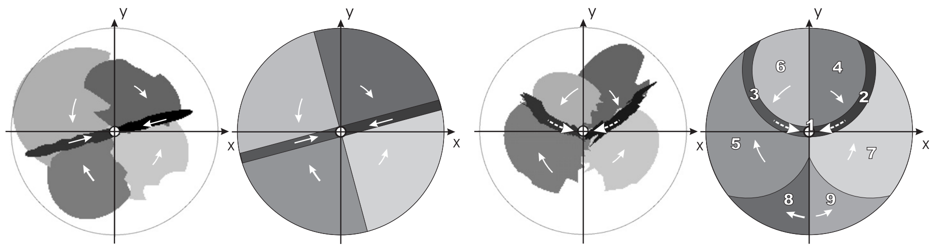


Fig. 5. A state space after convergence and the optimum state space

5.2 Experiment 2: Flexibility Test

To test the flexibility of STNS, we executed two experiments on computer simulation. One is a flexibility test to sensor trouble and the other is a flexibility test to actuator trouble. In both experiments, we prepared an STNS after 10000 behavior steps in experiment 1 (shown in Fig. 5), changed the environment, and then let the STNS continue learning in the new environment. As changes of environment, we rotated the direction of the goal sensor in the former test, and got the revolution rate of the left wheel lower in the latter test. Fig. 6 shows a typical result in the case of 15 degrees rotation of the goal sensor and a typical result in the case of 15% lower speed of the left wheel. As shown in this figure, passably good situation representation and good behavior policy were obtained.



State space after convergence Optimum state space State space after convergence Optimum state space

a. Sensor Trouble (15° rotation of the goal sensor) b. Actuator Trouble (15% lower speed of the left wheel)

Fig. 6. Results of the flexibility test

In both experiments, STNS can flexibly adapt to small changes while keeping the high performance by transforming the shapes of situations, and STNS can stably adapt to big changes by eliminating deformed and obstructive situations and starting again from the blank state space. These two types of adaptation are switched to each other autonomously. Even in the case of big changes, learning converged after a delay of at most 1000 behavior steps compared with learning from the blank state space. So it can be said that learning of STNS is stable.

6 Related Works

There are many systems that can learn the state representation for reinforcement learning. Some of them are based on the similarly of rewards. Chapman and Kaelbling [2] proposed the G algorithm that splits the state space statistically on the basis of rewards. Mahadevan and Connell [5] proposed an algorithm that extracts clusters of states in the state space statistically on the basis of rewards. Both methods deal with states each of which consists of a bit sequence. Therefore, their systems are different from STNS that deals with a problem with a continuous state space. Asada et al. [1] proposed a method that divides a continuous state space by hyper-ellipsoids. The method executes cognitive learning in a off-line way: the system first learns the whole state representation on the basis of experiences of random behaviors and then shifts to task execution. Ishiguro et al. [4] proposed a method that first divides a continuous state space by hyper-planes on the basis of experiences of random (or given) behaviors, and then learns a behavior policy by Q-learning. It can execute these two types of learning alternately. But it constructs fixed boundaries between states. It is different from STNS at this point because STNS constructs a flexible situation representation. Takahashi et al. [6] proposed a method that segments a continuous state space by the Nearest Neighbor methods while executing the task. The segmentation of the method is based on rewards in the areas near to rewards and based on the relationship between the sensory input and its gradient in the areas far from rewards. By the latter policy, the method can segment the state space stably even in the areas far from rewards. But the validity of this policy needs to be confirmed.

7 Conclusion

We have proposed STNS, a method for executing both cognitive learning and behavior learning simultaneously with task execution, and shown that the method is effective to obtain good state representation (symbols) and good behavior policy in continuous state space. The representation also have flexibility essential to the practical solution of the frame problem.

There need more work for the symbol processing system grounded on real environments. As mentioned in the section 2.1, current agents with cognitive learning can execute only very simple symbol processing. In order to expand this sort of symbol system, the following four functions are worth considering: structuring symbols, symbolizing objects, reusing symbols, and sharing symbols. Furthermore, in order to embed this sort of cognitive agent into the environment, parallel processing of information should be necessary.

References

1. Asada, M., Noda, S., and Hosoda, K. Action-based sensor space categorization for robot learning. *Proc. of IROS 96*, Vol. 3, pp. 1502–1509, 1996.
2. Chapman, D. and Kaelbling, L. P. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. *Proc. of IJCAI-91*, pp. 726–731, 1991.
3. Hasida, K. and Matsubara, H. Partiality of information and the structure of the frame problem. *Proc. of PRICAI '90*, pp. 711–716, 1990.
4. Ishiguro, H., Sato, R., and Ishida, T. Robot oriented state space construction. *Proc. of IROS 96*, Vol. 3, pp. 1496–1501, 1996.
5. Mahadevan, S. and Connell, J. Automatic programming of behavior-based robots using reinforcement learning. *Proc. of AAAI-'91*, pp. 768–773, 1991.
6. Takahashi, Y., Asada, M., and Hosoda, K. Reasonable performance in less learning time by real robot based on incremental state space segmentation. *Proc. of IROS 96*, Vol. 3, pp. 1518–1524, 1996.