Humanoid Robot Control Based on Reinforcement Learning

Shingo Iida[*] *Kiyotake Kuwayama*[*] *Masayoshi Kanoh*[**]
Shohei Kato[*] *Tsutomu Kunitachi*[***] *Hidenori Itoh*[*]

[*]Dept. of Intelligence and Computer Science, Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan
Email: {iida,kuwayama,shohey,itoh}@ics.nitech.ac.jp
[**]School of Life System Science and Technology, Chukyo University
101 Tokodachi, Kaizu-cho, Toyota 470-0393, Japan
Email: mkanoh@life.chukyo-u.ac.jp
[***]School of Informatics, Daido Institute of Technology
10-3 Takiharu-cho, Minami-ku, Nagoya 457-8530, Japan

Abstract:
Many existing methods of reinforcement learning have treated tasks in a discrete low dimensional state space. However, controlling humanoid robots smooth requires a continuous high-dimensional state space. In this paper, to treat the state space, we proposed an adaptive allocation method of basis functions for reinforcement learning. Up to now, grid or incremental allocation method have been proposed for allocation of basis functions. However, these methods may cause the curse of dimensionality, and fall into local minima. On the other hand, our method avoids local minima which are assessed by the trace of activity of basis functions. That is, if current state is judged to fall into a local minimum, our method eliminates a basis function which affects the state most. Moreover our method learns with a low number of basis functions because of the elimination process. To confirm the effectiveness of our method, we used a maze task to compare our method with an existing method, which has only an allocation process. Moreover, as learning of continuous high-dimensional state spaces, our method was applied to motion control of a humanoid robot. We demonstrate that our method is capable of providing better performance than the existing method.

1 Introduction

Recently, robot's demands have been changing from the industrial machines to domestic them which live together human beings. Such robots are hoped for communicating and interacting with human beings, and they require body and interface to do them. Humanoid robots which have multi-degree-of-freedom have been appeared, because they can work in our daily lives by their physical mechanism which is similar to human beings'. However, it is very difficult to control intricately robots with a controller made by human beings. The controller should be obtained by themselves in a human-like way, not be made manually. Human beings learn any action by trial and error, or by emulating someone's actions. We therefore apply reinforcement learning for controlling humanoid robots because it resembles a human's learning process at the point with learning by trial and error.

Many existing methods in reinforcement learning of control tasks are to discretize state space using BOXES [1] or CMAC [2] to approximate a value function which specifies what is good in the long run. However these methods are not to do generalization, and cause perceptual aliasing. Other methods use basis functions network for treating continuous state space and actions.

Networks with sigmoid functions have the problem of catastrophic interference. They are suitable for off-line learning, but not adequate to on-line learning such as learning motion [3, 4]. On the other hand, networks with radial basis functions are suitable for on-line learning, however learning with them requires a numerous number of units, because they can be not to do generalization enough. The normalized Gaussian functions have both feature of sigmoid functions and radial basis functions. Networks with the normalized Gaussian functions can therefore assess state space locally and globally. We use this type of networks to learning motion of humanoid robots.

Most of approaches using basis functions network prepare and allocate basis functions in advance in state space. If dimensionality increases, these approaches easily break down. To overcome this problem, improvement method have been proposed by Morimoto and Doya [5, 4]. Their method allocates basis functions if required, and is named Adaptive Gaussian Softmax Basis Function Network (A-GSBFN). However, A-GSBFN has a crucial issue that is possibility to fall into local minima, because it performs only allocation of basis functions. In this paper, we propose a method for adaptive allocation, Allocation/Elimination Gaussian Softmax Basis Function Network (AE-GSBFN). Our method avoids to fall into local minima by allocating or eliminating basis functions if needed. Moreover our

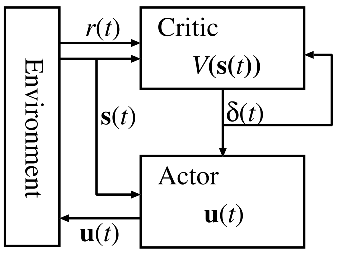

Figure 1. Actor-critic architecture.

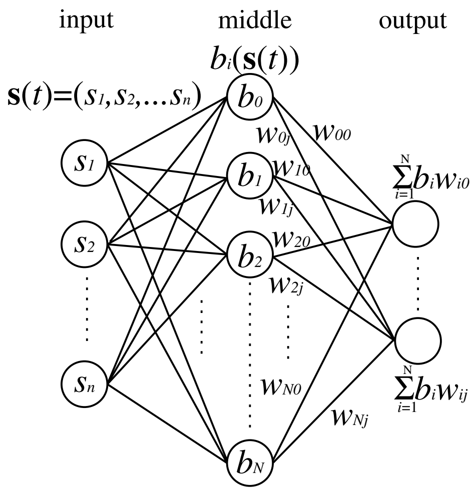

Figure 2. Basis function network.

method learns with a low number of basis functions because of the elimination process.

In order to confirm whether this method can avoid a fall into local minima and overcome the curse of dimensionality, we do two experiments: solving a maze and controlling a humanoid robot.

2 Actor-Critic Method

In this section, we describe an actor-critic method using basis functions. We apply this type of method to our method.

Actor-critic methods are TD methods that have a separate memory structure to explicitly represent the policy independent of the value function [6]. Actor-critic methods are constructed by an actor and a critic, as suggested by Figure 1. The policy structure is known as the actor, because it is used to select actions, and the estimated value function is known as the critic, because it criticizes the actions made by the actor.

The actor and the critic each have a basis function network for learning of continuous state spaces. Basis function networks have a three-layer structure as shown in Figure 2, and basis functions are placed in middle-layer units. Repeating the following procedure, in an actor-critic method using basis function networks, the critic correctly estimates the value function $V(\boldsymbol{s})$, and then the actor obtains actions which maximize $V(\boldsymbol{s})$.

1. When state $\boldsymbol{s}(t)$ is observed in the environment, the actor calculates the j-th value $u_j(t)$ of the action $\boldsymbol{u}(t)$ as follows[7]:

$$u_j(t) = u_j^{\max} g\left(\sum_i^N \omega_{ij} b_i(\boldsymbol{s}(t)) + n_j(t)\right), \tag{1}$$

where $u_j^{\max}$ is a maximal control value, N is the number of basis functions, $b_i(\boldsymbol{s}(t))$ is a basis function, ω_{ij} is a weight, $n_j(t)$ is a noise function, and $g()$ is a logistic sigmoid activation function whose outputs lie in the range $(-1,1)$. The Output value of actions is saturated into $u_j^{\max}$ by $g()$.

2. The critic receives the reward $r(t)$, and then observes the resulting next state $\boldsymbol{s}(t+\Delta t)$. The critic provides the TD-error $\delta(t)$ as follows:

$$\delta(t) = r(t) + \gamma V(\boldsymbol{s}(t+\Delta t)) - V(\boldsymbol{s}(t)), \tag{2}$$

where γ is a discount factor, and $V(\boldsymbol{s})$ is an estimated value function. $V(\boldsymbol{s}(t))$ is calculated as follows:

$$V(\boldsymbol{s}(t)) = \sum_i^N v_i b_i(\boldsymbol{s}(t)), \tag{3}$$

where v_i is a weight.

3. The actor updates weight ω_{ij} using TD-error:

$$\omega_{ij} \leftarrow \omega_{ij} + \beta\delta(t) n_j(t) b_i(\boldsymbol{s}(t)), \tag{4}$$

where β is a learning rate.

4. The critic updates weight v_i:

$$v_i \leftarrow v_i + \alpha\delta(t) e_i, \tag{5}$$

where α is a learning rate and e_i is eligibility trace. e_i is calculated as follows:

$$e_i \leftarrow \gamma\lambda e_i + b_i(\boldsymbol{s}(t)), \tag{6}$$

where λ is a trace-decay parameter.

5. Time is updated.

$$t \leftarrow t + \Delta t. \tag{7}$$

Note that Δt is 1 in general, but we used the description of Δt for interval of the control of humanoid robots.

3 Dynamic Allocation of Basis Functions

In this paper, we propose a dynamic allocation method of basis functions. This method is an extended application of the Adaptive Gaussian Softmax Basis Function Network (A-GSBFN) [5, 4]. A-GSBFN only allocates basis functions, on the other hand, our method allocates and eliminates them. In this section, we first mention A-GSBFN in Section 3.1, and then propose our method, Allocation/Elimination Gaussian Softmax Basis Function Network (AE-GSBFN), in Section 3.2.

3.1 A-GSBFN

Gaussian softmax basis function is used in A-GSBFN, and it is given by the following equation:

$$b_i(\boldsymbol{s}(t)) = \frac{a_i(\boldsymbol{s}(t))}{\sum_{k=1}^{N} a_k(\boldsymbol{s}(t))}, \tag{8}$$

where $a_i(\boldsymbol{s}(t))$ is a radial basis function, and N is the number of radial basis functions. Radial basis function $a_i(\boldsymbol{s}(t))$ in the i-th unit is calculated by the following equation:

$$a_i(\boldsymbol{s}(t)) = \exp(-\frac{1}{2}\|M(\boldsymbol{s}(t) - \boldsymbol{c}_i)\|^2), \tag{9}$$

where $\boldsymbol{c}_i$ is the center of i-th basis function, and M is a matrix that determines the shape of the basis function.

In A-GSBFN, a new unit is allocated if the error is larger than a threshold $\delta_{\max}$ and the activation of all existing units is smaller than a threshold $a_{\min}$:

$$|h(t)| > \delta_{\max} \quad \text{and} \quad \max_i a_i(\boldsymbol{s}(t)) < a_{\min}, \tag{10}$$

where $h(t)$ is defined as $h(t) = \delta(t)n_j(t)$ at the actor, and $h(t) = \delta(t)$ at the critic. The new unit is initialized with $\boldsymbol{c}_i = \boldsymbol{s}$, $\omega_i = 0$.

3.2 Allocation/Elimination GSBFN

In order to perform allocation and elimination of basis functions, we introduce three criteria into A-GSBFN: trace ε_i of activation of radial basis functions [8], additional control time η, and existing time τ_i of radial basis functions.

The criteria ε_i and τ_i are prepared for all basis functions, and η is prepared for both networks of the actor and the critic. A learning agent can gather further information on its own states by using these criteria.

We now define the condition of allocation of basis functions.

When a basis function is eliminated, the learning agent can avoid allocating another basis function immediately at near the state of the eliminated function by using the additional control time η. We define the condition of allocation considering η as follows.

Definition 3.1 Allocation
A new unit is allocated at $\boldsymbol{c} = \boldsymbol{s}(t)$ if the following condition is satisfied at the actor or critic networks:

$$|h(t)| > \delta_{\max} \quad \text{and} \quad \max_i a_i(\boldsymbol{s}(t)) < a_{\min} \quad \text{and} \quad \eta > T_{\text{add}}, \tag{11}$$

where T_{add} is a threshold. □

We then define the condition of elimination using ε_i and τ_i.

The trace ε_i of the activation of radial basis functions is updated in each step in the following manner:

$$\varepsilon_i \leftarrow \kappa\varepsilon_i + a_i(\boldsymbol{s}(t)), \tag{12}$$

where κ is a discount rate. Using ε_i, the learning agent can sense states which are recently taken by itself. The value of ε_i takes a high value if the agent stays in almost the same state. This situation is considered when the learning fell into a local minimum. Using the value of ε_i, we consider to how avoid the local minimum. Moreover, using τ_i, we consider to how inhibit a basis function from immediate elimination after it is allocated. We therefore define the condition of elimination using ε_i and τ_i as follows.

Definition 3.2 Elimination
The basis function $b_i(\boldsymbol{s}(t))$ is eliminated if the following condition is satisfied in the actor or critic networks.

$$\varepsilon_i > \varepsilon_{\max} \quad \textit{and} \quad \tau_i > T_{\text{erase}}, \tag{13}$$

where $\varepsilon_{\max}$ and T_{erase} are thresholds. □

Note that η is initialized at 0, when a basis function is eliminated.

4 Experiments

In order to confirm the effectiveness of AE-GSBFN, we did two experiments: solving a maze and controlling a humanoid robot. We first compared AE-GSBFN with A-GSBFN in the maze experiment. We checked performance of the avoidance of local minima in the experiment. We then did the experiment on controlling a humanoid robot to confirm whether AE-GSBFN can treat high-dimensional state space.

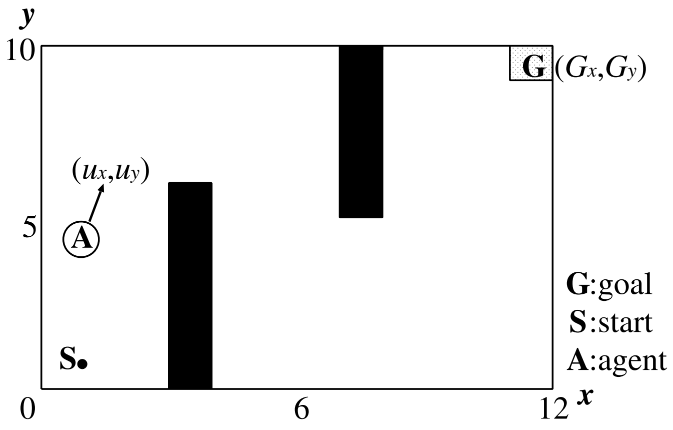

Figure 3. Maze task.

4.1 Solving the Maze

Consider the continuous two-dimensional maze shown in Figure 3. This environment has start state $\boldsymbol{S}$ and goal state $\boldsymbol{G}$. In this environment, the learning agent can sense $\boldsymbol{G}$, and takes action $\boldsymbol{u}(t) = (u_x, u_y)$ to reach it. Rewards $r(t)$ are determined by goal state $\boldsymbol{G} = (G_x, G_y)$ and the current state $\boldsymbol{s}(t) = (s_x, s_y)$, as follows:

$$r(t) = \sqrt{(G_x - s_x)^2 + (G_y - s_y)^2}. \tag{14}$$

One trial terminates when the agent reached $\boldsymbol{G}$ or exceeded 600 steps. We used $u_j^{\max} = 0.5$, $\gamma = 0.9$, $\beta = 0.1$, $\alpha = 0.02$, $\lambda = 0.6$ and $\Delta t = 1$ [step] for parameters in Section 2, $M = \mathrm{diag}(4.0, 4.0)$, $\delta_{\max} = 0.5$ and $a_{\min} = 0.4$ in Section 3.1, and $T_{\mathrm{add}} = 100$ [step], $\kappa = 0.9$, $\varepsilon_{\max} = 5.0$ and $T_{\mathrm{erase}} = 100$ [step] in Section 3.2.

Figure 4 shows the reward per trial with AE-GSBFN and A-GSBFN, and it is desirable that the value is large. The solid line in the figure represents the trial run with AE-GSBFN and the dotted line is the trial run with A-GSBFN. The results shown in the figure are averaged over 20 repetitions of the experiment. This results indicate that AE-GSBFN obtains more rewards than A-GSBFN. We looked at performances of both method in detail, using the number of steps required to travel from $\boldsymbol{S}$ to $\boldsymbol{G}$. The curves in Figure 5 represent the number of steps taken by the agent in each trial. Results of the figure are also averaged over 20 repetitions of the experiment. This results indicate that AE-GSBFN traveled from $\boldsymbol{S}$ to $\boldsymbol{G}$ faster than A-GSBFN.

Figures 6 and 7 plot basis functions allocations in successful experiments. In successful experiments, the allocation of basis functions with AE-GSBFN differs little from A-GSBFN. Statistics however indicated that AE-GSBFN achieved $\boldsymbol{G}$ 18 times for 20 repetitions, but A-GSBFN achieved only 9 times. It can be seen that AE-GSBFN

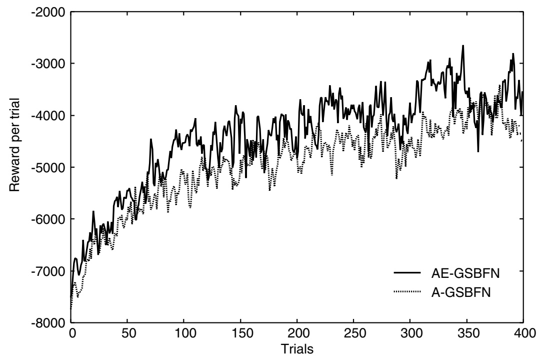

Figure 4. Rewards per trial.

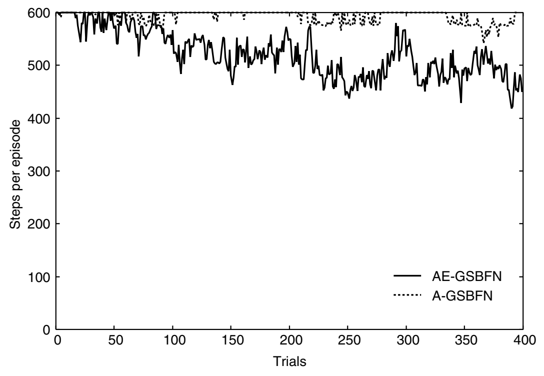

Figure 5. Average learning curves for AE-GSBFN and A-GSBFN.

performs better than A-GSBFN, and we consider that AE-GSBFN avoided a fall into local minima through the above experiments.

4.2 Controlling Humanoid Robot

In this section, as learning of continuous high-dimensional state spaces, AE-GSBFN is applied to a humanoid robot learning to stand up from a chair (Figure 8). The learning was simulated using the virtual body of the humanoid robot HOAP1 made by Fujitsu Automation Ltd. Figure 9 shows HOAP1. The robot is 48 centimeters tall, weighs 6 kilograms, has 20 DOFs, and has 4 pressure sensors each on the soles of its feet. Both of sensors of angular rate and acceleration are mounted in its breast. To simulate learning, we used the Open Dynamics Engine [9].

The robot can observe the following vector $\boldsymbol{s}(t)$ as its own state:

$$\boldsymbol{s}(t) \;=\; (\theta_W, \dot{\theta}_W, \theta_K, \dot{\theta}_K, \theta_A, \dot{\theta}_A, \theta_P, \dot{\theta}_P), \tag{15}$$

where θ_W, θ_K and θ_A are waist, knee, and ankle angles respectively, and θ_P is the pitch of its body (see Figure 8). Action $\boldsymbol{u}_j(t)$ of the robot is determined as follows:

$$\boldsymbol{u}_j(t) = (\dot{\theta}_W, \dot{\theta}_K, \dot{\theta}_A), \tag{16}$$

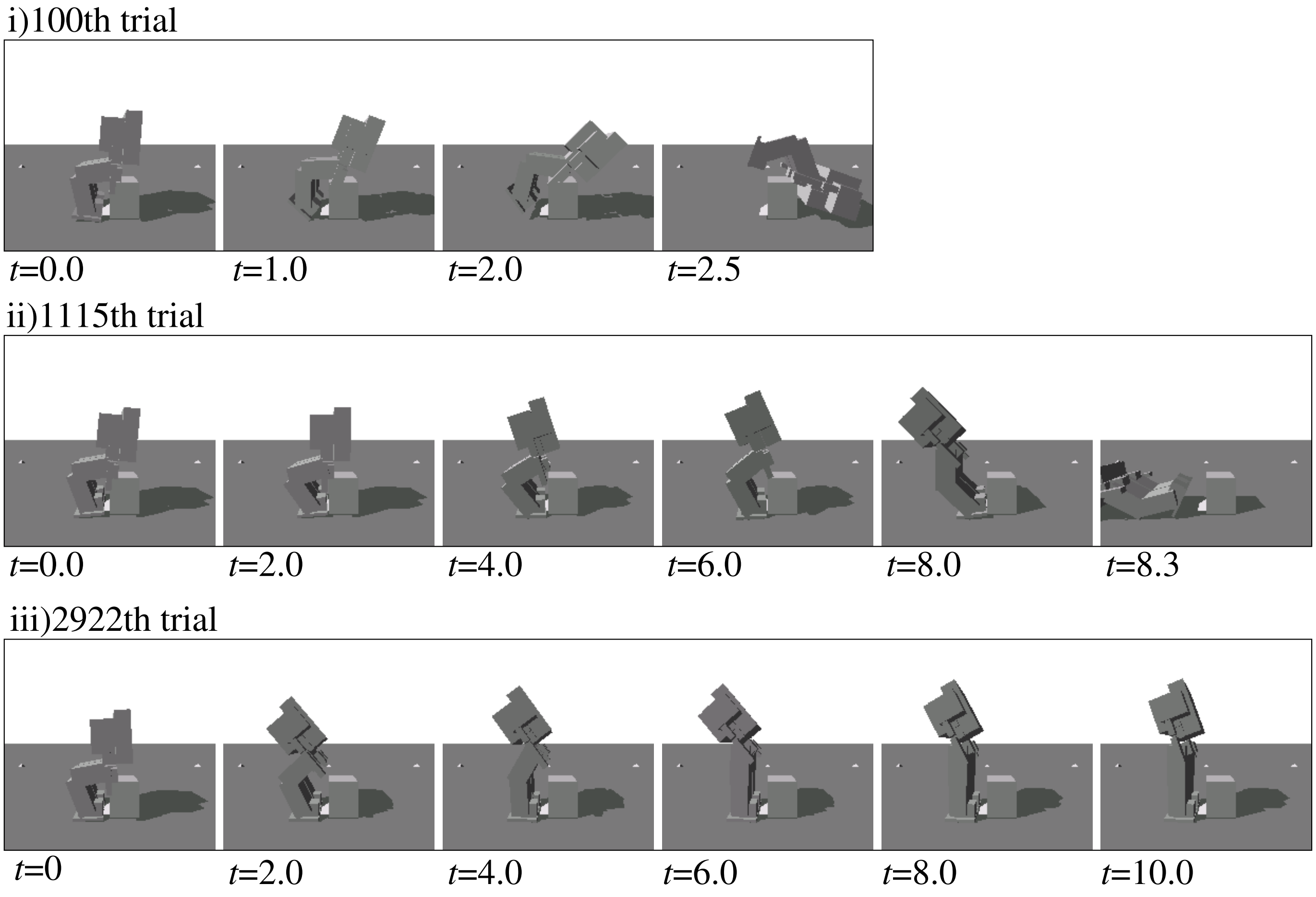

Figure 10. Learning results.

One trial terminates when the robot fell down or time passed over $t_{\text{total}} = 10$ [s]. Rewards $r(t)$ are determined by height y [cm] of the robot's breast:

$$r(t) = \begin{cases} 20 \times |\dfrac{l_{\text{stand}} - y}{l_{\text{stand}} - l_{\text{down}}}| & \text{(during trial)} \\ 20 \times |t_{\text{total}} - t| & \text{(on failure)} \end{cases}, \qquad (17)$$

where $l_{\text{stand}} = 35$ [cm] is the position of the robot's breast in an upright posture, $l_{\text{down}} = 20$ [cm] is its center in a falling-down posture. We used $u_j^{\max} = \frac{1}{36}\pi$ [rad], $\gamma = 0.9$, $\beta = 0.1$, $\alpha = 0.02$, $\lambda = 0.6$ and $\Delta t = 0.01$ [s] for parameters in Section 2, $M = \text{diag}(2.0, 0.57, 2.0, 0.57, 2.0, 0.57, 2.0, 0.57)$, $\delta_{\max} = 0.5$ and $a_{\min} = 0.4$ in Section 3.1, and $T_{\text{add}} = 1$ [s], $\kappa = 0.9$, $\varepsilon_{\max} = 5.0$ and $T_{\text{erase}} = 3$ [s] in Section 3.2.

Figure 10 shows learning results. First, the robot learned to fall down backward, as shown by i). Second, the robot intended to stand up from a chair, but fell forward as shown by ii), because it could not yet fully control its balance. Finally, the robot stood up while maintaining its balance, as shown by iii). The number of basis functions in the 2922th trial were 72 in both actor and critic networks.

We compared the number of basis functions in AE-GSBFN with the number of basis functions in A-GSBFN. Figure 11 shows the number of basis functions of the actor, averaged over 20 repetitions. In these experiments, learning with both AE-GSBFN and A-GSBFN succeeded in the motion learning, but the figure indicates that the number of basis functions required by AE-GSBFN is fewer than A-GSBFN. That is, high dimensional learning may be done using AE-GSBFN.

5 Conclusion

In this paper, we proposed a dynamic allocation method of basis functions, AE-GSBFN, in reinforcement learning. Through the allocation and elimination processes, AE-GSBFN overcomes the curse of dimensionality and avoids a fall into local minima. To confirm the effectiveness of

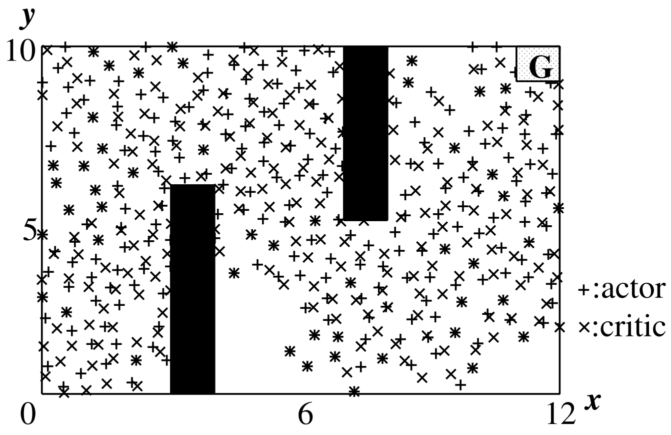

Figure 6. Allocation of basis functions with AE-GSBFN (actor:243, critic:247). This is a result of successful experiment.

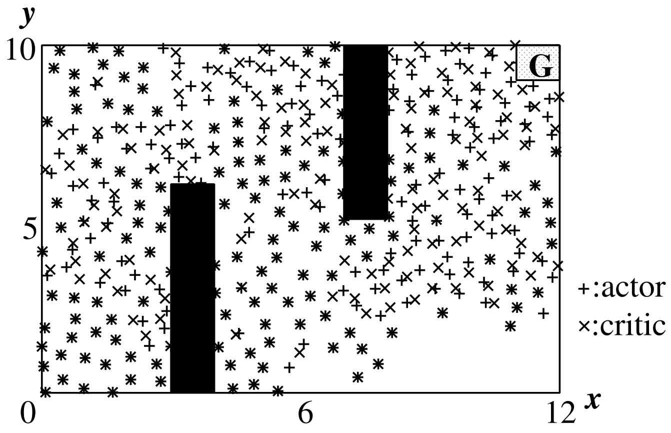

Figure 7. Allocation of basis functions with A-GSBFN (actor:270, critic:278). This is a result of successful experiment.

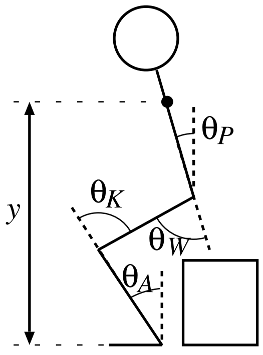

Figure 8. Learning motion.

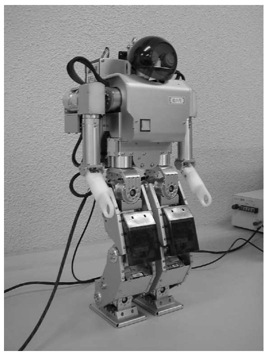

Figure 9. HOAP1: Humanoid for Open Architecture Platform.

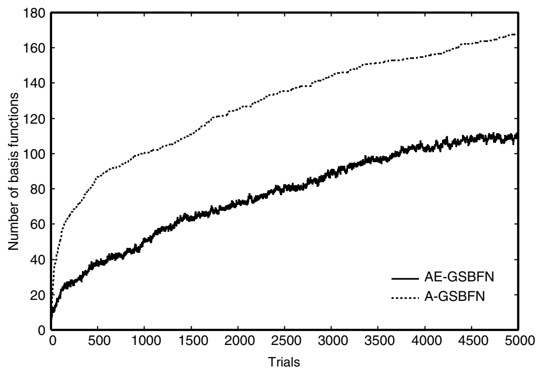

Figure 11. Number of the basis functions at the actor network (averaged over 20 repetitions).

AE-GSBFN, it was applied to a maze task and to the motion control of a humanoid robot. We demonstrated that AE-GSBFN is capable of providing better performance than A-GSBFN, and succeeded in enabling the learning of motion control of the robot.

References

[1] A.G.Barto et al. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Sys., Man, & Cyb.*, 13:834–846, 1983.

[2] J.S.Albus. *Brain, Behavior, and Robotics.* Byte Books, 1981.

[3] S Schaal and C. G. Atkeson. From isolation to cooperation: An alternative view of a system of experts. *Neural Information Processing Systems*, 8:605–611, 1996.

[4] J. Morimoto and K. Doya. Reinforcement learning of dynamic motor sequence: Learning to stand up. *IROS*, 3:1721–1726, 1998.

[5] J. Morimoto and K. Doya. Learning dynamic motor sequence in high-dimensional state space by reinforcement learning –learning to stand up–. *IEICE*, J82-D-II(11):2118–2131, 1999. (in Japanese).

[6] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[7] V. Gullapalli. A stchastic reinforcement learning algorithm for learning real-valued functions. *Neural Netwarks*, 3:671–692, 1990.

[8] T. Kondo and K. Ito. A proposal of an on-line evolutionary reinforcement learning. *13th Autonomous Distributed Symposium*, 2001. (In Japanese).

[9] R. Smith. *Open Dynamics Engine.* http://opende. sourceforge. net/ode. html.