PAPER

A State Space Compression Method Based on Multivariate Analysis for Reinforcement Learning in High-dimensional Continuous State Spaces

Hideki SATOH[†a)], ***Member***

SUMMARY A state space compression method based on multivariate analysis was developed and applied to reinforcement learning for high-dimensional continuous state spaces. First, useful components in the state variables of an environment are extracted and meaningless ones are removed by using multiple regression analysis. Next, the state space of the environment is compressed by using principal component analysis so that only a few principal components can express the dynamics of the environment. Then, a basis of a feature space for function approximation is constructed based on orthonormal bases of the important principal components. A feature space is thus autonomously construct without preliminary knowledge of the environment, and the environment is effectively expressed in the feature space. An example synchronization problem for multiple logistic maps was solved using this method, demonstrating that it solves the curse of dimensionality and exhibits high performance without suffering from disturbance states.

key words: *reinforcement learning, curse of dimensionality, multivariate analysis, approximation, nonlinear*

1. Introduction

Reinforcement learning [1] has been receiving much attention because it can construct, without a supervisor or a model of an environment, a control function that computes an action based on the state variables of the environment. To apply reinforcement learning to the real world, we need to extend it to high-dimensional continuous state spaces because most problems in the real world comprise high-dimensional continuous states. However, as the dimension of the state space increases, the dimension of a feature space, in which a nonlinear control function is approximated, increases exponentially [1]. That is, the number of parameters to be learned increases exponentially. This is referred to as the "curse of dimensionality" and is one of the most serious problems in function approximation and reinforcement learning.

To approximate a nonlinear control function and to solve the curse of dimensionality, various approaches to autonomously constructing a feature space have been investigated. In the wire fitting approach, a feature space for function approximation is constructed based on the control wires, that fit the control surface and create the relationship between the state and the action. Restricting the number of control wires enables the optimal action for a given state to be quickly found and reduces the number of parameters to be learned [2]. Tile coding with hashing allocates tiles whose position and size are randomly selected in a state space, and uses the tiles as a basis that constructs a feature space. By restricting the number of tiles, tile coding compresses the state space [1]. A tree-based algorithm has been developed that can handle a large continuous state space [3]. The self-organising map can perform function approximation autonomously in a feature space with a limited dimension [4]. These methods aim at effective construction of a feature space for function approximation. However, their function approximations are redundant because the basis of the feature space is not orthogonal. Because the mathematical structure of the feature space is unclear, it is difficult to ensure important properties such as the stability of the system. Hierarchical reinforcement learning is another approach to solving the curse of dimensionality. It constructs a hierarchical structure in the environment and controls each level in the hierarchy by using multiple agents. The number of parameters that each agent has to learn is reduced by dividing the state space based on the hierarchical structure [5], [6]. However, because preliminary knowledge is required in most cases to construct a hierarchical structure, it is difficult to apply hierarchical reinforcement learning to an environment whose structure is unknown. It is also difficult to mathematically clarify the demand and reason for constructing the structure.

In this paper, a method is presented that solves these problems and the curse of dimensionality. It is a state space compression method that uses multivariate analysis. It autonomously constructs a feature space without preliminary knowledge of the environment. It is based on orthogonal bases, so the mathematical structure of the system is clear in the feature space, and it is possible to mathematically analyze the dynamics and stability in a moment vector space [7]. Its ability to solve a synchronization problem for multiple logistic maps [8] using reinforcement learning demon-

Manuscript received December 22, 2005.
Manuscript revised April 4, 2006.
Final manuscript received April 17, 2006.

[†]The author is with the Future University-Hakodate, Hakodate-shi, 041-8655 Japan.
a) E-mail: jamisato@m.ieice.org

strated that the state space compression method using multivariate analysis exhibits high performance and solves the curse of dimensionality.

2. Actor-critic Method for Continuous-state Continuous-action Environment

2.1 Reinforcement Learning

A system with reinforcement learning [1] comprises two elements: an agent and an environment. The former provides a control function, and the latter denotes the target system to be controlled. The agent observes the state variables and the reward from the environment, updates the control function accordingly, and outputs a new control input to the environment. The control function and control input are referred to as policy and action, respectively.

Let us consider a discrete-time continuous-state continuous-action environment defined by a Markov decision process (MDP). Let $\boldsymbol{u} \stackrel{\text{def}}{=} {}^{\text{t}}(u_1, \cdots, u_{d_\text{u}})$ be the action of dimension d_u, $\boldsymbol{s} \stackrel{\text{def}}{=} {}^{\text{t}}(s_1, \cdots, s_{d_\text{s}})$ be the state of dimension d_s, $\boldsymbol{\mathcal{D}_\text{s}} \stackrel{\text{def}}{=} \{s_i | s_{\text{min}i} \leq s_i \leq s_{\text{min}i} + D_{\text{s}i}, 1 \leq i \leq d_\text{s}\}$, $\boldsymbol{\mathcal{D}_\text{u}} \stackrel{\text{def}}{=} \{u_i | u_{\text{min}i} \leq u_i \leq u_{\text{min}i} + D_{\text{u}i}, 1 \leq i \leq d_\text{u}\}$, r be the immediate reward, and superscript t denote transposition. In an MDP environment, transition probability density function $prob(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{u}_t)$ is defined to describe the relationships among $\boldsymbol{s}_t$, $\boldsymbol{u}_t$, and $\boldsymbol{s}_{t+1}$, and r_t is given when the agent takes action $\boldsymbol{u}_t$ and the state changes from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$. Here subscript t denotes the discrete time.

Reinforcement learning differs from supervised learning, such as neural networks, machine learning, and statistical pattern recognition. Without any help from a supervisor, it maximizes discount return R_t defined by

$$\begin{aligned} R_t &\stackrel{\text{def}}{=} \sum_{k=0}^{t_\text{max}} {\nu_\text{DR}}^k r_{t+k+1} \quad \text{for} \quad 0 < t_\text{max} \leq \infty \\ &= r_{t+1} + \nu_\text{DR} R_{t+1}, \end{aligned} \tag{1}$$

where ν_DR is the discount rate ($0 \leq \nu_\text{DR} \leq 1$). Actor-critic methods [9], [10], which is a class of methods for solving reinforcement learning problems, consist of two parts: an actor and a critic. The actor observes state $\boldsymbol{s}_t$ and decides on action $\boldsymbol{u}_t$ based on policy $p(\boldsymbol{u}_t|\boldsymbol{s}_t)$, which denotes the conditional probability density function (PDF) of $\boldsymbol{u}_t$. The critic observes r_t, updates state-value function $V(\boldsymbol{s}) \stackrel{\text{def}}{=} E[R_t|\boldsymbol{s}_t = \boldsymbol{s}]$ accordingly, and learns policy $p(\boldsymbol{u}|\boldsymbol{s})$.

2.2 Function Approximation

Function approximation is essential for reinforcement learning in a continuous-state continuous-action environment. It constructs an approximation of a nonlinear function. In this section, linear function approximations [1] are summarized that approximately express $V(\boldsymbol{s})$ and $p(\boldsymbol{u}|\boldsymbol{s})$.

State-value function $V(\boldsymbol{s})$ is approximated by $V(\boldsymbol{s}, \boldsymbol{c})$, which is defined by the following expansion:

$$\begin{aligned} V(\boldsymbol{s}, \boldsymbol{c}) &\stackrel{\text{def}}{=} \sum_{i=0}^{N_\text{V}} c_i \phi_{\text{V}\,i}(\boldsymbol{s}) \\ &= {}^{\text{t}}\boldsymbol{c} \cdot \boldsymbol{\phi}_\text{V}(\boldsymbol{s}), \end{aligned} \tag{2}$$

where N_V denotes the degree of expansion of $V(\boldsymbol{s})$, $\boldsymbol{c} \stackrel{\text{def}}{=} {}^{\text{t}}(c_0, \cdots, c_{N_\text{V}})$, and $\boldsymbol{\phi}_\text{V}(\boldsymbol{s}) \stackrel{\text{def}}{=} {}^{\text{t}}(\phi_{\text{V}\,0}(\boldsymbol{s}), \cdots, \phi_{\text{V}\,N_\text{V}}(\boldsymbol{s}))$. Here, an orthonormal basis described in Appendix A is used for $\boldsymbol{\phi}_\text{V}(\boldsymbol{s})$. An $N_\text{V} + 1$ dimensional vector space with basis $\boldsymbol{\phi}_\text{V}(\boldsymbol{s})$ is referred to as a feature space. From Eq. (2), $V(\boldsymbol{s})$ is expressed by vector $\boldsymbol{c}$ in the feature space with basis $\boldsymbol{\phi}_\text{V}(\boldsymbol{s})$.

To simplify approximation of $p(\boldsymbol{u}|\boldsymbol{s})$, let us assume that u_i obeys a normal distribution and that u_i and u_j are independent for $i \neq j$. Then, $p(\boldsymbol{u}|\boldsymbol{s})$ can be approximated by the following equation:

$$p(\boldsymbol{u}|\boldsymbol{s}) \cong \prod_{i=1}^{d_\text{u}} p_\text{G}(u_i|\boldsymbol{s}, \boldsymbol{\xi}_{\text{m}i}, \boldsymbol{\xi}_{\sigma i}), \tag{3}$$

$$p_\text{G}(u_i|\boldsymbol{s}, \boldsymbol{\xi}_{\text{m}i}, \boldsymbol{\xi}_{\sigma i}) \stackrel{\text{def}}{=} \frac{1}{(2\pi)^{1/2}\sigma_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i})} \exp\left(\frac{-(u_i - m_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\text{m}i}))^2}{2\sigma_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i})^2}\right),$$

where $m_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\text{m}i})$ denotes an approximation of the mean of u_i when the state of the environment is $\boldsymbol{s}$, $\sigma_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i})$ denotes an approximation of the standard deviation of u_i when the state of environment is $\boldsymbol{s}$, $\boldsymbol{\xi}_{\text{m}i} \stackrel{\text{def}}{=} {}^{\text{t}}(\xi_{\text{m}i0}, \cdots, \xi_{\text{m}iN_\text{m}})$, $\boldsymbol{\xi}_{\sigma i} \stackrel{\text{def}}{=} {}^{\text{t}}(\xi_{\sigma i0}, \cdots, \xi_{\sigma iN_\sigma})$, N_m denotes the degree of expansion of m_G, and N_σ denotes the degree of expansion of σ_G.

Let $\boldsymbol{\phi}_\text{m}(\boldsymbol{s}) \stackrel{\text{def}}{=} {}^{\text{t}}(\phi_{\text{m}0}(\boldsymbol{s}), \cdots, \phi_{\text{m}N_\text{m}}(\boldsymbol{s}))$ be an orthonormal basis. In the same manner as the approximation of $V(\boldsymbol{s})$, $m_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\text{m}i})$ is defined as follows:

$$m_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\text{m}i}) \stackrel{\text{def}}{=} {}^{\text{t}}\boldsymbol{\xi}_{\text{m}i} \cdot \boldsymbol{\phi}_\text{m}(\boldsymbol{s}). \tag{4}$$

Because $\sigma_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i})$ is positive, it cannot be directly expanded using an orthonormal basis, as in Eqs. (2) and (4). Thus, the following function approximation is used to transform a variable expanded by an orthonormal function expansion into a positive number:

$$\sigma_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i}) \stackrel{\text{def}}{=} \left(\frac{\tilde{\sigma}_\text{umax} - \tilde{\sigma}_\text{umin}}{1 + \exp(-{}^{\text{t}}\boldsymbol{\xi}_{\sigma i} \cdot \boldsymbol{\phi}_\sigma(\boldsymbol{s}))} + \tilde{\sigma}_\text{umin}\right) D_{\text{u}i}, \tag{5}$$

where $\boldsymbol{\phi}_\sigma(\boldsymbol{s}) \stackrel{\text{def}}{=} {}^{\text{t}}(\phi_{\sigma 0}(\boldsymbol{s}), \cdots, \phi_{\sigma N_\sigma}(\boldsymbol{s}))$ is the orthonormal basis. The $\min(\sigma_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i}))$ and $\max(\sigma_\text{G}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i}))$

are given by $\tilde{\sigma}_{\mathrm{umin}} D_{\mathrm{u}i}$ and $\tilde{\sigma}_{\mathrm{umax}} D_{\mathrm{u}i}$, respectively. Using these function approximations enables the mean of u_i to be expressed by vector $\boldsymbol{\xi}_{\mathrm{m}i}$ in a feature space with basis $\boldsymbol{\phi}_{\mathrm{m}}(\boldsymbol{s})$ and the standard deviation of u_i to be expressed by vector $\boldsymbol{\xi}_{\sigma i}$ in a feature space with basis $\boldsymbol{\phi}_{\sigma}(\boldsymbol{s})$.

2.3 Actor-critic Method

Let $\varepsilon_{\mathrm{TD}\,t}$, $\boldsymbol{c}_t$, $\boldsymbol{\xi}_{\mathrm{m}i;t}$, and $\boldsymbol{\xi}_{\sigma i;t}$ be the values of TD error $\varepsilon_{\mathrm{TD}}$ [1], $\boldsymbol{c}$, $\boldsymbol{\xi}_{\mathrm{m}i}$, and $\boldsymbol{\xi}_{\sigma i}$ at time t, respectively. An actor-critic method based on TD learning [1] and on the function approximations derived in Sect. 2.2 is given as follows:

Algorithm 1: Actor-critic method.

(1-1) Initialization-1: Set $\boldsymbol{c}_0$, $\boldsymbol{\xi}_{\mathrm{m}i;0}$, and $\boldsymbol{\xi}_{\sigma i;0}$ for $\forall i$.

(1-2) Initialization-2: Set $t = 0$, $\boldsymbol{s}_0$, and $\boldsymbol{u}_0$.

(1-3) Environment: Receive $\boldsymbol{u}_t$, and output r_{t+1} and $\boldsymbol{s}_{t+1}$ to the agent.

(1-4) Actor-critic:

(1) Critic-1: Calculate $\varepsilon_{\mathrm{TD}\,t}$ and $\boldsymbol{c}_{t+1}$ by using Eqs. (9) and (11) (Update $V(\boldsymbol{s}, \boldsymbol{c})$).

(2) Critic-2: Calculate $\boldsymbol{\xi}_{\mathrm{m}i;t+1}$, $\boldsymbol{\xi}_{\sigma i;t+1}$ by using Eqs. (12) and (13) (Update $p_{\mathrm{G}}(u_i|\boldsymbol{s}, \boldsymbol{\xi}_{\mathrm{m}i}, \boldsymbol{\xi}_{\sigma i})$) for $\forall i$.

(3) Actor: Set $u_{i;t+1}$ based on $p_{\mathrm{G}}(u_{i;t+1}|\boldsymbol{s}_{t+1}, \boldsymbol{\xi}_{\mathrm{m}i;t+1}, \boldsymbol{\xi}_{\sigma i;t+1})$ for $\forall i$.

(1-5) Set $t = t + 1$. If an episode finished, go to Step (1-2), else go to Step (1-3).

Here, episode denotes a trial [1]. In Step (1-1), initial values $\boldsymbol{c}_0$, $\boldsymbol{\xi}_{\mathrm{m}i;0}$, and $\boldsymbol{\xi}_{\sigma i;0}$ are set for $\forall i$ as follows:

$$\boldsymbol{c}_0 = \boldsymbol{0},$$

$$\xi_{\mathrm{m}ij;0} = \begin{cases} u_{\mathrm{min}i} + D_{\mathrm{u}i}/2 & \text{for} \quad j = 0 \\ 0 & \text{for} \quad 1 \le j \le N_{\mathrm{m}}, \end{cases}$$

$$\xi_{\sigma ij;0} = \begin{cases} \xi_0 & \text{for} \quad j = 0 \\ 0 & \text{for} \quad 1 \le j \le N_{\sigma}, \end{cases}$$

where ξ_0 is set so that $\sigma_{\mathrm{G}}(\boldsymbol{s}, \boldsymbol{\xi}_{\sigma i})$ takes an appropriate value for initial learning. In Sect. 5, ξ_0 is set to -1.0 for $\forall i$, so that $\sigma_{\mathrm{G}}(\boldsymbol{s}_0, \boldsymbol{\xi}_{\sigma i;0}) = 0.135 D_{\mathrm{u}i}$.

Let $p(\boldsymbol{s})$ be the PDF of $\boldsymbol{s}$ and $MSE(\boldsymbol{c})$ be the mean square error (MSE) of state-value function $V(\boldsymbol{s})$ defined by

$$MSE(\boldsymbol{c}) \stackrel{\mathrm{def}}{=} \int_{\boldsymbol{s} \in \boldsymbol{\mathcal{D}}_{\mathrm{s}}} p(\boldsymbol{s}) (V(\boldsymbol{s}) - V(\boldsymbol{s}, \boldsymbol{c}))^2 \mathrm{d}\boldsymbol{s}. \quad (6)$$

A method [1] that minimizes $MSE(\boldsymbol{c})$ with respect to $\boldsymbol{c}$ and updates $V(\boldsymbol{s}, \boldsymbol{c})$ is summarized below. Let $\hat{V}(\boldsymbol{s})$ be an estimation of the state-value function. We can obtain $\hat{V}(\boldsymbol{s})$ by using the following equation:

$$\hat{V}(\boldsymbol{s}_t) = r_{t+1} + \nu_{\mathrm{DR}} V(\boldsymbol{s}_{t+1}, \boldsymbol{c}_t). \quad (7)$$

Thus, by replacing $\boldsymbol{x}$ with $\boldsymbol{s}$, ζ with ζ_{V}, g with V, and $\hat{g}$ with $\hat{V}$ in Eq. (A·8), we obtain the following equation:

$$\boldsymbol{c}_{t+1} = \boldsymbol{c}_t + \zeta_{\mathrm{V}} (\hat{V}(\boldsymbol{s}_t) - V(\boldsymbol{s}_t, \boldsymbol{c}_t)) \nabla_c V(\boldsymbol{s}_t, \boldsymbol{c}_t), \quad (8)$$

where ζ_{V} is the step size. By substituting TD error $\varepsilon_{\mathrm{TD}\,t}$, defined by

$$\begin{aligned} \varepsilon_{\mathrm{TD}\,t} &\stackrel{\mathrm{def}}{=} \hat{V}(\boldsymbol{s}_t) - V(\boldsymbol{s}_t, \boldsymbol{c}_t) \\ &= r_{t+1} + \nu_{\mathrm{DR}} V(\boldsymbol{s}_{t+1}, \boldsymbol{c}_t) - V(\boldsymbol{s}_t, \boldsymbol{c}_t), \end{aligned} \quad (9)$$

and $\nabla_c V(\boldsymbol{s}, \boldsymbol{c}) = \boldsymbol{\phi}_{\mathrm{V}}(\boldsymbol{s})$ into Eq. (8), we obtain

$$\boldsymbol{c}_{t+1} = \boldsymbol{c}_t + \zeta_{\mathrm{V}} \varepsilon_{\mathrm{TD}\,t} \boldsymbol{\phi}_{\mathrm{V}}(\boldsymbol{s}_t). \quad (10)$$

Let ν_{Tr} be the trace-decay parameter [1]. By introducing eligibility trace $\boldsymbol{\eta}_t$ [1] into the above equation, we obtain the following equation for updating $\boldsymbol{c}$:

$$\begin{aligned} \boldsymbol{\eta}_t &= \nu_{\mathrm{DR}} \nu_{\mathrm{Tr}} \boldsymbol{\eta}_{t-1} + \boldsymbol{\phi}_{\mathrm{V}}(\boldsymbol{s}_t), \\ \boldsymbol{c}_{t+1} &= \boldsymbol{c}_t + \zeta_{\mathrm{V}} \varepsilon_{\mathrm{TD}\,t} \boldsymbol{\eta}_t. \end{aligned} \quad (11)$$

Here, $\boldsymbol{\eta}_0$, the initial value of $\boldsymbol{\eta}_t$, is set to $\boldsymbol{0}$.

In the same manner as for $V(\boldsymbol{s}, \boldsymbol{c})$, $p_{\mathrm{G}}(u_i|\boldsymbol{s}, \boldsymbol{\xi}_{\mathrm{m}i}, \boldsymbol{\xi}_{\sigma i})$ can be updated. Let ζ_{m} be the step size to update $\boldsymbol{\xi}_{\mathrm{m}i;t}$, and ζ_{σ} be the step size to update $\boldsymbol{\xi}_{\sigma i;t}$. By replacing $\boldsymbol{c}_t$ with $\boldsymbol{\xi}_{\mathrm{m}i;t}$, $\tilde{g}$ with $u_{i;t}$, $g(\boldsymbol{x}_t, \boldsymbol{c}_t)$ with $m_{\mathrm{G}}(\boldsymbol{s}_t, \boldsymbol{\xi}_{\mathrm{m}i;t})$, and $\nabla_{\mathrm{c}} g(\boldsymbol{x}, \boldsymbol{c})$ with $\nabla_{\xi_{\mathrm{m}i}} \boldsymbol{\xi}_{\mathrm{m}i} \cdot \boldsymbol{\phi}_{\mathrm{m}}(\boldsymbol{s}_t) = \boldsymbol{\phi}_{\mathrm{m}}(\boldsymbol{s}_t)$ in Eq. (A·9), we obtain the following equation:

$$\begin{aligned} \boldsymbol{\xi}_{\mathrm{m}i;t+1} = \boldsymbol{\xi}_{\mathrm{m}i;t} \\ + \zeta_{\mathrm{m}} \varepsilon_{\mathrm{TD}\,t} (u_{i;t} - m_{\mathrm{G}}(\boldsymbol{s}_t, \boldsymbol{\xi}_{\mathrm{m}i;t})) \boldsymbol{\phi}_{\mathrm{m}}(\boldsymbol{s}_t). \end{aligned} \quad (12)$$

In the same manner as for Eq. (12), by replacing $\boldsymbol{c}_t$ with $\boldsymbol{\xi}_{\sigma i;t}$, $\tilde{g}$ with $|u_{i;t} - m_{\mathrm{G}}(\boldsymbol{s}_t, \boldsymbol{\xi}_{\mathrm{m}i;t})|$, $g(\boldsymbol{x}_t, \boldsymbol{c}_t)$ with $\sigma_{\mathrm{G}}(\boldsymbol{s}_t, \boldsymbol{\xi}_{\mathrm{m}i;t})$, and $\nabla_{\mathrm{c}} g(\boldsymbol{x}, \boldsymbol{c})$ with $\nabla_{\xi_{\sigma i}} \boldsymbol{\xi}_{\sigma i} \cdot \boldsymbol{\phi}_{\sigma}(\boldsymbol{s}_t) = \boldsymbol{\phi}_{\sigma}(\boldsymbol{s}_t)$ in Eq. (A·9), we obtain the following equation:

$$\begin{aligned} \boldsymbol{\xi}_{\sigma i;t+1} = \boldsymbol{\xi}_{\sigma i;t} + \zeta_{\sigma} \varepsilon_{\mathrm{TD}\,t} (|u_{i;t} - m_{\mathrm{G}}(\boldsymbol{s}_t, \boldsymbol{\xi}_{\mathrm{m}i;t})| \\ - \sigma_{\mathrm{G}}(\boldsymbol{s}_t, \boldsymbol{\xi}_{\sigma i;t})) \boldsymbol{\phi}_{\sigma}(\boldsymbol{s}_t). \end{aligned} \quad (13)$$

3. State Space Compression Method and its Application to Reinforcement Learning

3.1 Framework of State Space Compression and Learning

Let us consider a system in which several controllers control an environment, and assume that the agent is one of the controllers. The controllers observe the state variables and the reward from the environment, and output control inputs to the environment. Suppose that the agent can observe state $\boldsymbol{x}_t$, reward r_{t+1}, and control input $\boldsymbol{v}_t$ that includes control inputs determined by the other controllers and action $\boldsymbol{u}_t$ determined by the agent. Note that it is assumed that $\boldsymbol{x}_t$ and $\boldsymbol{v}_t$ does not not always include all the states in the environment and all the control inputs determined by the

other controllers. To simplify the equations described later, the relationship between $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$ is defined as follows:

$$\boldsymbol{u}_t = G_{\mathrm{u}}\boldsymbol{v}_t + \boldsymbol{g}_{\mathrm{u}}. \tag{14}$$

Here, $\boldsymbol{x}_t \stackrel{\mathrm{def}}{=} {}^{\mathrm{t}}(x_{1;t}, \cdots, x_{d_{\mathrm{x}};t})$, $\boldsymbol{v}_t \stackrel{\mathrm{def}}{=} {}^{\mathrm{t}}(v_{1;t}, \cdots, v_{d_{\mathrm{v}};t})$, $\boldsymbol{D}_{\mathrm{x}} \stackrel{\mathrm{def}}{=} \{x_i \mid x_{\mathrm{min}i} \le x_i \le x_{\mathrm{min}i} + D_{\mathrm{x}i}, 1 \le i \le d_{\mathrm{x}}\}$, $\boldsymbol{D}_{\mathrm{v}} \stackrel{\mathrm{def}}{=} \{v_i \mid v_{\mathrm{min}i} \le v_i \le v_{\mathrm{min}i} + D_{\mathrm{v}i}, 1 \le i \le d_{\mathrm{v}}\}$, and G_{u} and $\boldsymbol{g}_{\mathrm{u}}$ are a matrix and a vector, respectively, obtained from $v_{\mathrm{min}i}$, $D_{\mathrm{v}i}$, $u_{\mathrm{min}i}$, and $D_{\mathrm{u}i}$.

When the basis of $\boldsymbol{x}$ for function approximation is constructed using the direct product of the bases of x_i, the degree of expansion of $\boldsymbol{x}$ increases proportionally to the d_{x}th power of the degree of expansion of x_i (see Eq. (A·4)). This is referred to as the "curse of dimensionality." The framework of state space compression and reinforcement learning presented in this section solves this problem. In the framework, the agent compresses the state space, constructs a feature space with a limited dimension, extracts the principal dynamics in $\boldsymbol{x}_t$, and determines $\boldsymbol{u}_t$. The procedure operated by the agent is summarized as follows (the numbers in parentheses correspond to (1), $\cdots$, (7) in Step (2-4) of Algorithm 2). (3) Compute statistics such as mean and covariance of $\boldsymbol{x}$. (4) Compute the resolution (which denotes the importance) of x_i using multiple regression analysis. Based on the resolution, extract useful components of $\boldsymbol{x}$ and remove meaningless ones. (5) Compute principal axes matrix M, which maps the useful components of $\boldsymbol{x}$ to principal component vector $\boldsymbol{y} = {}^{\mathrm{t}}(y_1, \cdots, y_{d_{\mathrm{y}}})$, which express the principal dynamics in the useful components of $\boldsymbol{x}$. (6) Compute the resolution of y_i. (7) Based on the resolution of y_i, determine bases $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$. (1) Transform $\boldsymbol{x}$ into $\boldsymbol{y}$ using principal axes matrix M, and normalize $\boldsymbol{y}$ into normalized principal state $\boldsymbol{s}$ so that $\boldsymbol{s}$ contains the principal dynamics in $\boldsymbol{x}$ in a bounded space. (2) Compute action $\boldsymbol{u}$ based on $\boldsymbol{s}$ using the actor-critic method in the feature spaces with bases $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$.

In the above procedure, it is necessary, before performing the actor-critic method, to obtain principal axes matrix M, bases $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$, and normalized principal state $\boldsymbol{s}$. Thus, before performing the actor-critic method, $\boldsymbol{u}$ is determined at random, and statistics such as the mean and covariance of $\boldsymbol{x}$ are computed. After a sufficiently long time has elapsed, M, $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$ are computed. They are updated at appropriate intervals so that they include the latest dynamics in the environment. The time when M, $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$ are computed is indicated by using two variables: n_{MA} and trg_{MA}. The former denotes the number of times M, $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$ have been updated, and the latter denotes the trigger to update M, $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$. Here, M, $\boldsymbol{\phi}_{\mathrm{v}}$, $\boldsymbol{\phi}_{\mathrm{m}}$, and $\boldsymbol{\phi}_{\sigma}$ are updated at regular interval T_{RB}. The structure and algorithm for the agent are shown in Fig. 1 and given by Algorithm 2, respectively.

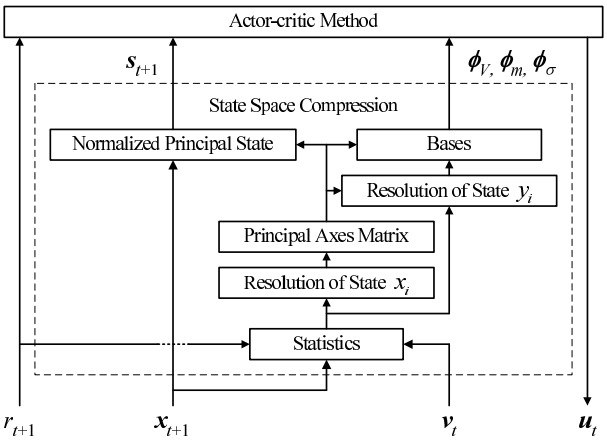

Fig. 1 Structure of agent with state space compression.

Algorithm 2: Reinforcement learning with state space compression.

(2-1) Initialization-1: Set $trg_{\mathrm{MA}} = 0$, $n_{\mathrm{MA}} = 0$, $\boldsymbol{c}_0$, $\boldsymbol{\xi}_{\mathrm{m}0}$, and $\boldsymbol{\xi}_{\sigma 0}$.

(2-2) Initialization-2: Set $t = 0$, $\boldsymbol{x}_0$, and $\boldsymbol{u}_0$.

(2-3) Environment: Receive $\boldsymbol{u}_t$, and output r_{t+1} and $\boldsymbol{x}_{t+1}$ to the agent.

(2-4) Agent:

(1) Transform $\boldsymbol{x}_{t+1}$ into $\boldsymbol{s}_{t+1}$ by using Eqs. (24) and (32) if $n_{\mathrm{MA}} \ge 1$.

(2) Perform actor-critic method (Step (1-4) in Algorithm 1) if $n_{\mathrm{MA}} \ge 1$, or set $\boldsymbol{u}_t$ randomly if $n_{\mathrm{MA}} = 0$.

(3) Compute statistics of $\boldsymbol{x}$, and set n_{MA} and trg_{MA} by using Algorithm 3.

(4) Compute resolution of state x_i by using Eq. (20) if $trg_{\mathrm{MA}} = 1$.

(5) Update principal axes matrix and mean by using Eqs. (23) and (25) if $trg_{\mathrm{MA}} = 1$.

(6) Compute resolution of y_i by using Eq. (31) if $trg_{\mathrm{MA}} = 1$.

(7) Update bases as shown in Sect. 3.7 and set $trg_{\mathrm{MA}} = 0$ if $trg_{\mathrm{MA}} = 1$.

(2-5) Set $t = t + 1$. If an episode finished, go to Step (2-2), else go to Step (2-3).

Each step of the agent is described in detail below.

3.2 Statistics of Environment

To enable multivariate analysis to be performed, the mean and covariance of $\boldsymbol{x}_t$, $\boldsymbol{x}_{t+1}$, $\boldsymbol{v}_t$, and r_{t+1} are computed at appropriate intervals by using Algorithm 3 so that they include the latest dynamics in the environment, where COV_{xx} denotes the covariance matrix of

$(x_{1;\tau}, \cdots, x_{d_{\rm x};\tau})$, $COV_{\rm vxvx}$ denotes the covariance matrix of $(v_{1;\tau}, \cdots, v_{d_{\rm v};\tau}, x_{1;\tau}, \cdots, x_{d_{\rm x};\tau})$, $COV_{\rm vxx'vxx'}$ denotes the covariance matrix of $(v_{1;\tau}, \cdots, v_{d_{\rm v};\tau}, x_{1;\tau}, \cdots, x_{d_{\rm x};\tau}, x_{1;\tau+1}, \cdots, x_{d_{\rm x};\tau+1})$, $\boldsymbol{cov}_{\rm x'vxi} \stackrel{\rm def}{=} {}^{\rm t}(cov(x_{i;\tau+1}, v_{1;\tau}), \cdots, cov(x_{i;\tau+1}, v_{d_{\rm v};\tau}), cov(x_{i;\tau+1}, x_{1;\tau}), \cdots, cov(x_{i;\tau+1}, x_{d_{\rm x};\tau}))$, $\boldsymbol{cov}_{\rm rvxx'} \stackrel{\rm def}{=} {}^{\rm t}(cov(r_{\tau+1}, v_{1;\tau}), \cdots, cov(r_{\tau+1}, v_{d_{\rm v};\tau}), cov(r_{\tau+1}, x_{1;\tau}), \cdots, cov(r_{\tau+1}, x_{d_{\rm x};\tau}), cov(r_{\tau+1}, x_{1;\tau+1}), \cdots, cov(r_{\tau+1}, x_{d_{\rm x};\tau+1}))$, $cov(x_\tau, y_\tau)$ denotes the covariance of x_τ and y_τ, $\bar{\boldsymbol{x}}$ denotes the mean of $\boldsymbol{x}_\tau$, and $t - T_{\rm S} + 1 \leq \tau \leq t$.

Algorithm 3: Compute Statistics.

(3-1) Initialization: If $n_{\rm MA} = 0$, set $T = T_{\rm S}$. If $n_{\rm MA} \geq 0$, set $T = T_{\rm RB}$.

(3-2) If the remainder of $t + 1$ upon division by T is equal to 0 (i.e., at periodic intervals of T), perform the following steps.

(3-3) Compute $COV_{\rm xx}$, $COV_{\rm vxvx}$, $COV_{\rm vxx'vxx'}$, $\boldsymbol{cov}_{\rm x'vxi}$, $\boldsymbol{cov}_{\rm rvxx'}$, and $\bar{\boldsymbol{x}}$.

(3-4) Set $n_{\rm MA} = n_{\rm MA} + 1$ and $trg_{\rm MA} = 1$.

Here, the statistics of $\boldsymbol{x}$ are initially computed at $t = T_{\rm S} - 1$. Thereafter, the bases are updated at periodic intervals of $T_{\rm RB}$.

3.3 Resolution of State x_i

To determine action $\boldsymbol{u}_t$, it is necessary to estimate which elements in state vector $\boldsymbol{x}_t$ are important. The resolution of $x_{i;t}$ for computing $\boldsymbol{u}_t$ is estimated by using the partial regression coefficients [11] and standard deviations.

First, the effect of $\boldsymbol{v}_t$ and $\boldsymbol{x}_t$ on $\boldsymbol{x}_{t+1}$ is estimated by using the partial regression coefficients for $\boldsymbol{x}_{t+1}$. The multiple regression equation [11] for criterion variable $\boldsymbol{x}_{t+1}$ with respect to explanatory variables $\boldsymbol{v}_t$ and $\boldsymbol{x}_t$ is expressed by

$$\boldsymbol{x}_{t+1} = \boldsymbol{\beta}_0 + B_{\rm v}\boldsymbol{v}_t + B_{\rm x}\boldsymbol{x}_t, \tag{15}$$

where β_{i0}, $\beta_{{\rm v}ij}$, and $\beta_{{\rm x}ij}$ are the partial regression coefficients, $\boldsymbol{\beta}_0 \stackrel{\rm def}{=} {}^{\rm t}(\beta_{10}, \cdots, \beta_{d_{\rm x}0})$, $\boldsymbol{\beta}_{{\rm v}i} \stackrel{\rm def}{=} {}^{\rm t}(\beta_{{\rm v}i1}, \cdots, \beta_{{\rm v}id_{\rm v}})$, $\boldsymbol{\beta}_{{\rm x}i} \stackrel{\rm def}{=} {}^{\rm t}(\beta_{{\rm x}i1}, \cdots, \beta_{{\rm x}id_{\rm x}})$, $B_{\rm v} \stackrel{\rm def}{=} {}^{\rm t}[\boldsymbol{\beta}_{{\rm v}1}, \cdots, \boldsymbol{\beta}_{{\rm v}d_{\rm x}}]$ is a $d_{\rm x} \times d_{\rm v}$ matrix, and $B_{\rm x} \stackrel{\rm def}{=} {}^{\rm t}[\boldsymbol{\beta}_{{\rm x}1}, \cdots, \boldsymbol{\beta}_{{\rm x}d_{\rm x}}]$ is a $d_{\rm x} \times d_{\rm x}$ matrix. Partial regression coefficients $\beta_{{\rm v}ij}$ and $\beta_{{\rm x}ij}$ are derived by the following equation [11]:

$$\boldsymbol{\beta}_{{\rm vx}i} = {COV_{\rm vxvx}}^{-1}\boldsymbol{cov}_{\rm x'vx}i, \tag{16}$$

where $\boldsymbol{\beta}_{{\rm vx}i} \stackrel{\rm def}{=} {}^{\rm t}({}^{\rm t}\boldsymbol{\beta}_{{\rm v}i}, {}^{\rm t}\boldsymbol{\beta}_{{\rm x}i})$.

Next, the effect of $\boldsymbol{v}_t$, $\boldsymbol{x}_t$, and $\boldsymbol{x}_{t+1}$ on r_{t+1} is estimated by using the partial regression coefficient for r_{t+1}. The multiple regression equation for criterion variable r_{t+1} with respect to explanatory variables $\boldsymbol{v}_t$, $\boldsymbol{x}_t$, and $\boldsymbol{x}_{t+1}$ is expressed by

$$r_{t+1} = b_0 + {}^{\rm t}\boldsymbol{b}_{\rm v}\boldsymbol{v}_t + {}^{\rm t}\boldsymbol{b}_{\rm x}\boldsymbol{x}_t + {}^{\rm t}\boldsymbol{b}_{\rm x'}\boldsymbol{x}_{t+1}, \tag{17}$$

where b_0, $b_{{\rm v}j}$, $b_{{\rm x}j}$, and $b_{{\rm x'}j}$ are the partial regression coefficients, $\boldsymbol{b}_{\rm v} \stackrel{\rm def}{=} {}^{\rm t}(b_{\rm v1}, \cdots, b_{{\rm v}d_{\rm v}})$, $\boldsymbol{b}_{\rm x} \stackrel{\rm def}{=} {}^{\rm t}(b_{\rm x1}, \cdots, b_{{\rm x}d_{\rm x}})$, and $\boldsymbol{b}_{\rm x'} \stackrel{\rm def}{=} {}^{\rm t}(b_{\rm x'1}, \cdots, b_{{\rm x'}d_{\rm x}})$. In the same manner as in Eq. (16), partial regression coefficients $b_{{\rm v}j}$, $b_{{\rm x}i}$, and $b_{{\rm x'}i}$ are derived by the following equation:

$$\boldsymbol{b}_{\rm vxx'} = (COV_{\rm vxx'vxx'})^{-1}\boldsymbol{cov}_{\rm rvxx'}, \tag{18}$$

where $\boldsymbol{b}_{\rm vxx'} \stackrel{\rm def}{=} {}^{\rm t}({}^{\rm t}\boldsymbol{b}_{\rm v}, {}^{\rm t}\boldsymbol{b}_{\rm x}, {}^{\rm t}\boldsymbol{b}_{\rm x'})$.

Let $res_{{\rm xv}ij}$ be the resolution of x_i for computing v_j. A combination of $\sigma(x_i)$, $D_{{\rm v}j}$, $|\beta_{{\rm v}ij}|$, and $|b_{{\rm x}i}| + |b_{{\rm x'}i}|$ is used to determine $res_{{\rm xv}ij}$ for three reasons:

(1) The effect of $x_{i;t}$ on $v_{j;t}$ denotes the agent itself. Thus, if the effect of $x_{i;t}$ on $v_{j;t}$ is used to determine the resolution, a positive feedback loop arises, and a system with the agent may become unstable. Thus, instead of the effect of $x_{i;t}$ on $v_{j;t}$, $|\beta_{{\rm v}ij}|$, which denotes the effect of $v_{j;t}$ on $x_{i;t+1}$, is used.

(2) Important information about the structure and dynamics of the system are contained in $\sigma(x_i)$ and $\sigma(v_j)$. However, $\sigma(v_j)$ contains the motion in the agent, so the system may become unstable if $\sigma(v_j)$ is used for the same reason described above. Thus, $D_{{\rm v}j}$ instead of $\sigma(v_j)$ and $\sigma(x_i)$ are used.

(3) Because the agent works to obtain a large reward, it is effective to introduce $|b_{{\rm x}i}| + |b_{{\rm x'}i}|$, which denotes the effect of $x_{i;t}$ and $x_{i;t+1}$ on r_{t+1}.

Based on these reasons, $res_{{\rm xv}ij}$ is determined by

$$res_{{\rm xv}ij} = \sigma(x_i)^{\gamma_1} {D_{\rm v}j}^{\gamma_2} |\beta_{{\rm v}ij}|^{\gamma_3} (|b_{{\rm x}i}| + |b_{{\rm x'}i}|)^{\gamma_4}, \tag{19}$$

where $\gamma_1, \cdots, \gamma_4$ are given constants used to adjust the combination. Here, $\sigma(x_i) = cov(x_i, x_i)^{1/2}$, and $\sigma(x_i)$ is obtaind in Step (3-3) of Algorithm 3.

Let $res_{{\rm xu}i}$ be the resolution of x_i used for computing $\boldsymbol{u}_t$. As shown in Sect. 3.1, the relationship between $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$ is defined by Eq. (14). Thus, $res_{{\rm xu}i}$ is determined by the weighted sum of $res_{{\rm xv}ij}$ with respect to j as follows:

$$res_{{\rm xu}i} = \sum_{j=1}^{d_{\rm u}} ({G_{\rm u}}^{\rm t}(res_{{\rm xv}i1}, \cdots, res_{{\rm xv}id_{\rm x}}))_j, \tag{20}$$

where $(\cdot)_j$ denotes entry j of a vector.

3.4 Principal Axes Matrix

As mentioned in Sect. 3.1, when the dimension of $\boldsymbol{x}_t$ is high, the degree of expansion of $\boldsymbol{x}_t$ is too large to perform reinforcement learning. Thus, it is necessary to express the dynamics in $\boldsymbol{x}_t$ in a low-dimensional feature space. In this section, principal components, which represent the important dynamics, are derived. Let $y_{i;t}$ be the ith principal component of $\boldsymbol{x}_t$, let $\boldsymbol{y}_t$ be ${}^{\rm t}(y_{1;t}, \cdots, y_{d_{\rm y};t})$. The importance of y_i is computed in

Sect. 3.5 and y_i used for control is determined in Sect. 3.7. Thus, $d_{\rm y}$ is set to $d_{\rm x}$.

Principal component vector $\boldsymbol{y}_t$ is obtained as the product of principal axes matrix M and state vector $\boldsymbol{x}_t$, and M is derived from the eigenvector of $COV_{\rm xx}$ [11]. Most of the dynamics in $\boldsymbol{x}_t$ can be expressed by using only a small number of elements in $\boldsymbol{y}_t$ if $x_{i;t}$ and $x_{j;t}$ for $i \neq j$ are correlated. However, it is not always true that elements in $\boldsymbol{x}_t$ with large dynamics contribute to reward r_{t+1}. Some elements may not be related to r_{t+1}. To remove the effect of such elements on $\boldsymbol{y}_t$, the covariance matrix for $res_{{\rm xu}i}x_i$ is used instead of that for x_i. Let ${\boldsymbol{x}^{\rm R}}_t \stackrel{\rm def}{=} {}^{\rm t}(res_{\rm xu1}x_{1;t}, \cdots, res_{{\rm xu}d_{\rm x}}x_{d_{\rm x};t})$ and $COV_{\rm xx}^{\rm R}$ be the covariance matrix of $\boldsymbol{x}^{\rm R}$. From the definition of the covariance, $COV_{\rm xx}^{\rm R}$ is obtained by using the elements of $COV_{\rm xx}$ as follows:

$$COV_{\rm xx}^{\rm R} = [res_{{\rm xu}i}res_{{\rm xu}j}cov(x_i, x_j)]. \tag{21}$$

Let ${\boldsymbol{e}^{\rm R}}_i$ be the ith eigen vector of $COV_{\rm xx}^{\rm R}$ and $M^{\rm R} \stackrel{\rm def}{=} [{\boldsymbol{e}^{\rm R}}_1, \cdots, {\boldsymbol{e}^{\rm R}}_{d_{\rm x}}]$ be the principal axes matrix of ${\boldsymbol{x}^{\rm R}}_t$. We then obtain principal component vector $\boldsymbol{y}_t$ as follows:

$$\begin{aligned}\boldsymbol{y}_t &= {}^{\rm t}M^{\rm R}{\boldsymbol{x}^{\rm R}}_t \\ &= {}^{\rm t}M^{\rm R}{\rm diag}[res_{{\rm xu}i}]\boldsymbol{x}_t,\end{aligned} \tag{22}$$

where ${\rm diag}[res_{{\rm xu}i}]$ denotes the diagonal matrix of $res_{{\rm xu}i}$. Let principal axes matrix M of $\boldsymbol{x}_t$ be defined by the following equation:

$$M \stackrel{\rm def}{=} {\rm diag}[res_{{\rm xu}i}]M^{\rm R}. \tag{23}$$

By substituting Eq. (23) into Eq. (22), we obtain

$$\boldsymbol{y}_t = {}^{\rm t}M\boldsymbol{x}_t. \tag{24}$$

Let λ_i be the variance in $y_{i;t}$, and $\bar{\boldsymbol{y}}$ be the mean of $\boldsymbol{y}_t$. The former is equal to the ith eigen value of $COV_{\rm xx}^{\rm R}$, and the latter is derived by

$$\bar{\boldsymbol{y}} = {}^{\rm t}M\bar{\boldsymbol{x}}. \tag{25}$$

3.5 Resolution of State y_i

As shown in Appendix A, bases $\boldsymbol{\phi}_{\rm v}$, $\boldsymbol{\phi}_{\rm m}$, and $\boldsymbol{\phi}_\sigma$ are determined by the degree of expansion of y_i. Thus, to obtain a large discount return, it is reasonable to assign a larger value to the degree of expansion of $y_{i;t}$ if $y_{i;t}$ is closely related to $\boldsymbol{u}_t$ and r_{t+1}. The resolution of $y_{i;t}$ is derived so as to achieve this aim in the same manner as in Sect. 3.3.

The multiple regression equation for $\boldsymbol{y}_{t+1}$ is described in the same manner as in Eq. (15) as follows:

$$\boldsymbol{y}_{t+1} = \tilde{\boldsymbol{\beta}}_0 + \tilde{B}_{\rm v}\boldsymbol{v}_t + \tilde{B}_{\rm y}\boldsymbol{y}_t, \tag{26}$$

where $\tilde{\beta}_{i0}$, $\tilde{\beta}_{{\rm v}ij}$, and $\tilde{\beta}_{{\rm y}ij}$ are the partial regression coefficients, $\tilde{\boldsymbol{\beta}}_0 \stackrel{\rm def}{=} {}^{\rm t}(\tilde{\beta}_{10}, \cdots, \tilde{\beta}_{d_{\rm x}0})$, $\tilde{\boldsymbol{\beta}}_{{\rm v}i} \stackrel{\rm def}{=} {}^{\rm t}(\tilde{\beta}_{{\rm v}i1}, \cdots, \tilde{\beta}_{{\rm v}id_{\rm v}})$, $\tilde{\boldsymbol{\beta}}_{{\rm y}_i} \stackrel{\rm def}{=} {}^{\rm t}(\tilde{\beta}_{{\rm y}_i1}, \cdots, \tilde{\beta}_{{\rm y}_id_{\rm y}})$, $\tilde{B}_{\rm v} \stackrel{\rm def}{=} {}^{\rm t}[\tilde{\boldsymbol{\beta}}_{{\rm v}1}, \cdots, \tilde{\boldsymbol{\beta}}_{{\rm v}d_{\rm y}}]$ is a $d_{\rm y} \times d_{\rm v}$ matrix, and $\tilde{B}_{\rm y} \stackrel{\rm def}{=} {}^{\rm t}[\tilde{\boldsymbol{\beta}}_{{\rm y}_1}, \cdots, \tilde{\boldsymbol{\beta}}_{{\rm y}_{d_{\rm y}}}]$ is a $d_{\rm y} \times d_{\rm y}$ matrix. By substituting Eq. (24) into Eq. (15), we obtain

$$\boldsymbol{y}_{t+1} = {}^{\rm t}M\boldsymbol{\beta}_0 + {}^{\rm t}MB_{\rm v}\boldsymbol{v}_t + {}^{\rm t}MB_{\rm x}M\boldsymbol{y}_t. \tag{27}$$

By comparing Eqs. (26) and (27), partial regression coefficients $\tilde{\beta}_{{\rm v}ij}$ is obtained by the following equation:

$$\tilde{B}_{\rm v} = {}^{\rm t}MB_{\rm v}. \tag{28}$$

By substituting Eq. (24) into Eq. (17), we can describe the multiple regression equation for r_{t+1} by using

$$\begin{aligned}r_{t+1} &= b_0 + \boldsymbol{b}_{\rm v}\boldsymbol{v}_t + {}^{\rm t}\boldsymbol{b}_{\rm x}M\boldsymbol{y}_t + {}^{\rm t}\boldsymbol{b}_{\rm x'}M\boldsymbol{y}_{t+1} \\ &= b_0 + \boldsymbol{b}_{\rm v}\boldsymbol{v}_t + {}^{\rm t}\tilde{\boldsymbol{b}_{\rm y}}\boldsymbol{y}_t + {}^{\rm t}\tilde{\boldsymbol{b}_{\rm y'}}\boldsymbol{y}_{t+1}.\end{aligned} \tag{29}$$

Here, ${}^{\rm t}\tilde{\boldsymbol{b}_{\rm y}} \stackrel{\rm def}{=} (\tilde{b_{\rm y}}_1, \cdots, \tilde{b_{\rm y}}_{d_{\rm y}}) = {}^{\rm t}\boldsymbol{b}_{\rm x}M$ and ${}^{\rm t}\tilde{\boldsymbol{b}_{\rm y'}} \stackrel{\rm def}{=} (\tilde{b_{\rm y'}}_1, \cdots, \tilde{b_{\rm y'}}_{d_{\rm y}}) = {}^{\rm t}\boldsymbol{b}_{\rm x'}M$. As a result, we obtain partial regression coefficients $\tilde{b_{\rm y}}_i$ and $\tilde{b_{\rm y'}}_i$ from

$$\begin{aligned}\tilde{\boldsymbol{b}_{\rm y}} &= {}^{\rm t}M\boldsymbol{b}_{\rm x}, \\ \tilde{\boldsymbol{b}_{\rm y'}} &= {}^{\rm t}M\boldsymbol{b}_{\rm x'}.\end{aligned} \tag{30}$$

Let $res_{{\rm yv}_{ij}}$ be the resolution of y_i for computing v_j and $res_{{\rm yu}_i}$ be the resolution of y_i for computing $\boldsymbol{u}$. In the same manner as in Eqs. (19) and (20), $res_{{\rm yu}_i}$ is determined as follows:

$$\begin{aligned}res_{{\rm yu}_i} &= \sum_{j=1}^{d_{\rm u}}(G_{\rm u}(res_{{\rm yv}i1}, \cdots, res_{{\rm yv}id_{\rm x}}))_j, \\ res_{{\rm yv}ij} &= \sigma(y_i)^{\tilde{\gamma}_1}D_{{\rm v}j}^{\tilde{\gamma}_2}|\tilde{\beta}_{{\rm v}ij}|^{\tilde{\gamma}_3}(|\tilde{b_{\rm y}}_i| + |\tilde{b_{\rm y'}}_i|)^{\tilde{\gamma}_4},\end{aligned} \tag{31}$$

where $\tilde{\gamma}_1, \cdots, \tilde{\gamma}_4$ are given constants, and $\sigma(y_i) = \lambda_i^{1/2}$.

3.6 State Transformation

Principal component vector $\boldsymbol{y}_t$ obtained by using Eq. (24) has a unique boundary because it is assumed that $\boldsymbol{x}_t \in \boldsymbol{D}_{\rm x}$, as mentioned in Sect. 3.1. However, $\boldsymbol{y}_t$ does not take all the values within the boundary. Thus, state vector $\boldsymbol{s}_t$ used by the actor-critic method is set by using the following equation so that it is distributed over the whole feature space and expresses the dynamics in $\boldsymbol{x}_t$ as accurately as possible:

$$s_{i;t} = \frac{D_{{\rm s}i}}{1 + \exp(-\dfrac{y_{i;t} - \bar{y}_i}{c_{\rm PN}\sigma(y_i)})} - \frac{D_{{\rm s}i}}{2} + \bar{s}_i, \tag{32}$$

where $d_{\rm s} = d_{\rm y}$, $\bar{s}_i = s_{{\rm min}i} + D_{{\rm s}i}/2$, and $c_{\rm PN}$ is set to 0.6 so that the first term on the right side of Eq. (32) is approximately equal to the probability distribution function of a normal distribution with mean $\bar{y}_i$ and standard deviation $\sigma(y_i)$.

3.7 Basis of Feature Space

Based on the resolution of y_i obtained by using Eq. (31), bases $\boldsymbol{\phi}_{\rm v}$, $\boldsymbol{\phi}_{\rm m}$, and $\boldsymbol{\phi}_\sigma$ are determined in this section. To simplify the algorithm, let $N_{\rm V}$, $N_{\rm m}$, and N_σ be set to N and let N be given in advance. Let $N_{{\rm V}i}$, $N_{{\rm m}i}$, and $N_{\sigma i}$ be set to N_i, let $\boldsymbol{\phi}_{\rm v}$, $\boldsymbol{\phi}_{\rm m}$, and $\boldsymbol{\phi}_\sigma$ be set to $\boldsymbol{\phi} \stackrel{\rm def}{=} {}^{\rm t}(\phi_0, \cdots, \phi_N)$, and let $\phi_i \stackrel{\rm def}{=} K(\boldsymbol{x}, \boldsymbol{k})$ as shown in Eq. (A·2) in Appendix A. Here, $K(\boldsymbol{x}, \boldsymbol{k})$ is a multi-dimensional orthonormal basis defined by Eq. (A·1), $\boldsymbol{k}$ is the index vector, and the relationship between i, the index of the basis, and $\boldsymbol{k}$ is obtained by Eq. (A·3).

Once N and $K(\boldsymbol{x}, \boldsymbol{k})$ are given, $\boldsymbol{\phi}$ is determined by setting N_i. As mentioned in Sect. 3.5, to obtain a large discount return, it is reasonable to assign a larger value to N_i if $res_{{\rm yu}_i}$ is large. Furthermore, Eq. (A·4), which shows the relationship between N and N_i, should hold. Thus, to satisfy these requirements, we determine N_i by the following equation:

$$N_i = {\rm int}(\alpha\, res_{{\rm yu}_i}), \tag{33}$$

where ${\rm int}(\cdot)$ denotes the integer part, and α is set so that $|N - \prod_{i=1}^{d_{\rm y}} (N_i + 1) - 1|$ is minimum for the given N and $res_{{\rm yu}_i}$.

Let $\boldsymbol{\phi}$ be the basis when the degree of expansion is N_j, and $\boldsymbol{\phi}'$ be the basis when the degree of expansion is N'_j. When the degree of expansion changes from N_j to N'_j, the basis also changes from $\boldsymbol{\phi}$ to $\boldsymbol{\phi}'$. Thus, new coefficients $\boldsymbol{c}'$, $\boldsymbol{\xi}'_{{\rm m}i'}$, and $\boldsymbol{\xi}'_{\sigma i'}$ for the new bases have to take the values of the old ones, $\boldsymbol{c}$, $\boldsymbol{\xi}_{{\rm m}i}$, and $\boldsymbol{\xi}_{\sigma i}$, for learning to proceed smoothly. That is, $c'_{i'}$ is set to c_i, $\boldsymbol{\xi}'_{{\rm m}i'}$ is set to $\boldsymbol{\xi}_{{\rm m}i}$, and $\boldsymbol{\xi}'_{\sigma i'}$ is set to $\boldsymbol{\xi}_{\sigma i}$, if $\phi'_{i'}$ is equal to ϕ_i. Further details are described in Appendix A.

4. Expansion to Multiple Agents

Let us consider an environment that L agents control, and in which the agents cannot always observe all the state variables and the control inputs. In this section, the agent described by Algorithm 2 is extended to multiple agents. Let $\hat{x}_i$ be the ith state variable of the environment, $\hat{\boldsymbol{x}} \stackrel{\rm def}{=} {}^{\rm t}(\hat{x}_1, \cdots, \hat{x}_{\hat{d}_{\rm x}}) \in \hat{\boldsymbol{D}}_{\rm x}$, $\hat{\boldsymbol{D}}_{\rm x} \stackrel{\rm def}{=} \{\hat{x}_i | \hat{x}_{{\rm min}i} \le \hat{x}_i \le \hat{x}_{{\rm min}i} + \hat{D}_{{\rm x}i}, 1 \le i \le \hat{d}_{\rm x}\}$, $\hat{v}_i$ be the ith control input of the environment, $\hat{\boldsymbol{v}} \stackrel{\rm def}{=} {}^{\rm t}(\hat{v}_1, \cdots, \hat{v}_{\hat{d}_{\rm v}})$, $\hat{\boldsymbol{D}}_{\rm v} \stackrel{\rm def}{=} \{\hat{v}_i | \hat{v_{\rm min}}_i \le \hat{v}_i \le \hat{v_{\rm min}}_i + \hat{D}_{{\rm v}i}, 1 \le i \le \hat{d}_{\rm v}\}$, $r^{(\ell)}$ be the immediate reward that the ℓth agent receives, $\boldsymbol{x}_t^{(\ell)}$ be the subset of $\hat{\boldsymbol{x}}_t$ that the ℓth agent can observe, $\boldsymbol{v}_t^{(\ell)}$ be the subset of $\hat{\boldsymbol{v}}_t$ that the ℓth agent can observe, and $\boldsymbol{u}_t^{(\ell)}$ be the action determined by the ℓth agent. To express the relationship between $\boldsymbol{x}_t^{(\ell)}$ and $\hat{\boldsymbol{x}}_t$, and that between $\boldsymbol{v}_t^{(\ell)}$ and $\hat{\boldsymbol{v}}_t$, let us introduce state observation matrix ${G_{\rm s}}^{(\ell)}$ and action observation matrix ${G_{\rm v}}^{(\ell)}$. By using ${G_{\rm s}}^{(\ell)}$ and ${G_{\rm v}}^{(\ell)}$, the ℓth agent observes $\boldsymbol{x}_t^{(\ell)}$ and $\boldsymbol{v}_t^{(\ell)}$ as follows:

$$\begin{aligned} \boldsymbol{x}_t^{(\ell)} &= {G_{\rm s}}^{(\ell)} \hat{\boldsymbol{x}}_t, \\ \boldsymbol{v}_t^{(\ell)} &= {G_{\rm v}}^{(\ell)} \hat{\boldsymbol{v}}_t. \end{aligned} \tag{34}$$

Actions $\boldsymbol{u}_t^{(1)}, \cdots, \boldsymbol{u}_t^{(L)}$ are combined into $\hat{\boldsymbol{v}}_t$ by

$$\hat{\boldsymbol{v}}_t = {G_{\rm c}}^{\rm t}({}^{\rm t}\boldsymbol{u}_t^{(1)}, \cdots, {}^{\rm t}\boldsymbol{u}_t^{(L)}) + \boldsymbol{g}_{\rm c}. \tag{35}$$

Here, $G_{\rm c}$ and $\boldsymbol{g}_{\rm c}$ are determined from $\hat{v_{\rm min}}_i$, $\hat{D}_{{\rm v}i}$, ${u_{\rm min}}_i^{(\ell)}$, and $D_{{\rm u}_i}^{(\ell)}$. Immediate reward $r_t^{(\ell)}$ for the ℓth agent is set based on r_t.

The structure and algorithm of reinforcement learning with state space compression for multiple agents are shown in Fig. 2 and given by Algorithm 4, respectively.

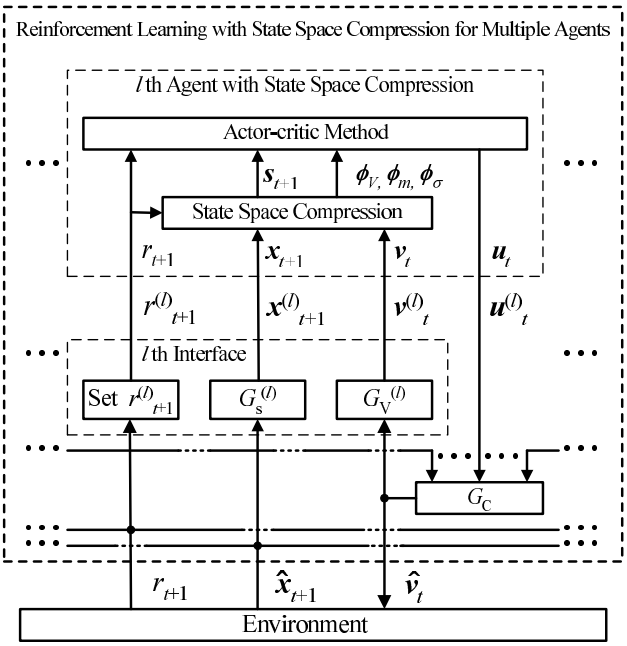

Fig. 2 Structure of reinforcement learning with state space compression for multiple agents.

Algorithm 4: Reinforcement learning with state space compression for multiple-agents.

(4-1) Initialization-1: Set $\boldsymbol{c}_0^{(\ell)}$, $\boldsymbol{\xi}_{{\rm m}0}^{(\ell)}$, $\boldsymbol{\xi}_{\sigma 0}^{(\ell)}$, ${n_{\rm MA}}^{(\ell)} = 0$, and ${trg_{\rm MA}}^{(\ell)} = 0$ for $1 \le \ell \le L$.

(4-2) Initialization-2: Set $t = 0$, $\hat{\boldsymbol{x}}_0$, and $\hat{\boldsymbol{v}}_0$.

(4-3) Environment: Receive $\hat{\boldsymbol{v}}_t$, and output $\hat{\boldsymbol{x}}_{t+1}$ to the interface.

(4-4) Interface: Set $\boldsymbol{x}_{t+1}^{(\ell)}$, $\boldsymbol{x}_t^{(\ell)}$, and $\boldsymbol{v}_t^{(\ell)}$ by using Eq. (34) for $1 \le \ell \le L$, and set $r_{t+1}^{(\ell)}$ for $1 \le \ell \le L$.

(4-5) Agent with State Space Compression: Compute $\boldsymbol{u}_{t+1}^{(\ell)}$ by using Step (2-4) in Algorithm 2 for $1 \le \ell \le L$.

(4-6) $G_{\rm c}$: Compute $\hat{\boldsymbol{v}}_{t+1}$ by using Eq. (35).

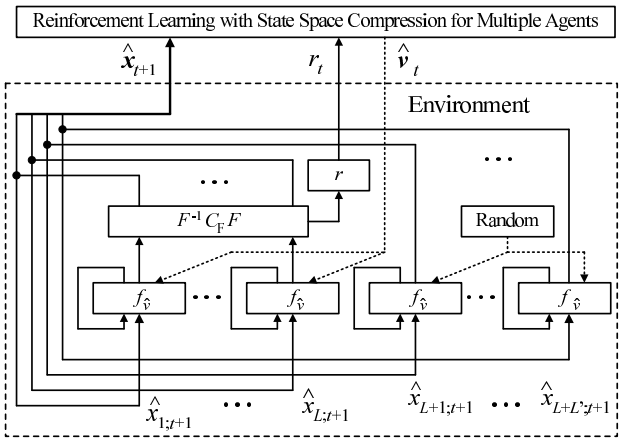

Fig. 3 Simple environment: Linear combination of logistic maps.

(4-7) Set $t = t + 1$. If an episode finished, go to Step (4-2), else go to Step (4-3).

5. Performance Evaluation

Consider the simple example environment shown in Fig. 3, which is a linear combination of $L+L'$ logistic maps [8]. Here, ${d_{\rm u}}^{(\ell)} = 1$ for $1 \le \ell \le L$, $\hat{d}_{\rm v} = d_{\rm v} = L$, $\hat{d}_{\rm x} = d_{\rm x} = L + L'$, $0 \le \hat{x}_{\ell;t} \le 1$ for $1 \le \ell \le L + L'$, I_ℓ is the $(\ell \times \ell)$-identity matrix, ${G_{\rm s}}^{(\ell)} = I_{L+L'}$, ${G_{\rm v}}^{(\ell)} = I_L$ for $1 \le \ell \le L$, i.e., $\boldsymbol{x}_t^{(\ell)} = \hat{\boldsymbol{x}}_t$ and $\boldsymbol{v}_t^{(\ell)} = \hat{\boldsymbol{v}}_t$ for $1 \le \ell \le L$, $G_{\rm c} = I_L$ and $\boldsymbol{g}_{\rm c} = 0$, i.e., $\hat{\boldsymbol{v}}_t = {}^{\rm t}({u_1^{(1)}}_t, \cdots, {u_1^{(L)}}_t)$, and the logistic map is defined by the following equation [8]:

$$\begin{aligned} x' &= f_v(x) \\ &\stackrel{\rm def}{=} vx(1-x). \end{aligned} \tag{36}$$

States $\hat{x}'_{1;t}, \cdots, \hat{x}'_{L;t}$ interact with each other, and $\hat{\boldsymbol{x}}_{t+1}$ is obtained by the following equation:

$$\begin{aligned} &\hat{x}'_{\ell;t} = \overbrace{f_{\hat{v}_{\ell;t}} \cdots f_{\hat{v}_{\ell;t}}}^{n_{\rm map}\text{times}}(\hat{x}_{\ell;t}) \quad \text{for} \quad 1 \le \ell \le L + L', \\ &\begin{cases} {}^{\rm t}(\hat{x}_{1;t+1}, \cdots, \hat{x}_{L;t+1}) \\ \qquad = F^{-1} C_{\rm F} F\, {}^{\rm t}(\hat{x}'_{1;t}, \cdots, \hat{x}'_{L;t}), \\ {}^{\rm t}(\hat{x}_{L+1;t+1}, \cdots, \hat{x}_{L+L';t+1}) \\ \qquad = {}^{\rm t}(\hat{x}'_{L+1;t}, \cdots, \hat{x}'_{L+L';t}), \end{cases} \end{aligned} \tag{37}$$

where $n_{\rm map} = 3$, $3.4 \le \hat{v}_{\ell;t} \le 4$ for $1 \le \ell \le L+L'$, F is a matrix that performs discrete Fourier transform, and $C_{\rm F} \stackrel{\rm def}{=} {\rm diag}[1.0, 0.9, 0.81, 0.73, 0.66, 0.59, 0.53, 0.48, \cdots]$, which denotes a low pass filter that reduces the power spectrum of $\hat{x}'_{1;t}, \cdots, \hat{x}'_{L;t}$ as the frequency increases.

The ℓth agent determines action ${u_1^{(\ell)}}_t = \hat{v}_{\ell;t}$ by using the reinforcement learning described in Algorithm 4, and $\hat{v}_{L+1;t}, \cdots, \hat{v}_{L+L';t}$ are set so that they independently obey a uniform distribution. Let $spc_{0;t}, \cdots, spc_{L-1;t}$ be the power spectrum of $\hat{x}_{1;t}, \cdots, \hat{x}_{L;t}$, and immediate reward $r_{t+1}^{(\ell)}$ for $1 \le \ell \le L$ be equal to r_{t+1} defined by

$$r_{t+1} \stackrel{\rm def}{=} (\Delta spc_{k;t+1})^2 - \sum_{j=1, j\neq k}^{L} (\Delta spc_{j;t+1})^2, \tag{38}$$

where the power spectrum denotes the square of the Fourier coefficients obtained by $F\,{}^{\rm t}(\hat{x}'_{1;t}, \cdots, \hat{x}'_{L;t})$, $\Delta spc_{j;t+1} \stackrel{\rm def}{=} spc_{j;t+1} - spc_{j;t}$, k is set to 2. Thus, r_{t+1} increases when $\hat{x}_{1;t}, \cdots, \hat{x}_{L;t}$ move so that the motion in $spc_{2;t}$ is large. Because $\hat{x}_{L+1;t}, \cdots, \hat{x}_{L+L';t}$ are not related to the immediate reward, they are disturbance states that disturb the 1st to the Lth agent.

The effectiveness of the state space compression algorithm described in Sect. 3 and the ability of Algorithm 4 to construct a feature space without being disturbed by the disturbance states were demonstrated by evaluating Algorithm 4 for various values for the degree of expansion, N, and the number of disturbance states, L', and for several combinations of $\boldsymbol{\gamma} \stackrel{\rm def}{=} (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$ and $\tilde{\boldsymbol{\gamma}} \stackrel{\rm def}{=} (\tilde{\gamma}_1, \tilde{\gamma}_2, \tilde{\gamma}_3, \tilde{\gamma}_4)$. Note that the parameters for each agent in Algorithm 4 have the same values, and superscript (ℓ) for variables and parameters is omitted in the following to simplify the explanation. The combinations of $\boldsymbol{\gamma}$ and $\tilde{\boldsymbol{\gamma}}$ evaluated were determined as follows. (1) $\tilde{\gamma}_1$: It is reasonable to construct a feature space based on the range of the ith principal component, y_i. In other words, degree of expansion N_i for y_i should be proportional to $\sigma(y_i)$. Thus, $\tilde{\gamma}_1$ is set to 1. (2) γ_2: Because it is clear that the difference in $D_{{\rm v}j}$ should be normalized, γ_2 is set to 1. Note that γ_2 does not affect the performance shown below, because $d_{\rm u} = 1$ in the environment described in Eq. (37). (3) γ_3 and γ_4: It is important to distinguish the states that contribute to the reward from the states that disturb the control algorithm. To do this, the effect of $v_{j;t}$ on $x_{i;t+1}$ and those of $x_{i;t}$ and $x_{i;t+1}$ on r_{t+1} are evaluated. The former is expressed by $|\beta_{{\rm v}ij}|^{\gamma_3}$, and the latter is expressed by $(|b_{{\rm x}i}| + |b_{{\rm x}'i}|)^{\gamma_4}$. Thus, Algorithm 4 is evaluated for $\gamma_3 = 0$ or 1, and $\gamma_4 = 0$ or 1. (4) γ_1, $\tilde{\gamma}_2$, $\tilde{\gamma}_3$, and $\tilde{\gamma}_4$: Because $\sigma(x_i)$ and $\sigma(y_i)$ express the same property (i.e., the standard deviation in the state) of the environment, γ_1 is set to 0 so that an agent does not take the standard deviation into account more than once. For the same reason, $\tilde{\gamma}_2$, $\tilde{\gamma}_3$, and $\tilde{\gamma}_4$ are set to 0. Therefore, $\tilde{\boldsymbol{\gamma}}$ is set to $(1, 0, 0, 0)$, and $\boldsymbol{\gamma}$ is set to $(0, 0, 0, 0)$, $(0, 1, 1, 0)$, $(0, 1, 0, 1)$, or $(0, 1, 1, 1)$, where the motions in Algorithm 4 for $(0, 0, 0, 0)$ and $(0, 1, 0, 0)$ are the same in the environment of Eq. (37), and $(0, 0, 0, 0)$ means that $res_{\rm xu1} = , \cdots, = res_{{\rm xu}L+L'} = 1$. The other parameters for reinforcement learning are set as follows: $T_{\rm S} = 5 \times 10^6$, $T_{\rm RB} = 10^3$, $\nu_{\rm Tr} = 0.3$, $\nu_{\rm DR} = 0.9$, and $\zeta_{\rm v} = \zeta_{\rm m} = \zeta_\sigma = 10^{-4}$.

The effect of N and L on Algorithm 4 was estimated by evaluating the change in immediate reward r_t when $\boldsymbol{\gamma} = (0, 0, 0, 0)$ and $L' = 0$. The performance of

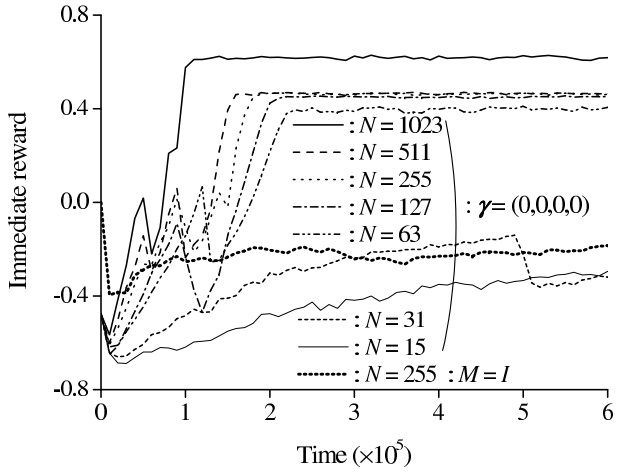

Fig. 4 Change in immediate reward for various values of degree of expansion after starting reinforcement learning ($L = 8$).

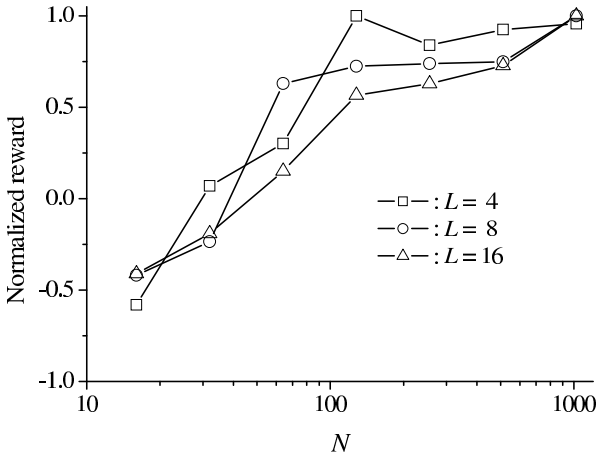

Fig. 5 The effect of N and L on normalized reward.

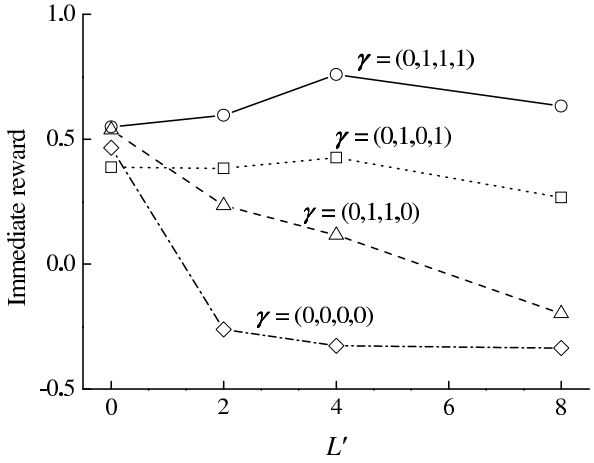

Fig. 6 Effect of number of disturbance states on immediate reward ($N = 255$ and $L = 8$).

the state space compression in Algorithm 4 was demonstrated by also evaluating an algorithm that does not use the state space compression, in which $M = I$ and N_i was given in advance ($N_1 =, \cdots, = N_L = 1$; i.e., $N = 255$ when $L = 8$). Figure 4 shows the change in r_t for $L = 8$ after starting reinforcement learning; that is, the time in Figure 4 indicates $t - T_{\rm S}$. The figure shows that Algorithm 4 with $\boldsymbol{\gamma} = (0, 0, 0, 0)$ provides a sufficiently large r_t when N is at least 63. On the other hand, Algorithm 4 without state space compression (i.e., $M = I$) provides a lower value of r_t although N (= 255) is larger than 63. Therefore, the state space compression method described in Sect. 3 is effective. Figure 5 shows the effect of N and L on r_t, where r_t is normalized by the maximum value of r_t for each value of L. The changes in the normalized reward for $L = 4, 8$, and 16 are almost the same as shown in this figure. This means that the performance is not always affected by L, the dimension of the system. The factors affecting the performance could be the complexity of the environment defined by $n_{\rm map}$ and $C_{\rm F}$.

The effect of the number of disturbance states on immediate reward r_t is shown in Fig. 6, where $N = 255$ and $L = 8$. This figure shows that the effect of γ_3 and γ_4 becomes noticeable as L', the number of disturbance states, increases, and that the largest r_t is obtained when $\gamma_3 = \gamma_4 = 1$ for each value of L'. Thus, $|\beta_{{\rm v}_{ij}}|$ and $|b_{{\rm x}i}| + |b_{{\rm x}'i}|$ are effective to remove the effect of the disturbance states, and we can conclude that Algorithm 4 with $\boldsymbol{\gamma} = (0, 1, 1, 1)$ constructs a feature space effectively and controls the environment without suffering the disturbance states.

The motion in state $\hat{x}_{\ell;t}$ was demonstrated by measuring the power spectrum and the change in state $\hat{x}_{1;t}, \cdots, \hat{x}_{L;t}$ for $M = I$ and $\boldsymbol{\gamma} = (0, 1, 1, 1)$ after r_t was stable and saturated. The results are shown in Figs. 7 to 9 for $N = 255$, $L = 8$, and $L' = 0$. The origin of the time in Figs. 8 and 9 indicates the time when measurement started. These figures show that spc_2 in Eq. (38) is extracted and Algorithm 4 works well when $\boldsymbol{\gamma} = (0, 1, 1, 1)$, although $\hat{x}_{1;t}, \cdots, \hat{x}_{L;t}$ moves almost randomly when $M = I$.

All combinations of $\boldsymbol{\gamma}$ and $\tilde{\boldsymbol{\gamma}}$ were not evaluated, and it was assumed that all states and control inputs in the environment could be observed. To better clarify the effectiveness of Algorithm 4, a more extensive investigation will be conducted.

6. Conclusion

A state space compression method based on multivariate analysis was developed and applied to reinforcement learning for high-dimensional continuous state spaces. It can autonomously construct a feature space without preliminary knowledge of the environment. An example synchronization problem for multiple logistic maps was solved, demonstrating that the state space compression method exhibits high performance without suffering from disturbance states. Because the basis of the

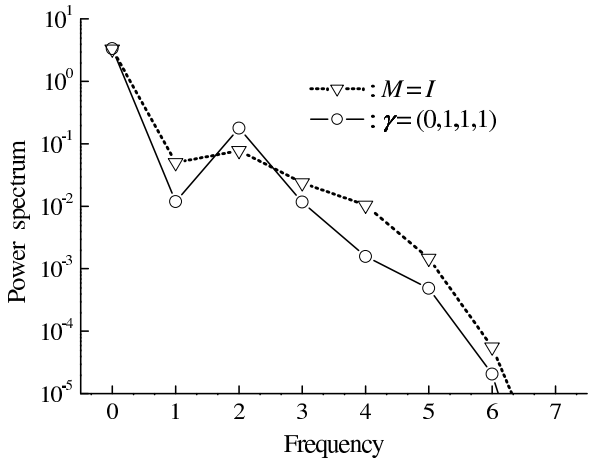

Fig. 7 Power spectrum of $\hat{x}_1, \cdots, \hat{x}_8$ ($N = 255$, $L = 8$, and $L' = 0$).

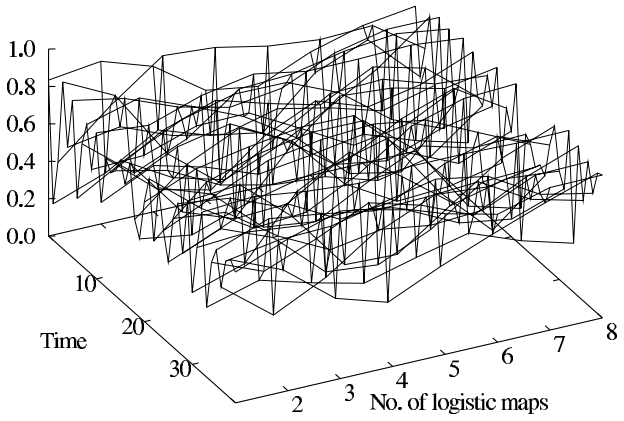

Fig. 8 Motion in $\hat{x}_1, \cdots, \hat{x}_8$ for $M = I$ ($N = 255$, $L = 8$, and $L' = 0$).

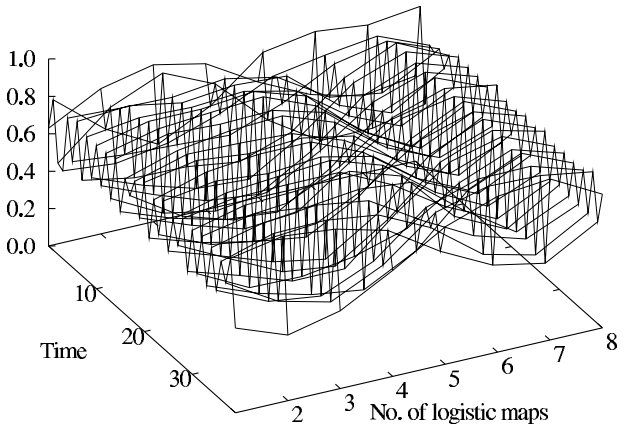

Fig. 9 Motion in $\hat{x}_1, \cdots, \hat{x}_8$ for $\boldsymbol{\gamma} = (0, 1, 1, 1)$ ($N = 255$, $L = 8$, and $L' = 0$).

feature space is orthogonal, the mathematical structure of the system in the feature space is clear, and it is possible to mathematically analyze the dynamics and stability in a moment vector space. The mathematical analysis of the dynamics will be reported in a future.

Appendix A: Orthonormal Basis for Function Approximation

Let $\boldsymbol{x} \stackrel{\text{def}}{=} {}^{\text{t}}(x_1, \cdots, x_{d_{\text{x}}})$ and $\boldsymbol{\mathcal{D}}_{\mathbf{x}} \stackrel{\text{def}}{=} \{x_j | x_{\min j} \leq x_j \leq x_{\min j} + D_{\text{x}j}, 1 \leq j \leq d_{\text{x}}\}$. In this appendix, the orthonormal bases [12] for $\boldsymbol{x} \in \boldsymbol{\mathcal{D}}_{\mathbf{x}}$ are summarized.

Let $\boldsymbol{k} \stackrel{\text{def}}{=} {}^{\text{t}}(k_1, \cdots, k_{d_{\text{x}}})$ be the index vector of the Fourier coefficient and $h(\boldsymbol{k})$ be the Fourier coefficient for index vector $\boldsymbol{k}$. The Fourier series expansion for function $f(\boldsymbol{x})$ is defined by the following equation [12]:

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \boldsymbol{\mathcal{Z}}} h(\boldsymbol{k}) K(\boldsymbol{x}, \boldsymbol{k}),$$
$$h(\boldsymbol{k}) \stackrel{\text{def}}{=} \int_{\boldsymbol{\mathcal{D}}_{\mathbf{x}}} f(\boldsymbol{x}) K^*(\boldsymbol{x}, \boldsymbol{k}) \mathrm{d}\boldsymbol{x},$$

where $\boldsymbol{\mathcal{Z}} \stackrel{\text{def}}{=} \{k_j | 0 \leq k_j \leq N_j,\ 1 \leq j \leq d_{\text{x}}\}$, N_j is the degree of expansion of x_j, $h(\boldsymbol{k})$ is the Fourier coefficient, superscript $*$ denotes a complex conjugate, and $\{K(\boldsymbol{x}, \boldsymbol{k})\}$ is a multi-dimensional orthonormal basis defined by the following equation:

$$K(\boldsymbol{x}, \boldsymbol{k}) \stackrel{\text{def}}{=} \prod_{j=1}^{d_{\text{x}}} K_j(x_j, k_j). \tag{A·1}$$

Here, $\{K_j(x_j, k_j)\}$ is one-dimensional orthonormal basis, $K_j(x_j, k_j)$ is set to a trigonometrical function defined by

$$K_j(x_j, k_j) \stackrel{\text{def}}{=}$$
$$\begin{cases} \sqrt{\dfrac{1}{D_{\text{x}j}}} & \text{for } k_j = 0 \\ \sqrt{\dfrac{2}{D_{\text{x}j}}} \sin\left(\dfrac{k_j + 1}{2} \omega_{0j} (x_j - x_{\min j})\right) & \text{for } k_j \in \boldsymbol{\mathcal{N}}_{\mathbf{o}}, \\ \sqrt{\dfrac{2}{D_{\text{x}j}}} \cos\left(\dfrac{k_j}{2} \omega_{0j} (x_j - x_{\min j})\right) & \text{for } k_j \in \boldsymbol{\mathcal{N}}_{\mathbf{e}} \end{cases}$$

$\omega_{0j} \stackrel{\text{def}}{=} 2\pi / D_{\text{x}j}$, $\boldsymbol{\mathcal{N}}_{\mathbf{e}}$ denotes a set of even numbers, and $\boldsymbol{\mathcal{N}}_{\mathbf{o}}$ denotes a set of odd numbers.

Let us define basis $\boldsymbol{\phi}(\boldsymbol{x}) = {}^{\text{t}}(\phi_0(\boldsymbol{x}), \cdots, \phi_N(\boldsymbol{x}))$ as

$$\phi_i(\boldsymbol{x}) \stackrel{\text{def}}{=} K(\boldsymbol{x}, \boldsymbol{k}), \tag{A·2}$$

where i is the index of the basis, N is the degree of expansion of $\boldsymbol{x}$, and $N+1$ is the dimension of the feature space with the basis. When $\boldsymbol{\mathcal{Z}} \stackrel{\text{def}}{=} \{k_j | 0 \leq k_j \leq N_j, 1 \leq j \leq d_{\text{x}}\}$, that is, $\boldsymbol{\mathcal{Z}}$ is a cube, the index of the basis is obtained from index vector $\boldsymbol{k}$ as follows:

$$i = \sum_{j=1}^{d_{\text{x}}} k_j \prod_{\nu=j+1}^{d_{\text{x}}} N_\nu. \tag{A·3}$$

From the degree of expansion of x_j, we can obtain the degree of expansion of $\boldsymbol{x}$ as follows:

$$N = \prod_{j=1}^{d_{\mathrm{x}}}(N_j + 1) - 1. \quad (\mathrm{A}\cdot 4)$$

Given that the degree of expansion of x_j is N_j, suppose that the basis, the index of the basis, and the domain of $\boldsymbol{k}$ are given by $\boldsymbol{\phi}$, $i(\boldsymbol{k})$, and $\boldsymbol{\mathcal{Z}}$, respectively. When the degree of expansion of x_j is N'_j, $\boldsymbol{\phi}'$, $i'(\boldsymbol{k})$, and $\boldsymbol{\mathcal{Z}}'$ are also given. As mentioned in Sect. 3.7, when the degree of expansion changes from N_j to N'_j, the new coefficients, $c'_{i'(k)}$, $\boldsymbol{\xi}'_{\mathrm{m}i'(k)}$, and $\boldsymbol{\xi}'_{\sigma i'(k)}$, for the new bases take the values of the old ones, $c_{i(k)}$, $\boldsymbol{\xi}_{\mathrm{m}i(k)}$, and $\boldsymbol{\xi}_{\sigma i(k)}$, as follows:

$$\begin{aligned} c'_{i'(k)} &= c_{i(k)} \quad \text{for} \quad \forall \boldsymbol{k} \in \boldsymbol{\mathcal{Z}}', \\ \boldsymbol{\xi}'_{\mathrm{m}i'(k)} &= \boldsymbol{\xi}_{\mathrm{m}i(k)} \quad \text{for} \quad \forall \boldsymbol{k} \in \boldsymbol{\mathcal{Z}}', \\ \boldsymbol{\xi}'_{\sigma i'(k)} &= \boldsymbol{\xi}_{\sigma i(k)} \quad \text{for} \quad \forall \boldsymbol{k} \in \boldsymbol{\mathcal{Z}}'. \end{aligned} \quad (\mathrm{A}\cdot 5)$$

If $\boldsymbol{k} \in \boldsymbol{\mathcal{Z}}'$ and $\boldsymbol{k} \notin \boldsymbol{\mathcal{Z}}$, that is, there is no $\phi_{i(k)}$ corresponding to $\phi'_{i'(k)}$, $c_{\prime i'(k)}$, $\xi'_{\mathrm{m}i'(k)j}$, and $\xi'_{\sigma i'(k)j}$ are set to 0 .

Appendix B: TD Learning Based on Steepest Descent Method

TD learning [1] based on the steepest descent method is summarized in this appendix. Let $\boldsymbol{x} \stackrel{\mathrm{def}}{=} {}^{\mathrm{t}}(x_1, \cdots, x_{d_{\mathrm{x}}})$, $\boldsymbol{\mathcal{D}}_{\mathbf{x}} \stackrel{\mathrm{def}}{=} \{x_i | x_{\mathrm{min}i} \le x_i \le x_{\mathrm{min}i} + D_{\mathrm{x}i}, 1 \le i \le d_{\mathrm{x}}\}$, $p(\boldsymbol{x})$ be a PDF of $\boldsymbol{x}$, $g(\boldsymbol{x}, \boldsymbol{c})$ be an approximation of function $g(\boldsymbol{x})$, and $\boldsymbol{c}$ be the parameter used by the approximation. Parameter $\boldsymbol{c}$ is determined so as to minimize $MSE(\boldsymbol{c})$ defined by

$$MSE(\boldsymbol{c}) \stackrel{\mathrm{def}}{=} \int_{\boldsymbol{x} \in \boldsymbol{\mathcal{D}}_{\mathbf{x}}} p(\boldsymbol{x})(g(\boldsymbol{x}) - g(\boldsymbol{x}, \boldsymbol{c}))^2 \mathrm{d}\boldsymbol{x}. \quad (\mathrm{A}\cdot 6)$$

Let us now derive parameter $\boldsymbol{c}$ by using the following steepest descent method [13].

Algorithm 5: Steepest Descent Method.

(1) Set initial value $\boldsymbol{c}_0$ and step size $\zeta > 0$.

(2) Set number of iterations $t = 0$.

(3) $\boldsymbol{c}_{t+1} = \boldsymbol{c}_t - \zeta \nabla_{\boldsymbol{c}} MSE(\boldsymbol{c}_t)$.

(4) $t = t + 1$.

(5) Go to Step (3).

Here, $\boldsymbol{c}_t$ is the value of vector $\boldsymbol{c}$ at t.

To realize Step (3) in Algorithm 5, let us make the following assumption.

Assumption 1: PDF $p(\boldsymbol{x})$ is the same as the PDF when Algorithm 5 has been performed.

By applying Assumption 1 and Eq. (A·6) to Step (3) in Algorithm 5, and by replacing ζ with $\zeta/2$, Step (3) in Algorithm 5 can be described as follows:

$$\begin{aligned} \boldsymbol{c}_{t+1} &= \boldsymbol{c}_t - (\zeta/2)\nabla_{\boldsymbol{c}} MSE(\boldsymbol{c}_t) \\ &= \boldsymbol{c}_t - (\zeta/2)\nabla_{\boldsymbol{c}}(g(\boldsymbol{x}_t) - g(\boldsymbol{x}_t, \boldsymbol{c}_t))^2 \\ &= \boldsymbol{c}_t + \zeta(g(\boldsymbol{x}_t) - g(\boldsymbol{x}_t, \boldsymbol{c}_t))\nabla_{\boldsymbol{c}} g(\boldsymbol{x}_t, \boldsymbol{c}_t). \end{aligned} \quad (\mathrm{A}\cdot 7)$$

Let $\hat{g}(\boldsymbol{x})$ be the estimated value of $g(\boldsymbol{x})$. When $g(\boldsymbol{x}_t)$ is unknown but $\hat{g}(\boldsymbol{x})$ can be obtained, Step (3) of Algorithm 5 is obtained by replacing $g(\boldsymbol{x})$ with $\hat{g}(\boldsymbol{x})$ in Eq. (A·7) as follows:

$$\boldsymbol{c}_{t+1} = \boldsymbol{c}_t + \zeta(\hat{g}(\boldsymbol{x}_t) - g(\boldsymbol{x}_t, \boldsymbol{c}_t))\nabla_{\boldsymbol{c}} g(\boldsymbol{x}_t, \boldsymbol{c}_t). \quad (\mathrm{A}\cdot 8)$$

Consider that $g(\boldsymbol{x}_t)$ is unknown and that we cannot obtain $\hat{g}(\boldsymbol{x})$; that is, we can obtain only an estimated value, $\tilde{g}(\boldsymbol{x})$, of $g(\boldsymbol{x})$ with unknown accuracy. In this case, introducing TD error $\varepsilon_{\mathrm{TD}\,t}$ into Eq. (A·7) and replacing $g(\boldsymbol{x})$ with $\tilde{g}(\boldsymbol{x})$ enable Step (3) of Algorithm 5 to be expressed by the following equation:

$$\boldsymbol{c}_{t+1} = \boldsymbol{c}_t + \zeta\varepsilon_{\mathrm{TD}\,t}(\tilde{g}(\boldsymbol{x}_t) - g(\boldsymbol{x}_t, \boldsymbol{c}_t))\nabla_{\boldsymbol{c}} g(\boldsymbol{x}_t, \boldsymbol{c}_t). \quad (\mathrm{A}\cdot 9)$$

References

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning, MIT Press, USA, 1998.

[2] Baird, L. C. and Klopf, A. H., "Reinforcement learning with high-dimensional, continuous actions," Technical Report WL-TR-93-1147, 1993.

[3] W. T. B. Uther and M. M. Veloso, "Tree based discretization for continuous state space reinforcement learning,", Proceedings of AAAI-98, pp. 769–774, July, 1998.

[4] A. J. Smith, "The applications of the self-organising map to reinforcement learning," Neural Networks archive, vol, 15, Issue 8-9 , pp. 1107–1124, Oct. 2002.

[5] T. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," Journal of Artificial Intelligence Research, vol. 13, pp. 227–303, 2000.

[6] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," Discrete event dynamic systems, vol. 13, Issue 4, pp. 341–379, Oct., 2003.

[7] H. Satoh, "Approximation and Analysis of Non-linear Equations Based on Moment Vector Equation," Tech. Rept. IEICE. NLP2005-1, pp. 1-6, Jan. 2005.

[8] E. Ott, Chaos in Dynamical Systems Second edition, Cambridge, UK, 2002.

[9] I. H. Witten, "An adaptive optimal controller for discrete-time Markov environments," Information and Control, vol. 34, pp. 286–295, 1977.

[10] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike elements that can solve difficult learning control problems," IEEE Trans. on Systems Man and Cybernetics, vol. 13, pp. 835–846, 1983.

[11] R. A. Johnson, D. W. Wichern, Applied Multivariate Statistical Analysis, 5th ed, Pearson Education (Prentice Hall USA), 2001.

[12] I. N. Bronshtein and K. A. Semendyayev, Handbook of Mathematics, Springer-Verlag, Berlin, 1997.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C 2nd Edition, Cambridge University Press, UK, 1992.