Control and Anti-control of Chaos Based on the Moving Largest Lyapunov Exponent Using Reinforcement Learning

Yanyan Han

Northwestern Polytechnical University

Jianpeng Ding

Northwestern Polytechnical University

Lin Du

Northwestern Polytechnical University

Youming Lei (✉ leiyuming@nwpu.edu.cn)

Northwestern Polytechnical University <https://orcid.org/0000-0002-8078-1067>

Original Research

Keywords: Control of chaos, Anti-control of chaos, Reinforcement learning, Moving largest Lyapunov exponent

Posted Date: February 4th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-176868/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Control and Anti-control of Chaos Based on the Moving Largest Lyapunov Exponent Using Reinforcement Learning

Yanyan Han · Jianpeng Ding · Lin Du · Youming Lei*

Abstract In this work, we propose a method of control and anti-control of chaos based on the moving largest Lyapunov exponent using reinforcement learning. In this method, we design a reward function for the reinforcement learning according to the moving largest Lyapunov exponent, which is similar to the moving average but computes the corresponding largest Lyapunov exponent using a recently updated time series with a fixed, short length. We adopt the density peaks-based clustering algorithm to determine a linear region of the average divergence index so that we can obtain the largest Lyapunov exponent of the small data set by fitting the slope of the linear region. We show that the proposed method is fast and easy to implement through controlling and anti-controlling typical systems such as the Henon map and Lorenz system.

Keywords Control of chaos · Anti-control of chaos · Reinforcement learning · Moving largest Lyapunov exponent

1 Introduction

Chaos is a ubiquitous phenomenon in nature, with randomness, local instability, sensitivity on initial conditions, long-term unpredictability and global boundedness. On the one hand, the characteristics of chaos often make it harmful in real-world systems. For example, chaotic traffic with local instability and long-term unpredictability can lead to traffic accidents in a macroscopic chaotic traffic flow model, which is dangerous for driving [1]. The existence of chaotic dynamics explains the abnormal deaths of fish and birds in the eco-epidemiological model on the Salton Sea, which should be avoided

Y. Han · J. Ding · L. Du · Y. Lei (✉)

School of Mathematics and Statistics, Northwestern Polytechnical University, Xi'an 710072, PR China

e-mail: leiyuming@nwpu.edu.cn

Y. Han e-mail: han_yanyan@mail.nwpu.edu.cn

J. Ding e-mail: dingjianpeng@mail.nwpu.edu.cn

L. Du e-mail: lindu@nwpu.edu.cn

L. Du · Y. Lei

Ministry of Industry and Information Technology Key Laboratory of Dynamics and Control of Complex Systems, Northwestern Polytechnical University, Xi'an, 710072, China

for ecological sustainability of the Sea [2]. Chaotic motions cause the unpredictability and high risk in asteroid explorations [3]. The emergence of a Shilnikov chaotic attractor in the economy model with financial bubbles in the banking sector may cause a financial crisis [4]. For the above and similar circumstances, chaos is harmful and should be suppressed [5]. In 1990, Ott, Grebogi and Yorke proposed an effective method of controlling chaos, called the OGY method [6]. They chose one of unstable orbits embedded in a chaotic attractor, which can improve the system performance, and then stabilized the system to the chosen orbit by using small but elaborately selected parameter perturbations. Their method can be applied to experimental situations in virtue of the technique of delay coordinate embedding. Subsequently, a large number of methods on control of chaos have been developed, such as the delayed feedback control [7-10], the open-plus-closed-loop control [11], and the adaptive control [12]. Unfortunately, most approaches usually rely on hypothetical knowledge or analytical descriptions of the system dynamics, which are difficult to obtain in practical applications. Based on reinforcement learning with Q-learning, Gadaleta and Dangelmayr introduced a general method for optimal chaos control [13]. They showed that this method not only can control high-dimensional discrete systems, 1-D and 2-D coupled logistic map lattices [14], but also can attack the targeting problem in a complex multi-stable system through guiding its trajectory to a metastable state [15]. Lei and Han successfully applied the method with Q-learning to the control of chaos in the Frenkel-Kontorova model [16]. The goal of reinforcement learning is to maximize the cumulative rewards, which determine whether the agent can ultimately learn the desired goal—stabilizing an unstable periodic orbit embedded in the chaotic attractor. The agent can acquire all information required to find the optimal control strategy through interaction with the system. The method is data driven and it provides an intelligent black box control because it does not require analytical descriptions of governing dynamical equations nor knowledge of targeted unstable periodic orbit.

On the other hand, the existence of chaotic characteristics makes dynamical systems flexible and useful. Davies *et al.* showed that chaotic behaviors can speed up the combustion process, because sensitive dependence of chaos on initial values promotes the mixing of reactants [17]. Li and Xu proposed that chaotification of the quasi-zero-stiffness system can mask line spectrum characteristics of acoustic noise of machinery vibration, so as to enhance the concealment capability of underwater vehicles [18]. In the neural network of biological systems with periodic solutions,

Kohannim and Iwasaki achieved an expected strange attractor by tuning a small parameter to destabilize oscillators' phase difference [19]. In this sense, chaos provides great flexibility for the system performance because the chaotic attractor is embedded in infinite unstable periodic orbits and we can accomplish different goals with the aid of these unstable orbits [20]. Chen and his coauthors designed a small-amplitude feedback controller to chaotify discrete and continuous systems [22, 23]. They showed that the controlled system is chaotic and has a positive Lyapunov exponent. Hereafter, various methods have been proposed for the anti-control of chaos, such as the impulse control [21], the feedback control [24, 25], and the topological conjugate mapping control [26]. Similarly, most of the existing methods on the anti-control of chaos face the problem of requiring explicit knowledge on systems in real world applications. We demonstrated that control of chaos using reinforcement learning is model-free and easy to employ in the FK model [16]. As far as we know, the method of anti-control of chaos using reinforcement learning has not been reported. A natural question is whether one can apply the reinforcement learning based method to the anti-control of chaos by appropriately designing states, actions and reward functions for the agents.

In this work, we consider the problem of control and anti-control of chaos using reinforcement learning based on the work of Gadaleta and Dangelmayr [13]. We show that the method is effective in controlling chaos with the stabilization of unstable periodic orbits by altering a control parameter or systems' variables and designing a reward function. If the goal changes from control to anti-control, one can design the opposite reward function by exchanging the actions of reward and punishment. Unfortunately, when applying the method to destabilizing a period-1 system for the purpose of anti-control of chaos, we find that the state of the controlled system may change to be in period-2. Using the method to anti-control chaos, we have to avoid suppressing the system to other high periodic orbits, instead of chaotic orbits. Even if we could avoid the problem of falling into a high periodic orbit, we may still face the problem of not distinguishing a quasi-periodic orbit from chaos. This will bring a difficulty in designing a reward function and should not be neglected. In view of this, we introduce a reward function based on the largest Lyapunov exponent of the system because it can be used to judge whether the corresponding system is chaotic. To describe the interactions of the agent and the environment in reinforcement learning, we need to make use of the concept of Markov decision process, which discards previous data after each interaction. Therefore, we should consider calculating the

largest Lyapunov exponent from a small data set. In this study, we define a moving largest Lyapunov exponent by using a recently updated time series with a fixed, short length. In 2018, Zhou and Wang proposed a practical method [27] for calculating the largest Lyapunov exponent from a small data set by using the density peaks-based clustering algorithm [28]. This method becomes a strong support for the moving largest Lyapunov exponent. On this basis, we adopt the density peaks-based clustering algorithm to determine a linear region of the average divergence index so that we can automatically obtain the largest Lyapunov exponent of the small data set by fitting the slope of the linear region. Once we derive the moving largest Lyapunov exponent, the reward function can be defined by the exponent, so as to implement a control policy using reinforcement learning.

The rest of the paper is organized as follows. In Sect.2, we briefly introduce reinforcement learning and describe the model-free control and anti-control algorithm based on the work of Ref. [13]. We take the Henon map and Lorenz systems as illustrative examples with numerical simulations. In Sect. 3, we propose a new method of control and anti-control of chaos based on the moving largest Lyapunov exponent using reinforcement learning. We also demonstrate the effectiveness and advantages of the proposed method with the two systems. Section 4 presents some conclusions and discussions.

2 Control and anti-control of chaos based on the states' period using reinforcement learning

This section discusses the problem of control and anti-control of chaos. Firstly, the theory of reinforcement learning is briefly reviewed, and then the control algorithm is presented and extended to the anti-control of chaos. Finally, the Henon map and Lorenz system are taken as illustrative examples to show the feasibility of the method.

2.1 Control policy with the states' period

In this study, we consider the use of model-free RL algorithm to control and anti-control chaos in dynamical systems. In the model-free RL algorithm, we don't need to have a clear understanding of a system. The agent's perception and cognition for the system

can be realized through continuous interaction with a simulation system, and the data obtained from the interaction is not used to model the system, but to optimize the agent's own behavior. This avoids the requirement of analytical knowledge of system models in the traditional control method.

Reinforcement learning (RL), as a common machine learning technique, is a process of intelligent decision-making. Its learning mode is closest to human learning mode in machine learning. Through process simulation and interaction, RL improves decision-making ability to quickly accomplish expected goals. RL includes five elements: agent, environment, state, action and reward. The specific learning process is as follows: when the environment's state is s_t at time t , the agent selects an action a_t according to the policy π (ε -greedy policy, deterministic policy *etc.*), and then the agent gets a next state s_{t+1} and a reward r_{t+1} after the environment executes a_t , where a probability of state transition $p(s_{t+1}, r_{t+1} | s_t = s, a_t = a)$ is obtained. Additionally, possible states of the environment constitute the state space S and possible actions that the agent can execute constitute the action space A . This process produces a sequence $\{s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T\}$. Usually, this sequence can be considered a finite Markov sequence, which assumes that the state and the reward from the environmental at the current time only depend on the state of the last time, that is, the state transition probability is $p(s_{t+1}, r_{t+1} | s_t, a_t) = p(s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a)$ at time t . This learning process is called the "agent-environment" interaction process in the Markov decision process.

In general, the goal of RL is to maximize the expectation of discounted accumulative rewards

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (1)$$

where $\gamma \in (0, 1)$ is a constant, which guarantees that the rewards are limited.

The RL algorithm is based on estimating the state-action value function as follows

$$q_{\pi}(s, a) = E_{\pi} \left[G_t | s_t = s, a_t = a \right] = E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right], \quad \forall s_t \in S, a_t \in A. \quad (2)$$

It is the expected reward of all possible decisions after the agent in a certain state s selects an action a according to the policy π . In Ref. [29], the Bellman equation of the state-action value function can be described by

$$\begin{aligned}
q_\pi(s, a) &= E_\pi [G_t | s_t = s, a_t = a] = \sum_{s', r} p(s', r | s, a) [r + \gamma V_\pi(s')] \\
&= \sum_{s', r} p(s', r | s, a) [r + \gamma \sum_{a'} \pi(a' | s') q_\pi(s', a')].
\end{aligned} \tag{3}$$

Note that the policy π is exactly a probability distribution, *i.e.*, it keeps a probability distribution and stores the probability of taking a certain action at each state.

Under the optimal policy π_* , whose cumulative reward is the highest, the optimal state-action value function is defined as $q_*(s, a) = \max_\pi q_\pi(s, a)$. Therefore, the bellman optimal equation of the state-action value function is

$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} q_*(s', a')]. \tag{4}$$

In general, state transition probabilities are hard to obtain. On this condition, one should seek the optimal strategy to control or anti-control chaos by using a model-free method. The temporal-difference (TD) method provides a model-free, one-step updated algorithm since it combines the Monte Carlo sampling method and bootstrapping of dynamic programming.

Gadaleta and Dangelmayr introduced a chaos control algorithm using the TD method in RL, in which the optimal state-action value function is approximated by Q or Sarsa learning, and they showed the performance of Q-learning is better than that of Sarsa learning [13]. Here, we select Q-learning as the control or anti-control algorithm and its state-action value function is updated with the following equation

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right], \tag{5}$$

where $0 < \alpha \leq 1$ is the learning rate and we set $\alpha = \gamma = 0.9$. In addition, a deterministic policy is used to choose action, which makes the current state-action pair's Q value maximum at each step. If there are several actions satisfied, we randomly choose one of them. Notice that the desired periodic or chaotic behaviors are unknown before the strategy is employed.

Each state-action pair's Q value is recorded in a Q-table, whose row and column represent the action space A and the reference state set W respectively (the Q values are initially 0). Moreover, the action space that depends on perturbed parameters of the system and the reference state set that depends on states of the system are set before an environment is controlled or anti-controlled. In each episode of the task of controlling or anti-controlling chaos, the first state s_0 is randomly initialized.

For each step $n = 1, 2, \dots$ in an episode, the agent first gets the reference state $w(s_n)$ of the system's state $s_n \in S$. Based on $w(s_n)$, the agent uses a deterministic strategy in the Q-table to select action a_n , and then implement a_n for the system getting the reward r_{n+1} and the next state s_{n+1} . At the same time, the value $Q(w(s_n), a_n)$ in the Q-table is updated by Eq. (5) until this episode reach the goal. In addition, our training method is an offline strategy. In other words, we train the Q-table through limited episodes (each episode randomly selects the initial state). According to the updated Q-table with any initial value, we can judge the best action to take in any state, so as to achieve the aim of control or anti-control.

The aim of controlling chaos is to guide a chaotic system to unstable periodic orbits or fixed points, which are embedded in the chaotic attractor in the environment (the system), or to make the chaotic system bifurcate to a periodic behavior. Due to the limitation of the Q-table's dimension, we need to reduce the dimension of the state space S before implementing control. The dimension reduction for the state space S can make use of reference state quantization of uncontrolled system's states. Here, we first choose N the number of clusters through hierarchical clustering for the states, and then according to the k-means clustering algorithm obtain the cluster centers as a set of the reference state set $W = \{w_1, w_2, \dots, w_N\}$. When the state space S is a one-dimensional interval, it is covered by a uniform distribution, and the midpoints w_i are used as the finite reference states of S . In any case, each state s in the state space S is projected onto a reference state $w(s)$ of the finite reference set W , in which the element is defines as

$$w(s) = \arg \min_{w \in W} \|w - s\|. \quad (6)$$

The target of control can be set with the aid of the reference set W . If the target is a fixed point, the criterion of target realization is $w_n = w_{n+1}$ (one-step control). If the target is a period 2 trajectory, the criterion is $w_n = w_{n+2}, w_n \neq w_{n+1}$. (two-step control). In the latter case, (w_{n-1}, w_n) is a reference state. If the target is a period l state, the criterion is $w_n = w_{n+l}, w_n \neq w_{n+1}, \dots, w_n \neq w_{n+l-1}$. So, the reward function can be defined as

$$r_{n+1} = \begin{cases} 0, & \text{if } w_n = w_{n+l}, w_n \neq w_{n+1}, \dots, w_n \neq w_{n+l-1}, \\ -1, & \text{otherwise.} \end{cases} \quad (7)$$

In the following, we will consider anti-control of chaos in view of the above control strategy. The anti-control of chaos refers to changing periodic behaviors of environments into chaotic behaviors. Unlike the control of chaos, where any of the unstable periodic orbits can be considered the target of control, the objective of anti-control of chaos is to destabilize the system and make it to reach a chaotic trajectory. For the anti-control of chaos using RL, there are two problems to be solved. Firstly, the reference state set W cannot be assigned in advance. To tackle this problem, we try to distribute a certain variable of the environment within a given interval, where the midpoints w_i constitute the states reference set W . In practical applications, one can choose different intervals for different aims. Similarly, according to Eq. (6), each state in the state space S is projected onto a reference state $w(s)$ of the finite reference set W . In the process of the anti-control, the reference state set can be adaptively changed according to the state of the slaved system. Secondly, when a periodic orbit of the system is destabilized, it may be forced to fall into another periodic orbit, instead of a chaotic one as expected. For the anti-control of chaos, we have to avoid this matter and design a reward function punishing the system falls into other periodic orbits. So, we define the reward function for the anti-control of chaos as follows

$$r_{n+1} = \begin{cases} 0, & \text{if } w_n \neq w_{n+1}, \dots, w_n \neq w_{n+l}, \\ -1, & \text{otherwise,} \end{cases} \quad (8)$$

where l is an integer.

2.2 Numerical simulation

In this subsection, we first take the Henon map as the simulation model to verify the algorithm in Sec.2.1. The Henon map has the following form,

$$\begin{cases} x_{n+1} = p - ax_n^2 + y_n + a_n, \\ y_{n+1} = bx_n, \end{cases} \quad (9)$$

where a_n is the controller exerted on the map. Without loss of generality, the initial values of the map are chosen uniformly distributed in the interval $[0, 1]$. And in the

offline strategy, we implement 300 episodes, that is, we use 300 randomly selected initial values to train the Q-table.

To illustrate the control of chaos, we fix the parameters of the Henon map as $p = 1.29, a = 1, b = 0.3$ and the system without control appears chaotic. We determine the reference state set W in accordance with the state space S of the uncontrolled system. We use 1000 points for hierarchical clustering to get the number $N = 400$ of clusters, and then use the cluster centers obtained by the k-means clustering algorithm as elements of the reference state set W . The action space is taken as $A = \{-0.025, 0., 0.025\}$. Assume that the target of the control is a period-one orbit. So,

the reward function is
$$r_{n+1} = \begin{cases} 0, & \text{if } w_n = w_{n+1}, \\ -1, & \text{otherwise.} \end{cases}$$

Figure 1 depicts that the variable x of the Henon map quickly turns from a chaotic behavior to a periodic one in the control of chaos once the control algorithm is employed.

The uncontrolled Henon map stays in a stable period-one state with parameters $p = 1.29, a = 0.1, b = 0.3$. To employ the anti-control technique, we divide the interval $[-2, 2]$ equally into 400 parts, where the midpoints of the parts constitute the reference state set W . The action space is $A = \{-0.02, -0.01, 0.0, 0.01, 0.02\}$. From Fig. 2(a), we show that the slaved Henon map changes from period 1 to period 2 when the reward function is
$$r_{n+1} = \begin{cases} 0, & \text{if } w_n \neq w_{n+1}, \dots, w_n \neq w_{n+4}, \\ -1, & \text{otherwise,} \end{cases}$$
 and in Fig. 2(b) the state variable

of the slaved Henon map changes from order to disorder when the reward function is

$$r_{n+1} = \begin{cases} 0, & \text{if } w_n \neq w_{n+1}, \dots, w_n \neq w_{n+20}, \\ -1, & \text{otherwise.} \end{cases}$$
 It is shown that when the period-one state of

the slaved system is destabilized, different results will be obtained by choosing different parameters l of the reward function, including ordered and disordered patterns. In this case, whether the goal of chaotification can be accomplished depends on the choice of the parameter l . Since it is difficult to choose an appropriate l in advance, the current reward function is not suitable for the purpose of the anti-control of chaos and should be modified.

Fig. 1 The evolution of the state variable of the slaved Henon map for the control of chaos, where the control action is performed on 200 iterations.

Fig. 2 The evolution of the slaved Henon map when a period-one orbit is destabilized, where the control action is performed on 200 iterations: (a) the evolution of variable x for $l=4$, (b) the evolution of variable x for $l=20$.

Next, we consider the Lorenz system with the following form

$$\begin{cases} \dot{x} = \sigma(-x + y), \\ \dot{y} = -xz + \rho x - y, \\ \dot{z} = xy - \beta z. \end{cases} \quad (10)$$

In the control of chaos, the parameters are set as $\sigma=16$, $\rho=50$, $\beta=4$ in Eqs. (10). We adopt the fourth-order Runge-Kutta algorithm to calculate Eqs. (10) with time step $h=0.005$ and show the system is chaotic. We use 10000 states (x, y, z) for hierarchical clustering to get the clustering number $N=3345$, and then take the clustering centers as elements of the reference state set W . The action a in the action space $A = \{-1, -0.5, 0, 0.5, 1\}$ is implemented on the state variable y at each step in

an episode. We take a period-one state as the target of the control and introduce a reward

function $r_{n+1} = \begin{cases} 0, & \text{if } w_n = w_{n+1}, \\ -1, & \text{otherwise.} \end{cases}$ The result is shown in Fig. 3.

Fig. 3 The evolution of variable x of the slaved Lorenz system for the control of chaos, where the control action is performed on 1000 iterations.

From Fig. 3, it is can be seen that the variable x of the slaved Lorenz system transfers from a chaotic oscillation to a fixed point. We conclude that the RL based method is also effective in controlling continuous chaotic systems.

In the anti-control of chaos, we set the parameters $\sigma = 16, \rho = 20, \beta = 4$ in the Lorenz system. The uncontrolled system stays in a stable fixed point after transients. We divide the interval $[-10, 10]$ of the variable y equally into 5000 parts, whose midpoints w_i form the reference set W . The action in the action space $A = \{-0.05, -0.04, -0.03, -0.02, -0.01, 0, 0.01, 0.02, 0.03, 0.04, 0.05\}$ is still exerted on variable y . In order to destabilize the fixed point, we try using different values l in

the reward function $r_{n+1} = \begin{cases} 0, & \text{if } w_n \neq w_{n+1}, \dots, w_n \neq w_{n+l}, \\ -1, & \text{otherwise,} \end{cases}$. We find that the variable

x transfers from a fixed point to periodic orbit after transient states when $l=50$, as shown in Fig. 4(a), and the variable x transfers from a fixed point to disorder after transient states when $l=60$, as shown in Fig. 4(b). In this sense, an appropriate choice of l will lead to disorder.

Fig. 4 The evolve of Lorenz system when a fixed point is destabilized, where the control action is performed on 1000 iterations: (a) the evolution of variable x for $l=50$, (b) the evolution of variable x for $l=60$.

In the above two systems, we show that the algorithm in Section 2.1 can quickly achieve the aim of controlling chaos. But we find that whether the anti-control of chaos succeeds depends on the choice of l in the reward function. Sometimes, the existing reward function can accomplish the task, but it cannot be employed easily. In an unknown environment, the value l in the reward function must be attempted and elaborately chosen, indicating that this technique is not suitable for general dynamical systems. Therefore, we have to modify the RL based method for the control and anti-control of chaos, in which the reward function is extremely important and must be redesigned.

3 Control and anti-control of chaos based on the moving largest Lyapunov exponent method using reinforcement learning

In this section, we define and calculate a moving Largest Lyapunov Exponent (mLLE), on which a new reward function is based and redesigned, thus proposing an algorithm for the control and anti-control of chaos using RL. We also use the Henon map and Lorenz system as illustrative examples to verify the proposed method.

3.1 Control policy with the moving largest Lyapunov exponent method

Choosing an appropriate reward function in RL is crucial when employing a control policy because the reward function relates desired goals to state variables of systems and determines whether tasks can be accomplished or not. As mentioned above, the anti-control of chaos in Sec. 2 often fails due to the reward function improperly chosen which brings the difficulty of finding a key parameter l for specific systems. For the purpose of seeking a universal strategy, we urgently need to redesign a reward function that can be applied to general systems. In a dynamical system, Lyapunov exponents measure average convergence or divergence rates of adjacent trajectories in phase space, and the sign of the largest Lyapunov exponent can be used to judge whether the system is chaos. Taking this into account, we design a novel reward function for the control and anti-control of chaos with the aid of the largest Lyapunov exponents.

With a model free control method, the model of the system is assumed to be unknown, and to compute its largest Lyapunov exponent, we usually require a big data set collected from the system. This requirement contradicts the assumption that the process of RL improving the current policy through observations of delayed rewards to seek an optimal policy is approximated as a finite Markov process that does not deal with a large amount of historic data. Starting from a set of small data and in view of the moving average method [27, 28], we define a moving largest Lyapunov exponent of the small data and use a clustering technique depending on density peaks to identify linear regions to calculate the mLLE with the following algorithm.

Algorithm 1 Calculating the moving largest Lyapunov exponent

Step1: Collect a set of small data with a fixed length L $\{x(t_0), x(t_0 + \Delta t), \dots, x(t_0 + (L-1)\Delta t)\}$, from an observable state of the environment where Δt is the sampling period of the data set.

Step2: Reconstruct the phase space of the small data as its element can be written as $X_r = (x(t_r), x(t_r + \tau\Delta t), \dots, x[t_r + (m-1)\tau\Delta t])$, where the index number $r \in \{0, 1, \dots, R\}$ and $R = L - (m-1)\tau - 1$, and the embedded dimension m and the delayed time τ are computed by the Cao method [30] and by the mutual information method [31], respectively.

Step3: Calculate the average divergence index set $\{y(k), k = 1, 2, \dots, K\}$. Firstly, compute $d_r(0) = \min_{r'} \|X_r - X_{r'}\|$, $r, r' = 0, 1, 2, \dots, R$ and $|r - r'| > p$ where p is the

mean period by the power spectrum method and then compute $d_r(k) = \|X_{r+k} - X_{r'+k}\|$, $k \in \{1, 2, 3, \dots, K_i\}$ and $K_i = \min(R-r, R-r')$. So, the average divergence index $y(k) = \frac{1}{\Delta t} \langle \ln d_r(k) \rangle$, where $\langle \cdot \rangle$ denote the average for r and $k = 1, 2, \dots, K$, $K = \max(K_0, \dots, K_R)$. If the average divergence index set is empty, the algorithm fails; else, turn to Step 4.

Step4: Determine the clustering number and clustering centers for $\{y(k), k = 1, 2, \dots, K\}$:

- Calculate distances $d_{k_1 k_2} = \|y(k_1) - y(k_2)\|$, $(k_1, k_2 = 1, 2, \dots, K)$ and cutoff distance d_c equals to the distance whose position is 2% of K in the ascending distances;
- Use the Gaussian kernel calculating the local densities $\rho_{k_1} = \sum_{k_2=1}^K e^{-d_{k_1 k_2}^2 / d_c^2}$;
- Calculate the minimum distance $\delta_{k_1} = \min_{k_2: \rho_{k_2} > \rho_{k_1}} d_{k_1 k_2}$ between the point k_1 and other points with higher local density;
- Determine clustering number and clustering centers. The ones, larger than $\max \rho_{k_1} / 2$ and $\max \delta_{k_1} / 2$, are selected as clustering centers and the clustering number denotes the number of the centers;
- Classify remaining points. Non-clustering centers are in the same cluster as the nearest and higher-density centers.

Step5: Pick the linear region. K points have a same cluster as $y(1)$ and their indexes increases sequentially. If the Pearson correlation coefficient of the cluster and their indexes is more than 0.9, turn to Step 6; else, implement Step 4 for the picked cluster. If $y(1)$ is an isolated cluster, the algorithm fails.

Step6: Fit the linear region using the Least Squares Method and the associated slope is taken as the current value of the mLLE.

Step7: Update the set of small data with length L , that is, delete the first point and insert a new data to the last point, and repeat the above procedure to calculate the current mLLE unless observable states of the environment end.

The proposed algorithm can forecast the sign of largest Lyapunov exponents by calculating current largest Lyapunov exponents of a continuously updated time series

with a fixed length. Therefore, we define a reward function according to the sign of moving largest Lyapunov exponents for control and anti-control of chaos using RL in Sec. 2.1. The reward function for control of chaos is defined as

$$r_{n+1} = \begin{cases} 0, & \text{if } \text{mLLE} < 0, \\ -1, & \text{if } \text{mLLE} \geq 0, \end{cases} \quad (11)$$

and the reward function for anti-control of chaos is

$$r_{n+1} = \begin{cases} 0, & \text{if } \text{mLLE} > 0, \\ -1, & \text{if } \text{mLLE} \leq 0. \end{cases} \quad (12)$$

For the control and anti-control of chaos based on the above reward functions using RL, the agent chooses an action a_n at the current state s_n of a system with the deterministic policy at the n^{th} step of an episode, and the system performs a_n to get the next state s_{n+1} and the agent obtains a reward r_{n+1} according to Eq. (11) or Eq. (12). After that, the agent uses Q-learning of the TD method to approximate the current state-action value function and update Q-table. The key elements of Q-learning, the reference set of state, the action space, and the Q-table are set with the techniques in analogy with the ones in Sec. 2.1. In addition, the offline strategy in Q-learning is employed. In this strategy, learning is carried out the deterministic policy in many episodes and each episode is started with a new randomly chosen state. For the learning process, an episode is terminated if the condition $r_{n+1} = 0$ is met for consecutive steps or the number of steps exceeds I . Moreover, if learning is performed over J episodes, the training of Q-table completes. The detailed procedure with the new reward function is shown in Algorithm 2.

Algorithm 2 Model-free Q-learning for Control/Anti-control of Chaos

Initialize the Q-table: $Q(w, u) : Q(w, u) = 0, \forall (w, u)$.

for episode = 1, J **do**

Initialize a random state s_0 and time series $\{s_0, s_1, \dots, s_{L-1}\}$.

for $n = 1, I$ **do**

Get the current reference state $w_n = w(s_n)$.

Select the action a_n under w_n through the deterministic policy.

Execute a_n in the system and then obtain s_{n+1} and w_{n+1} .

Calculate mLLE of the series $\{s_{n-L+1}, s_{n-L+2}, \dots, s_n\}$ using Algorithm 1.

Set the reward: control $r_{n+1} = \begin{cases} 0, & \text{if } \text{mLLE} < 0, \\ -1, & \text{if } \text{mLLE} \geq 0. \end{cases}$

Or, anti-control $r_{n+1} = \begin{cases} 0, & \text{if } \text{mLLE} > 0, \\ -1, & \text{if } \text{mLLE} \leq 0. \end{cases}$

Update Q-table using Q-learning:

$$Q_n(w_n, a_n) \leftarrow Q_n(w_n, a_n) + \alpha \left[r_{n+1} + \gamma \max_{a'} Q(w_{n+1}, a') - Q_n(w_n, a_n) \right].$$

Update the time series (delete s_{n-L+1} and add s_{n+1}).

Set $s_n \leftarrow s_{n+1}, w_n \leftarrow w_{n+1}$.

end for

end for

In Algorithm 2, the reward function based on the mLLE solves from two aspects the problems encountered in Sec. 2.1 that uses the reward function based on states' period. Firstly, the reward functions of Eq. (11) and Eq. (12) are more robust to environments than the ones of Eq. (7) and Eq. (8). We need not reselect the length L of the dataset used to calculate the mLLE in controlling and anti-controlling a same environment and at the same time we avoid the difficulty of choosing the state's period parameter l according to different environments. Secondly, the reward functions of Eq. (11) and Eq. (12) solve the problem that the reward functions Eq. (7) and Eq. (8) cannot distinguish whether the observed time series of states is chaotic, high-periodic or quasi-periodic. However, in Algorithm 2, the problem of determining the length L of the data set is yet to be solved. In this case, the parameter L is taken as a hyperparameter, which we will show in the numerical simulations can be defined in advance or learned from data in the learning process.

3.2 Numerical simulation

In this subsection, we still choose the Henon map and Lorenz system to check the validity of Algorithm 2. In the algorithm, we calculate at each step of an episode the mLLE before updating the time series by adding a new observed state of system to its end and deleting its first element. In the learning process, we set the maximum number of steps in each episode as $I = 500$, when the number of steps $i > I$ in an episode we

execute the next episode, and set the maximum number of episodes as $J = 300$, when the number of episodes $j > J$ control ends or fails. The Q-learning also adopts an offline strategy. After getting the learned Q-table, we select an initial value to verify the control effectiveness.

We first consider the performance of Algorithm 2 in the Henon map. For the control of chaos, we fix Henon map's parameters $p = 1.29, a = 1, b = 0.3$. The reference state set W and the action space A are the same as in the control of the Henon map in Sec.2.2, the reward function is defined in Eqs. (11), and the length $L = 100$ of observed states is used to calculate the mLLE. The result of controlling chaos is shown in Fig. 5. In Fig. 5, the variable x quickly changes from chaotic to periodic when the control is turned on after 200 iterations. Figure 5 verifies that Algorithm 2 can also control chaos in the Henon map even if we have no analytic information of the system and are short of prerequisite knowledge of a target periodic orbit.

Fig. 5 The evolution of the state variable of the slaved Henon map for the control of chaos, where the control action is performed on 200 iterations.

In the anti-control of chaos, the uncontrolled system stays in a fixed point with parameters set as $p = 1.29, a = 0.1, b = 0.3$. The reference set W and the action space A are the same as in Sec. 2.2. The reward function is defined in Eqs. (12). It can be seen in Fig. 6 the variable x of the slaved Henon map appears aperiodic, random but bounded after transient states when the control policy is performed on 100 iterations. Therefore, the mLLE based reward function also enables the RL technique to chaotify the Henon map. It is found that in the control and anti-control of the Henon map, the length L of the observable states remains unchanged, which is more applicable than

the return function in Sec. 2.2. Indeed, mLLEs of the observed states is greater than 0 after the anti-control technique is implemented, indicating that the slaved map is a chaotic system, which avoids the slaved system falling into a high periodic orbit. In addition, when we set different L values, such as $L=150, 200$, the goal of control or anti-control of chaos in the Henon map can still be achieved.

Fig. 6 The evolution of the state variable of the slaved Henon map when a period-one orbit is destabilized, where the control action is performed on 100 iterations.

Fig. 7 The evolution of variable x of the slaved Lorenz system for the control of chaos, where the control action is performed on 1000 iterations.

For the purpose of checking the performance of Algorithm 2 in a low-dimensional continuous dynamical system, we consider the Lorenz system an environment of RL. The Lorenz system exhibits a chaotic behavior with parameters $\sigma=16, \rho=50, \beta=4$. For the key elements of RL, the reference state set W and action space A are the same as those in Sec. 2.2. The reward function is chosen as Eqs. (11) and the hyperparameter $L=100$. Fig. 7. depicts the variable of the controlled system

eventually evolves along a fixed point. It is also shown that Algorithm 2 is effective for the control of chaos in the continuous system.

When the parameters are assumed to be $\sigma = 16$, $\rho = 20$, $\beta = 4$, the uncontrolled Lorenz system appears periodic. For the key parameters, we select the same action state A and reference set W as those in the control of chaos in the Lorenz system in Sec. 2.2. In the anti-control of chaos, the reward function of Algorithm 2 is selected as Eqs. (12) and the hyperparameter $L = 100$. The anti-control results are shown in Fig. 8 where the control action is executed after 1000 iterations. Fig. 8(a) shows that the state variable evolves from a fixed point to a irregular behavior and the phase space $x-y$ in Fig. 8(b) is bounded and chaotic after transient states. Besides, the mLLEs of the observed states are controlled to be greater than 0 with the anti-control strategy, so that the slaved system is chaotic. Similarly, we set the hyperparameter value $L = 150, 200$, and find Algorithm 2 can still achieve the aim of control and anti-control of chaos in the Lorenz system.

Fig. 8 The anti-control of Lorenz system, where the control action is performed on 1000 iterations. (a) the evolution of variable x , (b) the phase diagram $x-y$ after transient states.

Figures 5-8 show that the mLLE-based reward function, which makes the control and anti-control of chaos using RL still effective, is more general than the states' period-based reward function. With the same length L of observed states for calculating the mLLE, control and anti-control of chaos can be realized in both discrete and continuous dynamical systems.

4 Conclusions and discussions

This study considers the control and anti-control of chaos using a RL based method. The method is model-free and only requires observed data from the environment to interact, instead of mathematical models of systems and targeted orbits. Besides controlling chaos, we generalize the model-free RL method based on the state's period to the anti-control of chaos by reversing the original reward function, that is, interchanging reward and punishment of the function. We show with numerical simulations that the method can quickly achieve the aim of controlling chaos, but sometimes fails for the anti-control of chaos due to the improper choice of the state's period parameter making the systems fall into another periodic orbits. Furthermore, the method faces the problem of not distinguishing between chaos and high-periodic or quasi-periodic orbits. To overcome the shortcomings and seek a universal control strategy, we modify the RL based method and redesign a new reward function by defining a moving Largest Lyapunov Exponent (mLLE) for the observed data with a fixed, short length. To calculate the mLLE, we adopt the density peaks-based clustering algorithm to determine a linear region of the average divergence index so that we can automatically obtain the largest Lyapunov exponent of the small data by fitting the slope of the linear region. Since the mLLE borrows the idea of the moving average, it can be used to characterize current dynamical behaviors of the observed data and thus guide with the control strategy the system to the trajectory as expected with the aid of the reward function. Further numerical simulations show that the proposed method is fast and easy to implement through controlling and anti-controlling typical systems such as the Henon map and Lorenz system. In future, we will consider controlling spatiotemporal chaos using deep RL based on the mLLE.

Acknowledgements The work is supported by the National Natural Science Foundation of China (Grant Nos. 12072262, 11972292).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Lo, S.C., Cho, H.J.: Chaos and control of discrete dynamic traffic model. *J. Franklin Inst.* **342**(7), 839-851 (2005)

2. Upadhyay, R.K., Bairagi, N., Kundu, K., Chattopadhyay, J.: Chaos in eco-epidemiological problem of the salton sea and its possible control. *Appl. Math. Comput.* **196**(1), 392-401 (2008)
3. Lian, X.B., Liu, J.F., Zhang, J.X., Wang, C.: Chaotic motion and control of a tethered-sailcraft system orbiting an asteroid. *Commun. Nonlinear Sci. Numer. Simul.* **77**(1), 203-224 (2019)
4. Bella, G., Mattana, P.: Chaos control in presence of financial bubbles. *Econ. Lett.* **193**(6), 109314 (2020)
5. Boccaletti, S., Grebogi, C., Lai, Y.C., Mancini, H., Maza, D.: The control of chaos: theory and applications. *Phys. Rep.* **329**(3), 103-197 (2000)
6. Ott, E., Grebogi, C., Yorke, J.A.: Controlling chaos. *Phys. Rev. Lett.* **64**, 1196-1199 (1990)
7. Pyragas, K.: Continuous control of chaos by self-controlling feedback. *Phys. Lett. A* **170**(6), 421-428 (1992)
8. Pyragas, K., Tamaševičius, A.: Experimental control of chaos by delayed self-controlling feedback. *Phys. Lett. A* **180**(1), 99-102 (1993)
9. Lei, Y.M., Zheng, F., Shao, X.Z.: Chaos and chaos control of the frenkel-kontorova model with dichotomous noise. *Int. J. Bifurcation Chaos* **27**(4), 1750052 (2017)
10. Pyragas, V., Pyragas, K.: State-dependent act-and-wait time-delayed feedback control algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **73**(7), 338-350 (2019)
11. Jackson, E.A., Grosu, I.: An open-plus-closed-loop (OPCL) control of complex dynamic systems. *Phys. D* **85**, 1–9 (1995)
12. Sinha, S., Ramaswamy, R., Rao, J.: Adaptive control in nonlinear dynamics. *Phys. D* **43**(1), 118–128 (1990)
13. Gadaleta, S., Dangelmayr, G.: Optimal chaos control through reinforcement learning. *Chaos* **9**(3), 775 (1999)
14. Gadaleta, S., Dangelmayr, G.: Control of 1-D and 2-D coupled map lattices through reinforcement learning. In: *International Conference Proceedings of Control of Oscillations and Chaos*, vol.1, Saint Petersburg, Russia, 2000, pp. 109-112
15. Gadaleta, S., Dangelmayr, G.: Learning to control a complex multistable system. *Phys. Rev. E* **63**(3), 036217 (2001)
16. Lei, Y.M., Han, Y.Y.: Control of chaos in the Frenkel-Kontorova model using reinforcement learning. *Chin. Phys. B* 2020, doi: 10.1088/1674-1056/abd74f
17. Davies, M.L., Halford-Maw, P.A., Hill, J., Tinsley, M.R., Johnson, B.R., Scott, S.K., Kiss, I.Z., Gáspár, V.: Control of chaos in combustion reactions. *J. Phys. Chem. A* **104**(44), 9944-9952 (2000)
18. Li, Y., Xu, D.: Chaotification of quasi-zero-stiffness system with time delay control. *Nonlinear Dyn.* **86**, 353–368 (2016)
19. Kohannim, S., Iwasaki, T.: Design of coupled Andronov–Hopf oscillators with desired strange attractors. *Nonlinear Dyn.* **100**, 1659–1672 (2020)
20. Chen, G.R.: Chaotification via feedback control: theories, methods, and applications. In: *Proceedings of IISWC 2003—International Conference on Physics and Control*, vol. 2, Saint Petersburg, Russia, 2003, pp. 468-474

21. Yang, L., Liu, Z., Chen, G.R.: Chaotifying a continuous-time system via impulsive input. *Int. J. Bifurc. Chaos* **12**(05), 1121 (2002)
22. Wang, X., Chen, G.R.: Chaotifying a stable LTI system by tiny feedback control. *IEEE Trans. Circuits Syst. I* **47**, 410–415 (2000)
23. Wang, X., Chen, G.R.: Chaotification via arbitrarily small feedback controls: theory, method, and applications. *Int. J. Bifurc. Chaos* **10**, 549–570 (2000)
24. Zhou, T., Chen, G.R., Yang, Q.: A simple time-delay feedback anticontrol method made rigorous. *Chaos* **14**(3), 662–668 (2004)
25. Wang, X., Chen, G.R., Yu, X.: Anticontrol of chaos in continuous-time systems via time-delay feedback. *Chaos* **10**(4), 771 (2000)
26. Wang, X., Chen, G.R.: Generating topologically conjugate chaotic systems via feedback control. *IEEE Trans. Circ. Syst. I* **50**(6): 812-817 (2003)
27. Zhou, S., Wang, X.: Identifying the linear region based on machine learning to calculate the largest Lyapunov exponent from chaotic time series. *Chaos* **28**(12), 123118 (2018)
28. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. *Science* **344**, 1492-1496 (2014)
29. Singh, S.P., Barto, A.G., Grupen, R., Connolly, C.: Robust reinforcement learning in motion planning. In: *Proceeding of NIPS 6—Advances in Neural Information Processing Systems*, Morgan Kaufmann, 1994, pp. 655-662
30. Cao, L.: Practical method for determining the minimum embedding dimension of a scalar time series. *Phys. D* **110**, 43–50 (1997)
31. Liebert, W., Schuster, H.G.: Proper choice of the time delay for the analysis of chaotic time series. *Phys. Lett. A* **142**, 107–111 (1989)

Figures

Figure 1

The evolution of the state variable of the slaved Henon map for the control of chaos, where the control action is performed on 200 iterations.

Figure 2

The evolution of the slaved Henon map when a period-one orbit is destabilized, where the control action is performed on 200 iterations: (a) the evolution of variable x for $l = 4$, (b) the evolution of variable x for $l = 20$.

Figure 3

The evolution of variable x of the slaved Lorenz system for the control of chaos, where the control action is performed on 1000 iterations.

(a)

(b)

Figure 4

The evolve of Lorenz system when a fixed point is destabilized, where the control action is performed on 1000 iterations: (a) the evolution of variable x for $l = 50$, (b) the evolution of variable x for $l = 60$.

Figure 5

The evolution of the state variable of the slaved Henon map for the control of chaos, where the control action is performed on 200 iterations.

Figure 6

The evolution of the state variable of the slaved Henon map when a period-one orbit is destabilized, where the control action is performed on 100 iterations.

Figure 7

The evolution of variable x of the slaved Lorenz system for the control of chaos, where the control action is performed on 1000 iterations.

Figure 8

The anti-control of Lorenz system, where the control action is performed on 1000 iterations. (a) the evolution of variable x , (b) the phase diagram x - y after transient states.