

Model-free prediction of chaotic dynamics with parameter-aware reservoir computing

Jianmin Guo(郭建敏), Yao Du(杜瑶), Haibo Luo(罗海波), Xuan Wang(王晅), Yizhen Yu(于一真), and Xingang Wang(王新刚)

Citation: Chin. Phys. B, 2025, 34 (**4**): 040505. DOI: 10.1088/1674-1056/adb733
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

A fractional-order chaotic Lorenz-based chemical system: Dynamic investigation, complexity analysis, chaos synchronization, and its application to secure communication

Haneche Nabil and Hamaizia Tayeb
Chin. Phys. B, 2024, 33 (**12**): 120503. DOI: 10.1088/1674-1056/ad7fcf

Dynamical analysis, control, boundedness, and prediction for a fractional-order financial risk system

Kehao Yang(杨轲皓), Song Zheng(郑松), Tianhu Yu(余天虎), Aceng Sambas, Muhamad Deni Johansyah, Hassan Saberi-Nik, and Mohamad Afendee Mohamed
Chin. Phys. B, 2024, 33 (**11**): 110501. DOI: 10.1088/1674-1056/ad7afb

Two-dimensional-lag complex logistic map with complex parameters and its encryption application

Fangfang Zhang(张芳芳), Jinbo Wu(武金波), Lei Kou(寇磊), Fengying Ma(马凤英), Liming Wu(吴黎明), and Xue Zhang(张雪)
Chin. Phys. B, 2024, 33 (**5**): 050505. DOI: 10.1088/1674-1056/ad2a69

Fractional-order heterogeneous memristive Rulkov neuronal network and its medical image watermarking application

Dawei Ding(丁大为), Yan Niu(牛炎), Hongwei Zhang(张红伟), Zongli Yang(杨宗立), Jin Wang(王金), Wei Wang(王威), and Mouyuan Wang(王谋媛)
Chin. Phys. B, 2024, 33 (**5**): 050503. DOI: 10.1088/1674-1056/ad322b

Coexistence behavior of asymmetric attractors in hyperbolic-type memristive Hopfield neural network and its application in image encryption

Xiaoxia Li(李晓霞), Qianqian He(何倩倩), Tianyi Yu(余天意),Zhuang Cai(才壮), and Guizhi Xu(徐桂芝)
Chin. Phys. B, 2024, 33 (**3**): 030505. DOI: 10.1088/1674-1056/acf281

COMPUTATIONAL PROGRAMS FOR PHYSICS

Model-free prediction of chaotic dynamics with parameter-aware reservoir computing

Jianmin Guo(郭建敏)[1,2], Yao Du(杜瑶)[1], Haibo Luo(罗海波)[1], Xuan Wang(王晅)[1], Yizhen Yu(于一真)[1,†], and Xingang Wang(王新刚)[1]

[1] *School of Physics and Information Technology, Shaanxi Normal University, Xi'an* 710062, *China*
[2] *College of Physics and Electronic Information Engineering, Qinghai Normal University, Xining* 810008, *China*

(Received 27 December 2024; revised manuscript received 15 February 2025; accepted manuscript online 18 February 2025)

Model-free, data-driven prediction of chaotic motions is a long-standing challenge in nonlinear science. Stimulated by the recent progress in machine learning, considerable attention has been given to the inference of chaos by the technique of reservoir computing (RC). In particular, by incorporating a parameter-control channel into the standard RC, it is demonstrated that the machine is able to not only replicate the dynamics of the training states, but also infer new dynamics not included in the training set. The new machine-learning scheme, termed parameter-aware RC, opens up new avenues for data-based analysis of chaotic systems, and holds promise for predicting and controlling many real-world complex systems. Here, using typical chaotic systems as examples, we give a comprehensive introduction to this powerful machine-learning technique, including the algorithm, the implementation, the performance, and the open questions calling for further studies.

Keywords: chaos prediction, time-series analysis, bifurcation diagram, parameter-aware reservoir computing

PACS: 05.45.Ac, 05.45.Tp, 07.05.Mh, 89.75.–k

DOI: 10.1088/1674-1056/adb733 **CSTR:** 32038.14.CPB.adb733

1. Introduction

Since the discovery of chaos in nonlinear dynamical systems, predicting chaotic motion has remained a topic of enduring interest across various fields.[1,2] Chaos prediction generally falls into two categories, depending on the application scenarios. The first category focuses on state evolution, where the system dynamics are fixed, and the task is to forecast how the system will evolve in the future based on past information. This approach is applied in tasks such as weather forecasting, stock market prediction, and population evolution interference. Although the inherent sensitivity of chaos excludes long-term predictions, short-term forecasts are feasible if accurate models of the system dynamics are available. The second category addresses the bifurcation diagram, exploring how a small change in the system parameters might affect its dynamics. Typical examples of this kind include climate systems, neural networks, and ecosystems. In these cases, numerical methods can provide insights, but, again, the applicability of the methods depends on the availability of accurate models. However, for real-world chaotic systems, precise models are generally lacking, what are available are only measured data, rendering data-driven prediction of chaotic systems a challenge of paramount interest in nonlinear science.[1,2]

Whereas sophisticated techniques have been developed in the literature for predicting chaos based on measured data, the techniques are generally system-dependent (different techniques are adopted for different systems and prediction tasks), and the success of the techniques relies heavily on the complexity of the system dynamics. For instance, the technique of sparse optimization has been proposed for finding the governing equations of chaotic systems from measured data, yet it applies to only cases where the system functions contain a small number of elementary mathematical terms.[3,4] A new approach proposed recently for predicting chaos involves exploiting the deep-learning techniques in machine learning. In particular, the technique of reservoir computing (RC) has been employed to predict chaos in different systems, and great successes have been achieved.[5–16] The RC technique was proposed in the early 2000s by Jaeger (the echo state networks) and Maass (the liquid state machine) independently,[5,6] which is a special kind of recurrent neural network (RNN) designed for processing time-sequential data. Unlike the other deep-learning techniques (e.g., the convolutional neural network, CNN), a reservoir computer contains only a single hidden layer (namely, the reservoir) and, except for the output matrix (which is to be estimated from the data through a training process), all other settings of the machine are fixed at the construction, including the input matrix, the reservoir network, and the updating rules. This feature makes RC appealing and favorable in applications when data is scarce and computational resources are limited (i.e., edge devices). Though structurally simple, RC has shown great potential in many data-

[†]Corresponding author. E-mail: yzyu@snnu.edu.cn

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

oriented applications,[17–19] e.g., speech recognition, channel equalization, robot control, and chaos prediction. For instance, it has been demonstrated that a properly trained RC is able to predict accurately the state evolution of typical chaotic systems for more than half a dozen Lyapunov times,[6] which is much longer than the prediction horizon of the traditional methods developed in nonlinear science. Besides predicting the short-term state evolution, RC is also able to replicate faithfully the long-term statistical properties of chaotic systems, e.g., the dimension of strange attractors and the Lyapunov exponents.[7,8] Very recently, the RC technique has also been applied successfully to predicting high-dimensional spatiotemporal systems by a parallel scheme,[20,21] holding promise in dealing with real-world complex systems.

While initial studies of RC focus on the prediction of state evolutions (the first category of chaos prediction), recent attempts have begun to exploit RC for inferring the bifurcation of the system dynamics based on partial observations (the second category of chaos prediction). Different from state-evolution predictions, the general question in bifurcation prediction is: Given the time series measured at a few system states of distinct parameters, what happens to the system dynamics if the parameter is changed to a new one? The standard RC technique, which is designed for predicting chaos evolution, fails to address this question, as what the machine learns from the training data is the system dynamics associated with a specific parameter. To make the RC technique capable of predicting new dynamics not included in the training data, it is necessary to endue the machine with the ability of knowledge transfer. Specifically, the dynamical information of the target system that the machine learned from the sampled states can be transferred and used to predict the dynamics of a new state.

One approach to achieving knowledge transfer in RC is introducing a control channel to the standard algorithm, with the bifurcation parameter serving as the additional input.[22–33] In this approach, in addition to the time series of the sampling states, the machine is also fed with the bifurcation parameter that generates the data. In the training phase, the time series of the sampling states are fed into the machine through the state channels in sequence, and the associated bifurcation parameters are also inputted through the parameter channel. In the predicting phase, the machine is fed with the bifurcation parameter only, while its outputs are used as the inputs in the state channels. As the machine is evolving under the "supervision" of the bifurcation parameter in both the training and predicting phases, this approach is termed parameter-aware reservoir computing (PARC) in Refs. [24,25]. It is demonstrated that the new machine is able to not only replicate accurately the dynamics of the sampling states, but also infer faithfully the statistical properties of the system dynamics under new bifurcation parameters. In Ref. [24], the PARC technique is applied successfully to the prediction of critical transitions of chaotic systems, in which the training data are acquired from the normal states, and it is shown that the trained machine is able to predict not only the occurrence of crises but also the transient time of system collapse. In Ref. [25], the PARC technique is exploited to infer the transition path from desynchronization to synchronization in coupled systems, where the sampling states are taken from the desynchronization regime while the trained machine is able to anticipate the critical coupling for achieving synchronization. Similar results are also obtained for coupled systems showing amplitude death, as reported in Ref. [26]. In Ref. [27], this technique is applied to nonlinear Hamiltonian systems, and it is demonstrated that the machine that is trained by the time series of a few states can reproduce the whole KAM bifurcation diagram in the phase space accurately. Applications of PARC to experimental data generated by chaotic circuits are reported in Ref. [31], in which it is shown that by the noisy signals measured at several sampling states, the machine can reproduce the whole bifurcation diagram with high precision. In addition to predicting critical transitions and new dynamics, PARC can also be employed to track the variation of the system parameter (implementing the machine reversely) and store multiple dynamical attractors (using attractor labels as parameter inputs).[33–35] Inspired by the great success achieved, growing attention has been paid in recent years to the application of PARC to the prediction and control of complex dynamical systems, as well as using theories in nonlinear and network science to explore the working mechanism of PARC.[22–33]

The purpose of the present work is to introduce the technique of PARC to colleagues in the community of nonlinear science and complex systems, with the hope of attracting their interest to this powerful technique, either using it as a tool to address the long-standing questions in the fields or using theories in nonlinear and complexity science to explore the working mechanism of this powerful artificial neural network. The rest of the paper is organized as follows. In Section 2, we shall introduce the PARC algorithm in detail, including the machine structure, the preparation of the training data, and the three phases for implementing the machine. Applications of the PARC technique to the reconstruction of bifurcation diagrams for three typical chaotic systems, including the logistic map, the Chua's circuit, and the Lorenz-96 climate network, will be demonstrated in Section 3. The influence of the sampling states on the prediction performance will also be discussed in this section. Finally, open questions in the study of PARC will be discussed in Section 4.

2. Parameter-aware reservoir computing

Similar to the standard RC, the PARC also consists of three modules:[24,25] the input layer (input-to-reservoir),

the hidden layer (reservoir network), and the output layer (reservoir-to-output). The structure of the PARC machine is schematically shown in Fig. 1(a). The input layer is characterized by the matrix $\boldsymbol{W}_{\rm in} \in \mathbb{R}^{D_{\rm r}\times D_{\rm in}}$, which couples the input vector $\tilde{\boldsymbol{u}}_\beta(t) \in \mathbb{R}^{D_{\rm in}}$ to the reservoir network. Different from the standard RC, here the input layer contains two channels: the state channel and the parameter-control channel. The state vector acquired from the target system, $\boldsymbol{u} \in \mathbb{R}^{D_{\rm in}-1}$, is inputted through the state channel, while the bifurcation parameter associated with the state, β, is inputted through the parameter-control channel. The elements of $\boldsymbol{W}_{\rm in}$ are randomly drawn from a uniform distribution within the range $[-\sigma,\sigma]$. The reservoir network contains $D_{\rm r}$ nodes, with the initial states of the nodes being randomly chosen from the interval $[-1,1]$. The states of the nodes in the reservoir network, $\boldsymbol{r}(t) \in \mathbb{R}^{D_{\rm r}}$, are updated as

$$\boldsymbol{r}(t+\Delta t) = (1-\alpha)\boldsymbol{r}(t) + \alpha\tanh[\boldsymbol{A}\boldsymbol{r}(t) + \boldsymbol{W}_{\rm in}\tilde{\boldsymbol{u}}_\beta(t)]. \quad (1)$$

Here, Δt is the time step for updating the reservoir network, $\alpha \in (0,1]$ is the leaking rate, tanh represents the hyperbolic tangent function, $\boldsymbol{A} \in \mathbb{R}^{D_{\rm r}\times D_{\rm r}}$ is a weighted adjacency matrix representing the coupling relationship between nodes in the reservoir. $\tilde{\boldsymbol{u}}_\beta(t) = [\boldsymbol{u}(t),\beta(t)]^{\rm T}$ is the input vector at time t, with $\boldsymbol{u}$ the system state and β the system parameter assocaited with the state. The adjacency matrix $\boldsymbol{A}$ is constructed as a sparse random Erdös–Rényi matrix: with the probability p, each element of the matrix is arranged a nonzero value drawn randomly from the interval $[-1,1]$. The matrix $\boldsymbol{A}$ is rescaled to make its spectral radius equal to λ. The output layer is characterized by the matrix $\boldsymbol{W}_{\rm out} \in \mathbb{R}^{D_{\rm out}\times \tilde{D}_{\rm r}}$, which generates the output vector, $\boldsymbol{v}(t) \in \mathbb{R}^{D_{\rm out}}$, according to the equation

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{W}_{\rm out}\tilde{\boldsymbol{r}}(t+\Delta t), \quad (2)$$

where $\tilde{\boldsymbol{r}} \in \mathbb{R}^{\tilde{D}_{\rm r}}$ (with $\tilde{D}_{\rm r} = 2D_{\rm r}+1$) is the new state vector transformed from the reservoir state. Specifically, we set $\tilde{r}_1 = 1$, $\tilde{r}_i = r_i$ for $i = 2,\ldots,D_{\rm r}+1$, and $\tilde{r}_i = r_i^2$ for $i = D_{\rm r}+2,\ldots,\tilde{D}_{\rm r}$. Here, $\boldsymbol{W}_{\rm out}$ is the output matrix to be estimated through a training process. Except $\boldsymbol{W}_{\rm out}$, all other parameters of the RC, e.g., α, $\boldsymbol{W}_{\rm in}$ and $\boldsymbol{A}$, are fixed at the construction. For the sake of simplicity, we set $D_{\rm out} = D_{\rm in} - 1$ in our studies.[7,8,20]

The implementation of PARC consists of three phases: training, validation, and prediction. The mission of the training phase is to find a suitable output matrix $\boldsymbol{W}_{\rm out}$ so that the output vector $\boldsymbol{v}(t+\Delta t)$ as calculated by Eq. (2) is as close as possible to the input vector $\boldsymbol{u}(t+\Delta t)$ for $t = (\tau+1)\Delta t,\ldots,(\tau+\hat{L})\Delta t$, with $T_0 = \tau\Delta t$ the transient period (used for removing the impact of the initial conditions of the reservoir) and $\hat{L}$ the length of the training series. This is done by minimizing the cost function with respect to $\boldsymbol{W}_{\rm out}$,[7,8,20] which gives

$$\boldsymbol{W}_{\rm out} = \boldsymbol{U}\boldsymbol{V}^{\rm T}(\boldsymbol{V}\boldsymbol{V}^{\rm T} + \eta\mathbb{I})^{-1}. \quad (3)$$

Here, $\boldsymbol{V} \in \mathbb{R}^{\tilde{D}_{\rm r}\times\hat{L}}$ is the state matrix whose k-th column is $\tilde{\boldsymbol{r}}[(\tau+k)\Delta t]$, $\boldsymbol{U} \in \mathbb{R}^{D_{\rm out}\times\hat{L}}$ is a matrix whose k-th column is $\boldsymbol{u}[(\tau+k)\Delta t]$, $\mathbb{I}$ is the identity matrix, and η is the ridge regression parameter for avoiding overfitting. We note that in the training phase the input data consists of two different time series: (i) the input vector $\boldsymbol{u}_\beta(t)$ representing the state of the target system and (ii) the control (bifurcation) parameter $\beta(t)$ labeling the condition under which the input vector $\boldsymbol{u}_\beta(t)$ is acquired. Assuming the data are acquired from m sampling states ($\beta \in \{\beta_i\}$, $i = 1,\ldots,m$), with the time series of each state containing $\hat{n}$ data points, the training dataset is composed of a concatenated time series of length $\hat{L} = m\times\hat{n}$. Please note also that the time series of $\beta(t)$ is a step-function of time, as the system parameter is kept unchanged for each sampling series. The structure of the training data is schematically shown in Fig. 1(b).

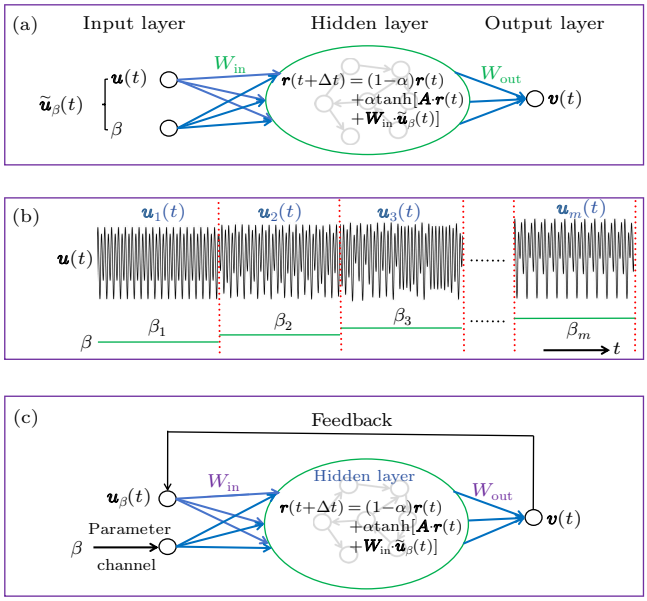


Fig. 1. Schematic of the PARC technique. (a) The open-loop configuration of the machine operating in the training phase. (b) Schematic of the training data. (c) The closed-loop configuration of the machine operating in the predicting phase.

The machine that performs well on the training data might not perform equally well on the testing data. The finding of the optimal machine performing well on both the training and testing data is the mission for the validation phase. The set of hyperparameters to be optimized in the machine includes $D_{\rm r}$ (the size of the reservoir network), p (the density of the adjacency matrix $\boldsymbol{A}$), σ (the range defining the input matrix), λ (the spectral radius of the adjacency matrix $\boldsymbol{A}$), η (the regression coefficient), and α (the leaking rate). In our studies, the optimal hyperparameters are obtained by scanning each hyperparameter over a certain range in the parameter space using conventional optimization algorithms such as the Bayesian and surrogate optimization algorithms.[24] After finding the optimal machine, we then utilize it to infer the system dynamics, namely, the predicting phase. Figure 1(c) shows the flowchart

of the machine in the predicting phase. In making the predictions, we replace $\boldsymbol{u}_\beta(t)$ with $\boldsymbol{v}(t)$ (so that the machine is working in the closed-loop configuration), while setting the control parameter β to a specific value of interest. As such, in the predicting phase the machine is still driven by the externally added parameter β. The output vector $\boldsymbol{v}(t)$ then gives the predictions. Still, before making the predictions, a short transient is discarded to avoid the impact of the initial conditions of the reservoir. If the mission is to predict the state evolution of the system under parameter β, the transient evolution is driven by a "warm-up" time series $\tilde{\boldsymbol{u}}_\beta(t) = [\boldsymbol{u}(t), \beta(t)]^{\mathrm{T}}$ (containing τ data points); otherwise, if the mission is to infer the statistical properties of the system dynamics, namely, the dynamics "climate",[7] the reservoir is driven by only the parameter β during the transient period (also containing τ data points), while the input $\boldsymbol{u}_\beta(t)$ is replaced by the output $\boldsymbol{v}(t)$.

3. Results

We demonstrate the power of PARC by employing it to infer the dynamics of three classical chaotic systems, the logistic map, the Chua's circuit, and the Lorenz-96 climate network. In each case, the training data is composed of the time series of a handful of sampling states, while the mission is to obtain an optimized machine capable of not only replicating the observable dynamics of the sampling states, but also inferring the new dynamics not included in the training set.

3.1. Logistic map

The logistic map is described by the equation

$$x(n+1) = \beta x(n)[1 - x(n)], \tag{4}$$

where $x(n)$ is the system state at iteration n and $\beta \in [0,4]$ is the bifurcation parameter. Without loss of generality, we choose the sampling states as generated by parameters $\beta = 3.3$ (period-2 orbit), 3.5 (period-4 orbit), 3.6 (chaotic orbit) and 3.8 (chaotic orbit). For each of the sampling states, we obtain by numerical simulation a time sequence containing 3100 successive states. The initial condition of the map is chosen randomly within the range $(0,1)$, and data are collected after a transient of 100 iterations. Each sequence is divided into three segments. The first segment contains $\tau = 100$ data points, which will be used as the "warm-up" sequence for removing the influence of the initial conditions of the reservoir. The second segment contains $\hat{n} = 2000$ data points, which will be used as the training data for obtaining the output matrix. The third segment contains $\hat{n}' = 100$ data points, which will be used as the validating data for optimizing the machine hyperparameters. The concatenation of the time series of all $m = 4$ sampling states, together with the associated time series of the bifurcation parameter, gives the final datasets. Specifically, the dataset used in the training phase for evaluating the output matrix contains $\hat{L} = m \times \hat{n} = 4 \times 2000 = 8000$ data points, and the dataset used in the validating phase for optimizing the machine hyperparameters contains $\hat{L}' = m \times \hat{n}' = 4 \times 100 = 400$ data points.

The implementation of PARC starts with training the output matrix $\boldsymbol{W}_{\mathrm{out}}$. In doing this, we first construct a candidate machine according to a set of hyperparameters chosen randomly within a certain region in the parameter space. For simplicity, here we fix the size of the reservoir network as $D_{\mathrm{r}} = 100$, while optimizing the machine by tuning the other hyperparameters. The ranges over which the hyperparameters are tuned are $[0.01,1]$ for p (the density of the reservoir network), $[0,3]$ for σ (the range of the input matrix), $[0,3]$ for λ (the spectral radius of the reservoir), $[1\times10^{-10}, 1\times10^{-2}]$ for η (the regression coefficient), and $[0.01,1]$ for α (the leaking rate). Having constructed the candidate machine, we next operate the machine in the open-loop configuration [see Fig. 1(a)] to evaluate the output matrix according to Eq. (3). Before inputting the training data, the machine is first fed with the "warm-up" sequence in order to remove the impacts of the initial conditions, which is done for each of the training series. [Please note that the order of the sampling series does not affect the output matrix, as depicted by Eq. (3).] The training of the candidate machine is completed once the output matrix is obtained.

The next step is to optimize the machine based on the validation dataset. In this phase, the machine trained above is operating in the closed-loop configuration [see Fig. 1(c)], with the parameter input β being one of the sampling parameters [$\beta_i \in (3.3, 3.5, 3.6, 3.8)$]. Let $x_{\beta_i}(n)$ and $x'_{\beta_i}(n)$ be the ground truth and machine output associated with parameter β_i, respectively; the performance of the machine on the validation dataset is measured by the prediction error (the cost function)

$$L_1 = \left\langle \left| x_{\beta_i}(n) - x'_{\beta_i}(n) \right| \right\rangle_{n,\beta_i}, \tag{5}$$

which is averaged over the concatenated validation series (containing $\hat{L}' = 400$ data points). In our studies, we adopt the gradient descent algorithm in MATLAB to find the optimal machine hyperparameters. To implement the algorithm, we need to define the ranges of the hyperparameters and, at each step of the optimization process, input the hyperparameters of the candidate machine and the associated prediction error. The algorithm feeds back with a new set of hyperparameters. By the suggested hyperparameters, a new candidate machine will be constructed, trained (by the training dataset), and evaluated (by the validation dataset). The same process is repeated for $M = 400$ rounds, and the last feedback of the algorithm gives the optimal hyperparameters. In this case, the optimal hyperparameters are $(p, \sigma, \lambda, \eta, \alpha) = (0.215, 0.88, 5\times10^{-2}, 1\times10^{-5}, 1)$. Finally, the set of optimal

hyperparameters, together with the associated output matrix, defines the optimal machine, which will be used for prediction purposes.

In the prediction phase, if the mission is to forecast the further evolution of the system state, the machine is first driven by the "warm-up" series (containing $\tau = 100$ data points) in the open-loop configuration, and then driven by only the control parameter in the closed-loop configuration. If the mission is to infer the "climate" of the system dynamics associated with a specific bifurcation parameter, the machine is driven by only the control parameter. In the latter, predictions are made after a short transient ($\tau = 100$ iterations) in order to avoid the impact of the initial conditions of the reservoir.

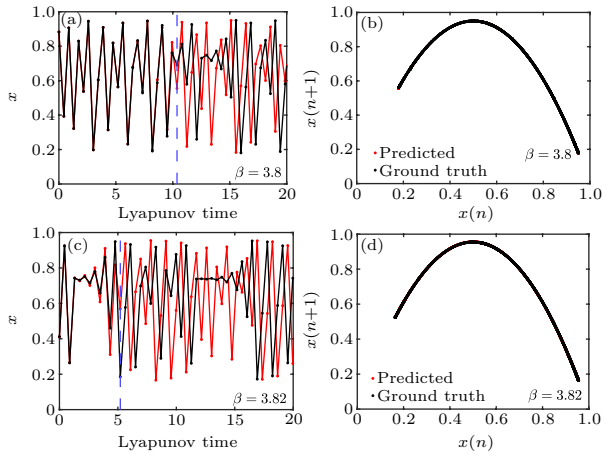


Fig. 2. Predicting the state evolution and replicating the dynamics "climate" of chaotic logistic map by PARC. (a) The predicted state evolution of the sampling state with parameter $\beta = 3.8$. The prediction horizon is about 10 Lyapunov times. (b) The returned map of the sampling state with parameter $\beta = 3.8$. (c) and (d) The results for the new state with parameter $\beta = 3.82$. The prediction horizon is about 5 Lyapunov times. Black dots are the ground truth. Red dots are results predicted by the machine.

We first demonstrate the capability of the machine in predicting the state evolution. Figure 2(a) shows the predictions (represented by red dots) made for the sampling state with the bifurcation parameter $\beta = 3.8$, together with the ground truth (represented by black dots) obtained based on model simulations. We see that the machine can predict accurately the system evolution for about 10 Lyapunov times. (The Lyapunov time is defined as the reciprocal of the largest Lyapunov exponent. For $\beta = 3.8$, the largest Lyapunov exponent of the logistic map is about $\Lambda = 0.432$.) To check whether the machine is capable of replicating the statistical properties of the system dynamics, we plot in Fig. 2(b) the returned map of the predicted states. We see that the returned maps of the predicted results and the ground truth well overlap, signifying that the dynamics "climate" is faithfully replicated. We proceed to demonstrate the capability of the machine in predicting the state evolution of a new state not included in the sampling set. Setting $\beta = 3.82$ as the input parameter (with which the largest Lyapunov exponent of the system dynamics is about $\Lambda = 0.434$), we plot in Figs. 2(c) and 2(d), respectively, the state evolution and returned map predicted by the machine. We see that the state evolution is accurately predicted for about 5 Lyapunov times, which is slightly worse than that of the sampling state shown in Fig. 2(a), but the returned map is still well replicated. The results in Fig. 2 suggest that the machine is able to forecast and replicate not only the dynamics of the sampling states, but also new dynamics not included in the sampling set.

We next employ the machine as a "generative model" to reconstruct the bifurcation diagram of the target system. In doing this, we scan β within the range $(3.2, 4)$ by the increment $\delta\beta = 1 \times 10^{-3}$ and, for each value of β, collect 500 successive outputs after a transient period of $\tau = 100$ predictions. The predicted results are plotted in Fig. 3(a). The ground truth is plotted in Fig. 3(b). We see that the predicted bifurcation diagram reproduces the ground truth accurately, including the locations of the bifurcation points, the positions of the periodic windows, and the structure of the "dark outlines" characterizing the unstable periodic orbits. It is worth mentioning that the bifurcation diagram shown in Fig. 3(a) is constructed by a single machine that is trained on the time series of only $m = 4$ sampling states (denoted by vertical dashed lines in the figure), signifying that the PARC machine is able to learn from data not only the dynamics of the sampling states, but also the variation of the system dynamics with respect to the bifurcation parameter.

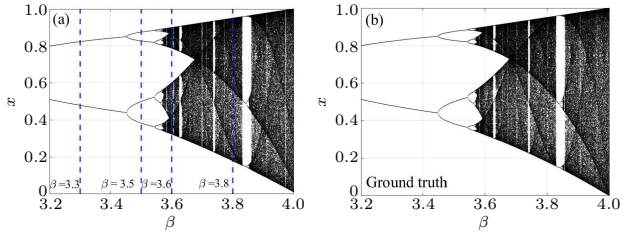


Fig. 3. Inferring the bifurcation diagram of logistic map by PARC. (a) The results predicted by the machine. Vertical dashed lines denote the sampling states. (b) The ground truth.

3.2. Chua's circuit

Our model of Chua's circuit reads

$$\begin{cases} \dot{x} = \beta[y - x - m_0 x + 0.5(m_0 - m_1)(|x+1| - |x-1|)], \\ \dot{y} = x - y + z, \\ \dot{z} = -33y, \end{cases} \tag{6}$$

with (m_0, m_1, β) the system parameters. In our studies, we fix $(m_0, m_1) = (-5/7, -8/7)$, while tuning β to generate different dynamics. In simulations, equation (6) is solved numerically by the 4th-order Runge–Kutta algorithm, with the time step $\delta t = 1 \times 10^{-3}$. The initial conditions of the system are randomly chosen from the range $(0, 1)$, and simulations are conducted for only the sampling parameters $\beta = 15.15$ (period-2), 15.323 (period-4), 15.496 (chaos), 15.67 (period-6). For

each of the sampling states, a transient period of $T = 100$ is discarded before collecting the data, and the data points, $\boldsymbol{u} = (x, y, z)$, are collected at intervals of 20 time steps. The state variables are normalized to be within the range $(0, 1)$. Each sampling series contains 5100 data points, among which the first segment of $\tau = 100$ points is used as the "warm-up" series (removing the impacts of the initial conditions of the reservoir network), the second segment containing $\hat{n} = 3000$ data points is used for training purposes, and the remaining data points are used for machine optimization. As there are $m = 4$ sampling states in total, the lengths of the concatenated training and validating series therefore are $\hat{L} = m \times \hat{n} = 12000$ and $\hat{L}' = m \times \hat{n}' = 8000$, respectively.

The training and validation processes are the same as for the logistic map, except that the cost function is replaced with

$$L_2 = \Big\langle [(x_{\beta_i}(n) - x'_{\beta_i}(n))^2 + (y_{\beta_i}(n) - y'_{\beta_i}(n))^2 + (z_{\beta_i}(n) - z'_{\beta_i}(n))^2]^{1/2} \Big\rangle_{n,\beta_i}, \tag{7}$$

where $[x_{\beta_i}(n), y_{\beta_i}(n), z_{\beta_i}(n)]$ and $[x'_{\beta_i}(n), y'_{\beta_i}(n), z'_{\beta_i}(n)]$ are the ground truth and machine output associated with parameter β_i, respectively. Still, the result is averaged over the concatenated time series ($n = 1, \ldots, \hat{L}$ for training and $n = 1 \ldots, \hat{L}'$ for validation purposes) and the sampling parameters $[\beta_i \in (15.15, 15.323, 15.496, 15.67)]$. For the sampling states chosen above, the optimal machine hyperparameters are $(p, \sigma, \lambda, \eta, \alpha) = (0.7477, 0.972, 2.3 \times 10^{-2}, 3 \times 10^{-4}, 0.5316)$ (the reservoir size is fixed as $D_{\rm r} = 500$). The optimal hyperparameters are obtained after 400 rounds of searching in the parameter space, and the searching ranges are $[0.01, 1]$ for p, $[0, 3]$ for σ, $[0, 3]$ for λ, $[1 \times 10^{-10}, 1 \times 10^{-2}]$ for η, and $[0.01, 1]$ for α. (The results are qualitatively the same when the hyperparameters are slightly changed.)

Figure 4(a) shows the time evolution of the x variable predicted by the machine for the sampling state of $\beta = 15.496$. For this parameter, the system presents the chaotic motion, with the largest Lyapunov exponent being about $\Lambda = 0.175$. We see that the state evolution is accurately predicted for more than 10 Lyapunov times. Based on the predictions, we plot in Fig. 4(b) the attractor replicated by the machine in the (x, y) plane. We see that the replicated attractor is consistent with the ground truth, suggesting that the machine also replicates the long-term, statistical properties of the system dynamics. To check the capability of the machine in inferring the new dynamics not included in the sampling set, we set $\beta = 15.5$ as the parameter input (for which the largest Lyapunov exponent of the system dynamics is about $\Lambda = 0.22$) and plot in Fig. 4(c) the time evolution of the x variable. (Before making predictions, the reservoir is driven by a "warm-up" series containing $\tau = 100$ data points generated from the model in order to remove the impacts of the initial conditions.) We see that for this new parameter, the prediction horizon is only about 3.5 Lyapunov times. Though the prediction horizon is significantly reduced, the machine is still capable of inferring the "climate" of the new dynamics, as depicted in Fig. 4(d).

Utilizing the machine as a "generative model" for inferring the system dynamics, we plot in Fig. 5(a) the bifurcation diagram constructed from the predictions. In doing this, we increase β from 15.1 to 15.7 by the increment $\delta\beta = 0.001$, and, for each value of β, record only the local maximum of the x variable predicted by the machine. Unlike the prediction of state evolution, here the machine is driven by only the parameter input in the "warm-up" stage (for $\tau = 100$ iterations), while the inputs of the state channel are replaced by the machine outputs. The ground truth obtained by model simulations is plotted in Fig. 5(b). We see that the bifurcation diagram reconstructed by the machine is in good agreement with the ground truth.

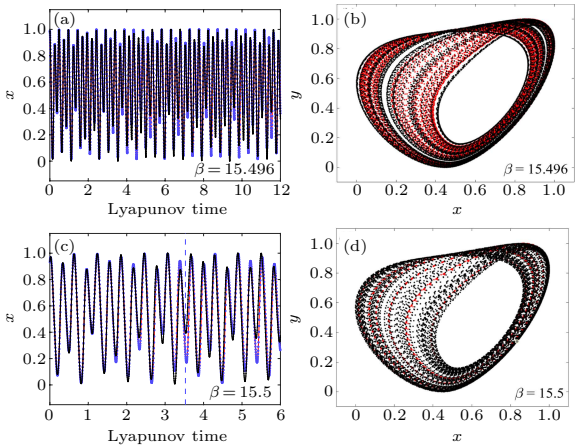


Fig. 4. Predicting the state evolution and replicating the "climate" of Chua's circuit by PARC. (a) The time evolution of the x variable predicted for the sampling state with parameter $\beta = 15.496$. The prediction horizon is more than 10 Lyapunov times. (b) The chaotic attractor inferred by the machine for the sampling state with parameter $\beta = 15.496$. (c), (d) The results for the new state with parameter $\beta = 15.5$. The prediction horizon is about 3.5 Lyapunov times. Black dots are the ground truth. Red dots are results predicted by the machine.

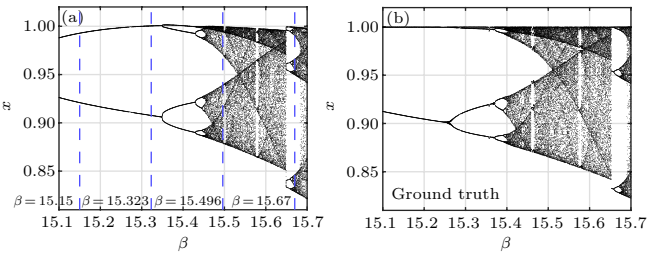


Fig. 5. Inferring the bifurcation diagram of Chua's circuit by PARC. (a) The results predicted by the machine. Vertical dashed lines denote the sampling states. (b) The ground truth.

3.3. Low-dimensional Lorenz-96 climate network

The Lorenz-96 climate network is represented by a ring of N identical dynamical nodes (units), with the dynamics of the ith node governed by the equation[36]

$$\dot{x}_i = x_{i-1}(x_{i+1} - x_{i-2}) - x_i + f(t). \tag{8}$$

Here, $i = 1,2,\ldots,N$ denotes the node index and $f(t) = \beta\sin(\omega t) + F$ is the driving force. Each node is coupled to three neighboring nodes on the ring, and the periodic boundary condition is adopted. In our studies, we set the network size as $N = 5$ and fix the parameters of the driving force as $(\omega, F) = (2,2)$, while treating the driving amplitude, β, as the bifurcation parameter of the system dynamics. Depending on the parameter β, the system can present periodic, quasi-periodic and chaotic motions. Our mission here is to reconstruct the bifurcation diagram of the climate network based on the time series of several sampling states.

In simulations, equation (8) is evolved by the 4th-order Runge–Kutta algorithm with the time step $\delta t = 1\times10^{-2}$. The sampling states are chosen at $\beta = 1.3$ (period-1), 2.0 (chaos), 2.5 (chaos), 3.0 (chaos), 3.5 (period-5), and 3.8 (chaos). For each of the sampling states, a transient period of $T = 100$ is discarded before collecting the data, and the data points, $\boldsymbol{u} = (x_1,\ldots,x_N)$, are collected at intervals of 5 time steps. The state variables are normalized to be within the range $(0,1)$. Each sampling series contains 8000 data points, among which $\hat{n} = 6000$ data points are used for training purposes and the remaining data points are used for machine optimization. As there are $m = 6$ sampling states in total, the lengths of the concatenated training and validating series therefore are $\hat{L} = m\times\hat{n} = 36000$ and $\hat{L}' = m\times\hat{n}' = 12000$, respectively. In searching for the optimal hyperparameters, we fix the reservoir size as $D_{\rm r} = 1000$, and the ranges over which the hyperparameters are scanned are the same as those in Chua's circuit. For the sampling states chosen, the optimal machine hyperparameters are $(p,\sigma,\lambda,\eta,\alpha) = (0.18,0.93,0.87,2.3\times10^{-3},0.15)$.

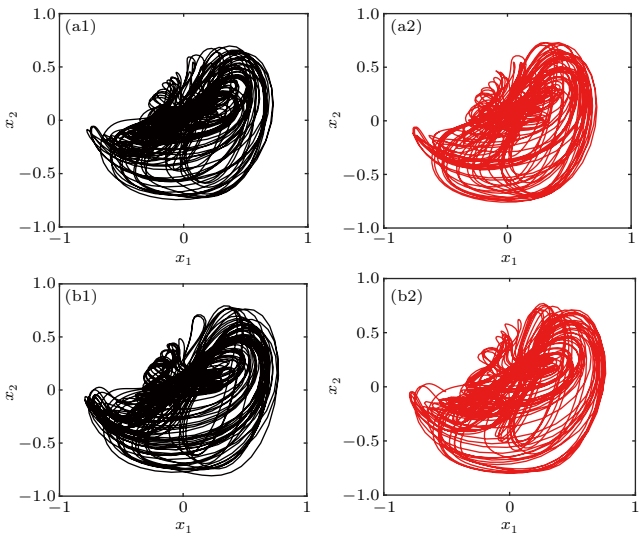


Fig. 6. Inferring the dynamics of the Lorenz-96 climate network containing $N = 5$ nodes by PARC. (a1) The attractor inferred by the machine for the sampling state $\beta = 2.5$. (a2) The ground truth of the sampling state $\beta = 2.5$. (b1) The attractor inferred by the machine for the parameter $\beta = 2.8$. (b2) The ground truth for the parameter $\beta = 2.8$.

Figure 6(a1) shows the attractor inferred by the machine under the sampling parameter $\beta = 2.5$. The results of model simulation are plotted in Fig. 6(a2). We see that the inferred attractor resembles the ground truth very well. Setting the control parameter to $\beta = 2.8$ (which is not included in the training set), we plot in Figs. 6(b1) and 6(b2) the attractor inferred by the machine and the ground truth, respectively. Again, we see that the ground-truth attractor is well inferred by the machine. Tuning β in the range $[1,4]$, we plot in Fig. 7(a) the bifurcation diagram inferred by the machine. Here, the bifurcation diagram is depicted by the local maximum of the variable x_1. The ground-truth results are shown in Fig. 7(b). We see that the inferred bifurcation diagram agrees with the ground truth reasonably. (The inferior performance around $\beta = 1.3$ is attributed to the complicated bifurcation behaviors of the system dynamics in the region $\beta\in(1,1.8)$.[36])

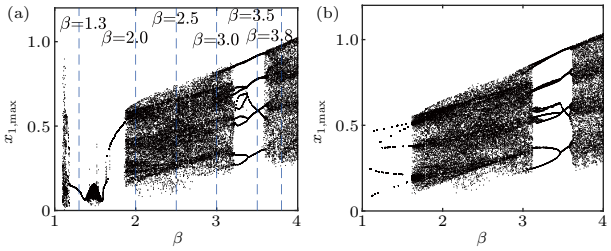


Fig. 7. Inferring the bifurcation diagram of the Lorenz-96 climate network of size $N = 5$ by PARC. (a) The results inferred by the machine. Vertical dashed lines denote the sampling states used for machine training and optimization. (b) The ground truth.

3.4. Impacts of sampling states

Besides the set of hyperparameters $(p,\sigma,\lambda,\eta,\alpha)$ associated with standard RC, an additional hyperparameter of the PARC machine is the sampling states, including the number and the location of the sampling states. As the success of PARC lies in its ability to "sense" the variation of the system dynamics with respect to the change of the bifurcation parameter, it is critically essential to "teach" the machine in an efficient manner by optimizing the sampling states. In general, better machine performance can be achieved by increasing the size of the training data, e.g., acquiring data from more sampling states, yet realistic concerns (e.g., experimental cost and limitations) render such an approach not always feasible. Given that the number of sampling states is fixed, a plausible approach to improve the machine performance would be choosing the sampling states appropriately. In what follows, we take the inference of the bifurcation diagram of Chua's circuit as an example, and demonstrate how the machine performance is affected by the number and location of the sampling states.

We demonstrate first how the machine performance is affected by the number, m, of the sampling states. Figure 8(a) shows the bifurcation diagram inferred by a new machine trained by the time series of $m = 3$ sampling states, $\beta = (15.15,15.41,15.67)$, with the length of each sampling series kept unchanged. (The system dynamics is chaotic for

$\beta = 15.41$.) Compared to the results shown in Fig. 5(a) (in which $m = 4$), we see that in Fig. 8(a) the bifurcation diagram in the region $\beta \in (15.65, 15.7)$ is clearly different from the ground truth [see Fig. 5(b)]. Figure 8(b) shows is the bifurcation diagram inferred by a new machine trained by the time series of $m = 5$ sampling states, $\beta = (15.15, 15.28, 15.41, 15.54, 15.67)$. (The system exhibits the period-4 and chaotic orbits for $\beta = 15.28$ and 15.54, respectively.) We see that compared with the results shown in Fig. 5(a) (in which $m = 4$), the accuracy of the inferred bifurcation diagram is improved. Specifically, the crisis transition from chaos to period-3 orbit (around $\beta = 15.65$) is well reproduced in Fig. 8(b), which is not well captured in Fig. 5(a). Results thus show that by increasing the number of sampling states, the machine performance can be improved.

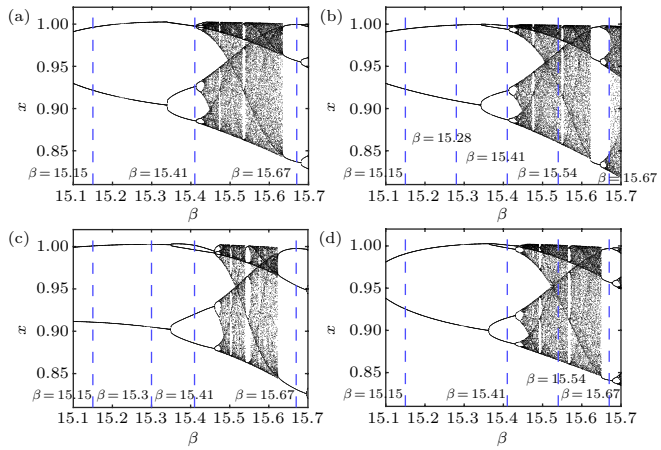


Fig. 8. Impacts of sampling states on the inference of the bifurcation diagram of Chua's circuit. (a) The $m = 3$ sampling states are chosen at $\beta = (15.15, 15.41, 15.67)$. The optimal machine hyperparameters are $(p, \sigma, \lambda, \eta, \alpha) = (0.553, 1.033, 2.589, 5 \times 10^{-4}, 0.525)$. (b) The $m = 5$ sampling states are chosen at $\beta = (15.15, 15.28, 15.41, 15.54, 15.67)$. The optimal machine hyperparameters are $(p, \sigma, \lambda, \eta, \alpha) = (0.425, 1.385, 0.558, 1.8 \times 10^{-3}, 0.298)$. (c) The $m = 4$ sampling states are chosen at $\beta = (15.15, 15.3, 15.41, 15.67)$. The optimal machine hyperparameters are $(p, \sigma, \lambda, \eta, \alpha) = (0.442, 1.197, 1.51, 4 \times 10^{-4}, 0.46)$. (d) The $m = 4$ sampling states are chosen at $\beta = (15.15, 15.41, 15.54, 15.67)$. The optimal machine hyperparameters are $(p, \sigma, \lambda, \eta, \alpha) = (7.2 \times 10^{-3}, 1.851, 0.784, 7.4 \times 10^{-3}, 0.474)$. Vertical dashed lines denote the locations of the sampling states in the parameter space.

We demonstrate next how the machine performance is affected by the location of the sampling states. Figure 8(c) shows the bifurcation diagram inferred by a new machine trained by the time series of $m = 4$ sampling states, $\beta = (15.15, 15.3, 15.41, 15.67)$. (The system presents the period-3 orbit for $\beta = 15.3$.) Compared to the results shown in Fig. 5(a), here the number of sampling states is still $m = 4$, but the locations of the 2nd and 3rd sampling states have been changed. In particular, the 2nd sampling state is changed from $\beta = 15.323$ (period-4) to 15.3 (period-4), while the 3rd sampling state is changed from $\beta = 15.496$ (chaos) to 15.41 (chaos). Compared to the results shown in Fig. 5(a), we see that in Fig. 8(c) the bifurcation diagram is clearly different from the ground truth [see Fig. 5(b)] in the region $\beta \in (15.65, 15.7)$. Changing the 2nd and 3rd sampling states to $\beta = 15.41$ (chaos) and to 15.54 (chaos), respectively, we plot in Fig. 8(d) the bifurcation diagram inferred by the new machine trained by the new data. Compared to the results shown in Fig. 5(a) (where $\beta \in (15.15, 15.323, 15.496, 15.67)$), we see that in Fig. 8(d) the bifurcation diagram is slightly improved, e.g., the crisis transition around $\beta = 15.65$ is smooth in Fig. 8(d) but is discontinous in Fig. 5(a). Clearly, the machine performance depends on the location of the sampling states.

Combining the results shown in Figs. 5(a) and 8, we have the following suggestions for the selection of the sampling states. First, the sampling states should be distributed evenly over the parameter space, as interpolations seem better than extrapolations in inferring new system dynamics. This feature can be partially seen by comparing Fig. 5(a) (in which the sampling states are evenly distributed in the parameter space) with Fig. 8(c) (in which the sampling states are unevenly distributed). Second, the sampling states should cover different dynamics regimes. While chaotic orbits do provide rich information about the system dynamics, periodic orbits are also important for the machine to "sense" the variation of the system dynamics with the bifurcation parameter. Comparing the results in Fig. 8(a) (in which the sampling states include 2 periodic states and 1 chaotic state) and Fig. 5(a) (in which the sampling states include 3 periodic states and 1 chaotic state), we see that in the latter the machine performance is significantly improved by adding a periodic state. Finally, while the machine performance can be improved in general by adopting more sampling states [as depicted in Fig. 8(b)], this approach will largely increase the cost (including data acquisition and computational resources), making the optimization of the sampling states an interesting question warranting further study.

4. Discussion

Whereas growing attention has been paid to PARC in recent years, and the power of this technique has been well demonstrated in a variety of applications, open questions have also arisen concerning the working mechanism, algorithm design, and application of PARC. Some of the questions are as follows.

(i) The working mechanism of PARC The merit of PARC lies in the generalization ability of the knowledge from the sampling states to the new states, namely, the ability of knowledge transfer.[11] This ability, which is lacking in standard RC, is realized by the parameter-control channel in PARC. While evidences show that the dynamics of the reservoir network undergo rich bifurcation behaviors when the control parameter (or some of the machine hyperparameters) is

changed, it remains not clear how the control parameter participates in the training and predicting processes. For instance, the ability of knowledge transfer indicates that the machine might learn the dynamical functions and the bifurcation parameter of the target system separately, yet the coupling mechanism between the dynamical functions and the bifurcation parameter is still unknown. For this reason, we are unable to tell whether the PARC technique is applicable when facing a new dynamical system, and, if applicable, how to optimize the location of the sampling states in preparing the training data.

(ii) Inferring the dynamics of high-dimensional complex systems Though the PARC technique has been justified as efficient for inferring the low-dimensional dynamical systems, its applicability to high-dimensional, spatiotemporal chaotic systems is yet to be checked. In particular, many real-world systems are represented by heterogeneous complex networks of coupled dynamical elements, in which not only the local connectivity of the system elements is of large variation, but also the local dynamics are different from each other. Data-based inference of high-dimensional complex systems naturally is the next challenge faced by researchers in the community. Recently, a parallel RC scheme has been proposed for learning high-dimensional dynamical systems,[20,21] but it applies to only homogeneous systems. The PARC technique, probably with some modifications, provides a feasible solution to the inference of heterogeneous complex systems, e.g., utilizing the node degree or the nodal parameter as the control parameter of PARC.

(iii) The generalized schemes of PARC While PARC is originally designed for inferring new dynamics not included in the training set, it can also be exploited for accomplishing other tasks. For instance, operating the machine in the reverse direction (i.e., using the time series as the input and the control parameter as the output), the machine can be utilized as an identifier to find the parameter generating the time series or trace the variation of the system parameter according to the non-stationary time series.[33] Replacing the sampling states with different chaotic attractors and the parameter input with attractor labels, the machine can also be exploited to replicate multiple systems described by entirely different dynamics, namely, the multi-functional RC.[34,35] Last but not least, it is also intriguing to see whether the generalization ability of the other RNN techniques can be improved by incorporating the parameter-aware scheme, e.g., the long short-term memory (LSTM) technique, the gated recurrent unit (GRU) technique, and the next-generation RC.[37]

(iv) Application to real-world systems The existing studies of PARC are mainly based on artificial data generated by modeling systems, and the applicability of the technique to real-world complex systems is largely unknown. Different from modeling systems, data acquired from realistic systems contain noise perturbations or even errors. In addition, for large-scale complex systems, usually the measured data contain only partial information about the system dynamics and, more importantly, the system dynamics might be non-stationary in the sense that the system bifurcation parameter could be changing with time. These concerns raise new challenges for the application of PARC in real-world systems, calling for the development of more sophisticated techniques.

To summarize, we have introduced a new technique of machine learning, namely, PARC, developed recently in the literature for inferring chaotic dynamics, and demonstrated the application of the technique by employing it to reconstruct the bifurcation diagram of typical chaotic systems based on the time series of a few sampling states. We have given a detailed description of the structure of the machine, showcased the machine implementation, and briefly analyzed the impacts of the sampling states on the machine performance. The new technique opens up new avenues for data-based analysis of chaotic systems, and is expected to have broad applications. We hope the present work could draw attention from researchers in the fields of machine learning and complex systems on this new machine-learning technique, and stimulate further studies along this direction.

Program availability

The source code is openly available in the Science Data Bank at https://doi.org/10.57760/sciencedb.20953.

Acknowledgments

Project supported by the National Natural Science Foundation of China (Grant No. 12275165). XGW was also supported by the Fundamental Research Funds for the Central Universities (Grant No. GK202202003).

References

[1] Ott E 2022 *Chaos in Dynamical Systems* 2nd Edn. (Cambridge: Cambridge University Press)
[2] Strogatz S H 2015 *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* (CRC Press)
[3] Wang W X, Yang R, Lai Y C, Kovanis V and Grebogi C 2011 *Phys. Rev. Lett.* **106** 154101
[4] Lai Y C 2021 *Chaos* **31** 082101
[5] Maass W, Natschlager T and Markram H 2002 *Neural Comput.* **14** 2531
[6] Jaeger H and Haas H 2004 *Science* **304** 78
[7] Pathak J, Lu Z, Hunt B, Girvan M and Ott E 2017 *Chaos* **27** 121102
[8] Lu Z, Hunt B R and Ott E 2018 *Chaos* **28** 061104
[9] Weng T, Yang H, Gu C, Zhang J and Small M 2019 *Phys. Rev. E* **99** 042203
[10] Fan H, Jiang J, Zhang C, Wang X G and Lai Y C 2020 *Phys. Rev. Res.* **2** 012080
[11] Guo Y L, Zhang H, Wang L, Fan H, Xiao J H and Wang X G 2021 *Chaos* **31** 011104
[12] Wang L, Fan H, Xiao J, Lan Y and Wang X G 2022 *Phys. Rev. E* **105** L052201
[13] Kong L W, Weng Y, Glaz B, Haile M and Lai Y C 2023 *Chaos* **33** 033111

[14] Du Y, Li Q, Fan H, Zhan M, Xiao J and Wang X G 2024 *Phys. Rev. Res.* **6** 013181
[15] Lin Z, Lu Z, Di Z and Tang Y 2024 *Nat. Commun.* **15** 6584
[16] Panahi S and Lai Y C 2024 *Chaos* **34** 051501
[17] Lukosevicius M and Jaeger H 2009 *Comput. Sci. Rev.* **3** 127
[18] Tanaka G, Yamane T, Heroux J B, Nakane R, Kanazawa N, Takeda S, Numata H, Nakano D and Hirose A 2019 *Neural Networks* **115** 100
[19] Yan M, Huang C, Bienstman P, Tino P, Lin W and Sun J 2024 *Nat. Commun.* **15** 2056
[20] Pathak J, Hunt B, Girvan M, Lu Z and Ott E 2018 *Phys. Rev. Lett.* **120** 024102
[21] Szunyogh I, Arcomano T, Pathak J, Wikner A, Hunt B and Ott E 2020 *Geophys. Res. Lett.* **47** e2020GL087776
[22] Klos C, Kossio Y F K, Goedeke S, Gilra A and Memmesheimer R M 2020 *Phys. Rev. Lett.* **125** 088103
[23] Zhao H 2021 *Sci. China-Phys. Mech. Astron.* **64** 270511
[24] Kong L W, Fan H, Grebogi C and Lai Y C 2021 *Phys. Rev. Res.* **3** 013090
[25] Fan H, Kong L W, Lai Y C and Wang X G 2021 *Phys. Rev. Res.* **3** 023237
[26] Xiao R, Kong L W, Sun Z K and Lai Y C 2021 *Phys. Rev. E* **104** 014205
[27] Zhang H, Fan H, Wang L and Wang X G 2021 *Phys. Rev. E* **104** 024205
[28] Kim J Z, Lu Z, Nozari E, Pappas G J and Bassett D S 2021 *Nat. Mach. Intell.* **3** 316
[29] Zhang H, Fan H, Du Y, Wang L and Wang X G 2022 *Chaos* **32** 083136
[30] Fan H, Wang L, Du Y, Wang Y, Xiao J and Wang X G 2022 *Phys. Rev. Res.* **4** 013137
[31] Luo H, Du Y, Fan H, Wang X, Guo J and Wang X G 2024 *Phys. Rev. E* **109** 024210
[32] Sisodia D and Jalan S 2024 *Phys. Rev. E* **110** 034211
[33] Zhai Z M, Moradi M, Blaz B, Haile M and Lai Y C 2024 *Phys. Rev. Res.* **6** 013196
[34] Kong L W, Brewer G A and Lai Y C 2024 *Nat. Commun.* **15** 4840
[35] Du Y, Luo H, Guo J, Xiao J, and Wang X G 2024 arXiv:2409.16719 [nlin.CD]
[36] Lorenz E N 1996 *Predictability: A problem partly solved, in Proceedings Seminar on Predictability (ECMWF)* Vol. 1
[37] Gauthier D J, Bollt E, Griffith A and Wendson A S B 2021 *Nat. Commun.* **12** 5564