Design of a prediction system based on the dynamical feed-forward neural network

Xiaoxiang GUO\(^1,2\), Weimin HAN\(^2\) & Jingli REN\(^1\)*

\(^1\)Henan Academy of Big Data, Zhengzhou University, Zhengzhou 450001, China; \(^2\)Department of Mathematics, University of Iowa, Iowa City IA 52242, USA

Received 20 May 2020/Revised 26 October 2020/Accepted 21 December 2021/Published online 11 October 2022

Abstract Analysis and prediction of time series play a significant role in scientific fields of meteorology, epidemiology, and economy. Efficient and accurate prediction of signals can give an early detection of abnormal variations, provide guidance on preparing a timely response and avoid presumably adverse impacts. In this paper, a prediction system is designed based on the dynamical feed-forward neural network. The trajectory information in the reconstructed phase space, which is topologically equivalent to the dynamical evolution of the system, is applied to establish the prediction model. Moreover, an integer constrained particle swarm optimization algorithm is employed to select the optimal time delay, which is the parameter of our system. Simulation results for applications on the Lorenz system, stock market index, and influenza data indicate that our proposed method can produce efficient and reliable predictions.

Keywords prediction system, phase space reconstruction, topological equivalence, dynamical feed-forward neural network, integer constrained particle swarm optimization algorithm

Citation Guo X X, Han W M, Ren J L. Design of a prediction system based on the dynamical feed-forward neural network. Sci China Inf Sci, 2023, 66(1): 112102, https://doi.org/10.1007/s11432-020-3402-9

1 Introduction

Data-driven predictions pervade nearly all aspects of modern science, with applications in the industry [1,2], agriculture [3], economy [4,5], meteorology [6–8], and epidemiology [9,10]. Essential information mined from data leads to predictability of scientific discovery, and provides future opportunities to improve predictions derived from the science of science [11]. Creative and insightful methods were proposed for prediction based on the time series in the last several decades.

In 1987, Farmer and Sidorowich [12] pioneered the work in this field by constructing a predictor to forecast the single variable chaotic time series. The time series was first reconstructed in a multi-dimensional phase space based on the lagged coordinates technique, and then the global or local neighbor information of the trajectory evolution in the reconstructed phase space was utilized to find and optimize the predictor. In the reconstructed phase space, Sugihara and May [13] defined a simplex projection to make short-term prediction about the trajectories of chaotic dynamic systems. This method can be applied to distinguish deterministic chaos from measurement error or environment error in the time series.

Inspired by the generalized embedding theorem from [14], Ye and Sugihara [15] developed a new idea and proposed the multi-view embedding (MVE) method to predict the multi-dimensional time series. The MVE is particularly effective in overcoming limitations of short and noisy time series. However, this method is computationally intensive if the embedding dimension is relatively high. There are \(C_{nτ}^m - C_{n(τ-1)}^m\) phase spaces to be constructed, where \(τ\) is the time delay (an integer), \(m\) is the embedding dimension, \(n\) is the number of variables, and \(C_{nτ}^m\) denotes the number of combinations picking \(m\) out of \(nτ\). Later on, Ma et al. [16] proposed a randomly distributed embedding (RDE) framework for predictions with datasets involving high-dimensional variables but short-term time series points. This RDE framework maps the randomly generated low-dimensional non-delay embedded phase to a delay

* Corresponding author (email: renjl@zzu.edu.cn)
embedded phase space constructed by the target variable which needs to be predicted. Each of these mappings can perform as a weak predictor which patches pieces of association information from various embeddings into a whole dynamics of the target variable. Then, a stronger predictor is built by an aggregated average strategy. The RDE framework can reduce the computational complexity required by the MVE method. Recently, Guo et al. [17] proposed a delay parameterized method (DPM) to predict low dimensional mid-term chaotic time series. The predictor was obtained from the correlation between non-delay reconstructed state space and delay reconstructed phase space which is time delay dependent. The piecewise mapping strategy reduces the number of model parameters in high-dimensional feature classification and avoids the risk of parameter explosion in fine-grained image recognition [18].

With the advancement of computer science and technology, machine learning offers new tools for researchers to understand and predict human behavior [19]. Jaeger and Haas [20] presented the echo state networks (ESNs) to learn the black-box models of nonlinear systems and then to predict chaotic systems. Application on equalizing a wireless communication channel showed that an improvement of two orders of magnitude of signal error rate was achieved. Recently, state-of-the-art sequential deep-learning models, including long short-term memory (LSTM) and attention-based recurrent neural network (RNN), have been applied to a variety of important problems, such as the speech recognition [21], COVID-19 prediction [22], influenza epidemic prediction [23], and traffic flow prediction [24]. However, the required computational time increases rapidly for many tasks with growing demands for improved performance in the state-of-the-art systems. Wang et al. [25] designed a dynamically routed network in the convolutional neural network (CNN) to reduce the average cost of model inference while retaining the accuracy of the full network. The dynamical routing policy through hybrid reinforcement learning determines particular layers of CNN to be included when a given image is processed. Bolukbas et al. [26] proposed an adaptive neural network for efficient inference. The adaptive early-exit strategy allows easy examples to bypass some of the network’s layers, and the adaptive network selection policy chooses the optimal route from a set of pre-trained networks to reduce the computational time. Huang et al. [27] used a two-dimensional multi-scale network architecture, which maintains coarse and fine level features all-throughout the network, on image classification. It is shown that the framework can improve the existing state-of-the-art systems.

By combining the phase space reconstruction technique with the machine learning technique, we design a dynamical feed-forward neural network for prediction in this paper. The dynamics of the target variable described by the delay embedding is treated as the output signal of the neural network, and the parameters in the system are optimized by an integer constrained particle swarm optimization algorithm. The rest of this paper is organized as follows. In Section 2, we introduce the phase space reconstruction according to the embedding theory. In Section 3, we present the system model and our proposed dynamical feed-forward neural network scheme. In Section 4, we show three examples for the application of our prediction system. In Section 5, we give a summary of the results of this paper and discuss about future work.

2 Phase space reconstruction and time delay effect

In this section, we introduce the phase space reconstruction technique by delay coordinates [12, 28]. For a time series \( \{x_1, x_2, \ldots, x_N\} \), the reconstructed phase space is defined as

\[
R = \begin{pmatrix}
x_1 & x_2 & \cdots & x_{N-(m-1)\tau} \\
x_1+\tau & x_{2+\tau} & \cdots & x_{N-(m-2)\tau} \\
\vdots & \vdots & \ddots & \vdots \\
x_1+(m-1)\tau & x_{2+(m-1)\tau} & \cdots & x_N
\end{pmatrix},
\]

where \( N \) is the length of time series, \( m \) is the embedding dimension of the phase space, \( \tau \) is the time delay (\( \tau \) is an integer). The number of points in this reconstructed phase space is \( M = N - (m-1)\tau \), and each point is an \( m \)-dimensional vector. According to Takens’ embedding theorem [29] and the generalized embedding theorem [14], the trajectories of the reconstructed phase space are topologically equivalent to that of the dynamical evolution of the system. For better understanding, we take the Lorenz system as an example to show this topological equivalence.

Observations from the Lorenz system can be regarded as the time series obtained by solving the
following first-order system of three ordinary differential equations:

\[
\begin{align*}
\dot{x} &= \alpha (y - x), \\
\dot{y} &= (\beta - \gamma) x - y, \\
\dot{z} &= x y - \gamma z.
\end{align*}
\]  

(2)

In our simulation, we choose $\alpha = 10$, $\beta = 28$, $\gamma = 8/3$. The butterfly-like attractor of the Lorenz system is shown in Figure 1(a). The reconstructed phase space is constructed by the delay coordinates corresponding to the variable $y$, i.e., $R = (y(i), y(i + \tau), y(i + 2\tau))^T$. Figure 1(b) shows the phase diagram of the reconstructed phase space. By Takens’ embedding theorem, the dynamics between the attractor of the Lorenz system and the reconstructed phase space are topologically equivalent.

There are two crucial parameters in the phase space reconstruction: the embedding dimension $m$ and the time delay $\tau$ [30]. The embedding dimension corresponds to the number of independent quantities needed to specify the state of the system at any given instant. If $m$ is too small, the attractor cannot be completely expanded, yet a large $m$ will increase the redundancy. Define a $k$th-order conditional probability distribution of a coordinate $x$, $P(x|x_1, x_2, \ldots; \tau)$, as the probability of observing the value $x$ given that $x_1$ was observed at time $\tau$ before, $x_2$ was observed at time $2\tau$ before, and so on. If we take $\tau$ to be small, the $k$ conditions are nearly equivalent to the specification of the value of $x$ at some time along with values of all its derivatives up to order $k - 1$. The points $X(t)$ and $X(t + \tau)$ in the reconstructed
phase space cannot be separated. If \( \tau \) is too large, \( X(t) \) and \( X(t + \tau) \) will be less correlated, and the information generated from the flow properties would randomize the samples with respect to each other. The reconstructed phase space cannot accurately reflect the evolution rules about the attractor unless the time delay and the embedding dimension are chosen appropriately.

From Figure 2 on the phase diagrams of the phase space reconstructed with different time delays, we observe that the attractor cannot fully stretch when \( \tau \) equals 1, whereas the attractor loses some correlation information when \( \tau \) equals 28. In other words, if \( \tau \) is too small, \( y(i), y(i + \tau) \), and \( y(i + 2\tau) \) cannot be separated, and the attractor cannot fully stretch; whereas if \( \tau \) is too large, \( y(i), y(i + \tau) \), and \( y(i + 2\tau) \) will be less correlated, and some evolution information will be lost. Thus, the selection of the value of the time delay is crucial for a good phase space reconstruction. Commonly used methods for choosing the time delay are autocorrelation method \([31]\) and mutual information method \([32]\). False neighbors method \([33]\) and Cao’s method \([34]\) are usually applied to identify the embedding dimension. Those methods are classical to reconstruct the phase space on analysing the stability of the dynamics \([35-38]\).

However, the methods may not be suitable for time series predicting, and other methods such as traversal algorithm, artificial intelligence algorithm should be applied in selecting the optimal parameters of the reconstruction for prediction \([17]\). In the following section, we design a prediction system based on the dynamical feed-forward neural network (DFNN) scheme. The time delay is the parameter to be selected by an integer constrained particle swarm optimization (ICPSO) algorithm.

3 The prediction system model and our proposed DFNN scheme

The predictability of a system is believed to come from various kinds of information hidden in the time series. Information of dynamical evolution of the system can provide a plentiful source in designing the prediction system. In Section 2, we have discussed the reconstructed phase space which is topologically equivalent to the dynamics of the system. In this section, we establish a prediction system based on the dynamical information.

A system’s outputs as the observations are multi-dimensional coupling time series. Thus, the essence of this prediction system is multi-dimensional time series forecasting. Limitations of the data should be discussed before the system modeling.

1. The dimension of the observations must be larger than one, each one-dimensional time series is treated as a variable of the system, and all the variables must be related.

2. The length of the time series should not be too long or too short. Short-length data does not have enough essential information of the system, whereas long-length data increases information redundancy. In our predicting scheme, the length is limited to several hundred.

3. Magnitudes of the observations should be on the same scale. Normalization is needed if the magnitudes are in different scales.

3.1 Prediction system and DFNN

The DFNN scheme is designed to bridge the observations and the dynamical information by feed-forward network. Radial basis function network is an artificial neural network that uses radial basis functions as activation functions; the output of this network is a linear combination of radial basis functions of the inputs and neuron parameters \([39]\). In our scheme, the inputs are the observations from the original system, and the output is designed to be the dynamical evolution of the system trajectory.

Assume the observations of the system are \( \{x_k(i) : 1 \leq i \leq N, 1 \leq k \leq m\} \). The dimension of the observations is \( m \), and the length of the time series is \( N \). To avoid the calculation error caused by the difference on the magnitude scale, the observations are normalized: \( s_k(i) = x_k(i)/(\max(x_k) - \min(x_k)) \).

Here, \( \max(x_k) \) and \( \min(x_k) \) represent the maximum and minimum of the variable \( x_k \), respectively. So the inputs of the prediction system are

\[
\text{inputs} = \begin{cases}
  s_1(1), s_1(2), \ldots, s_1(N), \\
  s_2(1), s_2(2), \ldots, s_2(N), \\
  \vdots \\
  s_m(1), s_m(2), \ldots, s_m(N).
\end{cases}
\]
The phase space reconstructed by the k-th variable \( s_k, \ k = 1, 2, \ldots, m \), which indicates a piece of dynamical information of the system’s trajectory, can be defined as

\[
\mathbf{S}_k = \begin{pmatrix}
    s_k(1) & s_k(2) & \cdots & s_k(N - (d - 1)\tau) \\
    s_k(1 + \tau) & s_k(2 + \tau) & \cdots & s_k(N - (d - 2)\tau) \\
    s_k(1 + 2\tau) & s_k(2 + 2\tau) & \cdots & s_k(N - (d - 3)\tau) \\
    \vdots & \vdots & \ddots & \vdots \\
    s_k(1 + (d - 1)\tau) & s_k(2 + (d - 1)\tau) & \cdots & s_k(N) 
\end{pmatrix}, \quad (4)
\]

where \( \tau \) is the time delay, \( d \) is the embedding dimension, \( N \) is the length of the time series. There are \( M = N - (d - 1)\tau \) points of \( d \)-dimensions in the reconstructed phase space. Tracking the evolution over the time, the state space is

\[
\hat{\mathbf{S}}_k = \begin{pmatrix}
    s_k(M + 1) & s_k(M + 2) & \cdots & s_k(N) \\
    s_k(M + 1 + \tau) & s_k(M + 2 + \tau) & \cdots & s_k(N + \tau) \\
    s_k(M + 1 + 2\tau) & s_k(M + 2 + 2\tau) & \cdots & s_k(N + 2\tau) \\
    \vdots & \vdots & \ddots & \vdots \\
    s_k(M + 1 + (d - 1)\tau) & s_k(M + 2 + (d - 1)\tau) & \cdots & s_k(N + (d - 1)\tau) 
\end{pmatrix}. \quad (5)
\]

It should be noted that the values \( \{s_k(N + j) : 1 \leq j \leq (d - 1)\tau\} \) in \( \hat{\mathbf{S}}_k \) are unknown. These are the values we want to predict. In this paper, we use the feed-forward neural network to train the network, where the first \( M \) epochs of the observations are taken as the input, and \( \mathbf{S}_k \) is chosen as the output. Taking the radial basis function neural network (RBFNN) as an example, we employ the MATLAB function \texttt{newrb} to design this radial basis network. The function \texttt{newrb} iteratively creates a radial basis network with one neuron at a time. Neurons are added to the network until the specified mean squared error falls below an error tolerance or a maximum number of neurons has been reached\(^1\). The syntax of this function is

\[
\text{net} = \text{newrb}(\mathbf{P}, \mathbf{T}, \text{goal}, \text{spread}, \text{mn}, \text{df}),
\]

where \( \text{net} \) is the desired network, \( \mathbf{P} \) is the \( R \times Q \) matrix of \( Q \) input vectors, \( \mathbf{T} \) is the \( S \times Q \) matrix of \( Q \) target vectors, \( \text{goal} \) is the mean squared error goal, \( \text{spread} \) is the spread of radial basis functions, \( \text{mn} \) is the maximum number of neurons, and \( \text{df} \) is the number of neurons to add between displays. The larger spread is, the smoother the function approximation. To see the training performance, the function \( \text{train} = \text{sim}(\text{net}, \mathbf{P}) \) is needed. Here \( \text{train} \) is the training result corresponding to \( \mathbf{T} \).

In our DFNN scheme,

\[
\mathbf{P} = \begin{pmatrix}
    s_1(1) & s_1(2) & \cdots & s_1(M) \\
    s_2(1) & s_2(2) & \cdots & s_2(M) \\
    \vdots & \vdots & \ddots & \vdots \\
    s_m(1) & s_m(2) & \cdots & s_m(M) 
\end{pmatrix}, \quad (6)
\]

is an \( m \times M \) matrix of \( M \) input vectors, and \( \mathbf{Q} = \mathbf{S}_k \) is a \( d \times M \) matrix of \( M \) output vectors. The dynamical information hidden in the matrix \( \mathbf{S}_k \) is involved in our scheme. It should be emphasized that two parameters, the embedding dimension \( d \) and the time delay \( \tau \) need to be selected. The embedding dimension is related to the degree of freedom of the system. In this paper, we fix the value of \( d \) to the dimension of the observations, \( m \), and choose the time delay as a parameter depending on the time series. We comment that further work can be done on the selection of embedding dimension for the model optimization.

\(^1\) https://www.mathworks.com/help/deeplearning/ref/newrb.html.
Assume the training results are

\[
S'_k = \begin{pmatrix}
  s'_k(1) & s'_k(2) & \cdots & s'_k(N - (d - 1)\tau) \\
  s'_k(1 + \tau) & s'_k(2 + \tau) & \cdots & s'_k(N - (d - 2)\tau) \\
  s'_k(1 + 2\tau) & s'_k(2 + 2\tau) & \cdots & s'_k(N - (d - 3)\tau) \\
  \vdots & \vdots & \ddots & \vdots \\
  s'_k(1 + (d - 2)\tau) & s'_k(2 + (d - 2)\tau) & \cdots & s'_k(N)
\end{pmatrix}. \tag{7}
\]

Selection of the time delay is based on minimizing the training error. For this purpose, we define a fitness function:

\[
f = \frac{1}{m} \sum_{k=1}^{m} \text{mean} \left( \left| \frac{s'_k(\cdot) - s_k(\cdot)}{s_k(\cdot)} \right| \right) / \text{std} \left( \left| \frac{s'_k(\cdot) - s_k(\cdot)}{s_k(\cdot)} \right| \right). \tag{8}
\]

Here, \( \text{mean}(w(\cdot)) \) and \( \text{std}(w(\cdot)) \) are the mean value and standard deviation of \( \{w(j) : M + 1 \leq j \leq N\} \). \tag{9}

This fitness function is one kind of normalized error, and formally it is similar to the Gaussian function. The optimal time delay minimizes the fitness function. After the network is obtained, a matrix \( \hat{P} \) composed of the \((M + 1)\)-th to \(N\)-th epochs of the input vectors is applied as new inputs of the network to predict the target vector \( \hat{S}_k \). We have

\[
\hat{P} = \begin{pmatrix}
  s_1(M + 1) & s_1(M + 2) & \cdots & s_1(N) \\
  s_2(M + 1) & s_2(M + 2) & \cdots & s_2(N) \\
  \vdots & \vdots & \ddots & \vdots \\
  s_m(M + 1) & s_m(M + 2) & \cdots & s_m(N)
\end{pmatrix}. \tag{10}
\]

The output matrix is defined as

\[
\hat{S}'_k = \begin{pmatrix}
  s'_k(M + 1) & s'_k(M + 2) & \cdots & s'_k(N) \\
  s'_k(M + 1 + \tau) & s'_k(M + 2 + \tau) & \cdots & s'_k(N + \tau) \\
  s'_k(M + 1 + 2\tau) & s'_k(M + 2 + 2\tau) & \cdots & s'_k(N - 2\tau) \\
  \vdots & \vdots & \ddots & \vdots \\
  s'_k(M + 1 + (d - 2)\tau) & s'_k(M + 2 + (d - 2)\tau) & \cdots & s'_k(N + (d - 2)\tau)
\end{pmatrix}. \tag{11}
\]

Note that the procedure iteratively increases one predicted value \( s'_k(N + 1) \) of each vector at one iteration. Here, \( s'_k(N + 1) \) is the first value on the last line of the matrix \( \hat{S}'_k \). Then, these predicted values are added to the observations for the next iteration process. To predict \( n \) values of the observation, this procedure needs \( n \) iterations. Finally, the predicted values \( s'_k(N + j), 1 \leq j \leq n, 1 \leq k \leq m \), are de-normalized to \( x'_k(N + j) = s'_k(N + j) \times (\max(x_k) - \min(x_k)) \). The prediction error is defined as

\[
\text{Error}(j) = \left| \frac{x'_k(N + j) - x_k(N + j)}{x_k(N + j)} \right|, \quad j = 1, 2, \ldots, n.
\]

To implement the algorithm for prediction, mean absolute error (MAE), mean squared error (MSE), and relative mean squared error (RMSE) are used to characterize the prediction error:

\[
\text{MAE} = \frac{1}{n} \sum_{j=1}^{n} \left| \frac{x'_k(N + j) - x_k(N + j)}{x_k(N + j)} \right|, \tag{12}
\]

\[
\text{MSE} = \frac{1}{n} \sum_{j=1}^{n} \left( \frac{x'_k(N + j) - x_k(N + j)}{x_k(N + j)} \right)^2, \tag{13}
\]

Guo X X, et al. Sci China Inf Sci January 2023 Vol. 66 112102:6
Algorithm 1 Prediction based on the DFNS scheme

Input: Observations $x_k(i)$, $1 \leq k \leq m$, $1 \leq i \leq N$;
Output: Predicted values $x_k'(N+j)$, $1 \leq k \leq m$, $1 \leq j \leq n$;

1: Normalize $x_k(i)$ to $s_k(i)$;
2: for $\tau = 1, 2, \ldots, \tau_{\text{max}}$ do
3: Build the matrices $P$, $S_k$, and $\hat{P}$ by $s_k(i)$;
4: Set $P$ as the input matrix and $S_k$ as the output matrix; then train the network;
5: Predict the values $s_k'(j)$, $M+1 \leq j \leq N$, with $\hat{P}$ and the obtained network;
6: Evaluate the fitness function $f$;
7: end for
8: Select the optimal time delay $\tau$ by minimizing the fitness function;
9: for each step, $j = 1, 2, \ldots, n$ do
10: Reconstruct the phase space $S_k$ using the optimal time delay and $s_k(i)$;
11: Set $P$ as the input matrix and $S_k$ as the output matrix; then train the network, where $k = 1, 2, \ldots, m$;
12: Predict the unknown values $s_k'(N+1)$ with $\hat{P}$ and the obtained network;
13: Add the predicted values $s_k'(N+1)$ to $s_k(i)$;
14: end for
15: Obtain the predicted values $s_k'(N+j)$, $k = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$;
16: De-normalize $s_k'(N+j)$ to $x_k'(N+j) = s_k'(N+j) \times (\max(x_k) - \min(x_k))$;
17: Calculate the values of MAE, MSE, and RMSE.

$$\text{RMSE} = \frac{1}{n} \sum_{j=1}^{n} \left( \frac{x_k'(N+j) - x_k(N+j)}{x_k'(N+j)} \right)^2$$ \quad (14)

A schematic diagram for the prediction is shown in Figure 3, and the pseudo code of the program is given in Algorithm 1.

Remark 1. Our proposed scheme employs the radial basis function network together with the dynamical information to train the model. The dynamical information characterized by the reconstructed phase space is set as the output vectors. Since our scheme is based on the feed-forward neural network, the back propagation neural network (BPNN) [40] can also be applied to combine the dynamical characters to build the model.

3.2 Parameter selection by the ICPSO algorithm

If the upper bound $\tau_{\text{max}}$ of the time delay is very large, the traversal algorithm will be much more time consuming. Therefore, an intelligent algorithm is employed to determine a suitable time delay for prediction.

Particle swarm optimization (PSO) was introduced by Kennedy and Eberhart [41] in 1995 as a method to optimize continuous nonlinear functions. The basic idea of PSO originates from the process of bird flock or fish school seeking food randomly. The birds (particles) search for food (the position) according to experience of its own (local best) and communication of swarm (global best), then adjust the searching direction and speed. The position and the velocity of the $i$-th particle are defined as $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ and $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$, respectively, where $d$ is the dimension of the variable. The updating equations
of the position and velocity are

\[
\begin{align*}
x_j^i(t+1) &= x_j^i(t) + v_j^i(t+1), \\
v_j^i(t+1) &= wv_j^i(t) + C_1 p_j^i(t) - x_j^i(t) + C_2 r_2(g_j(t) - x_j^i(t)),
\end{align*}
\]

where \( t \) represents the iteration number, \( x_j^i \) is the \( j \)-th element in the position vector of the \( i \)-th particle, \( v_j^i \) is the \( j \)-th element in the velocity vector of the \( i \)-th particle, \( w \) is the inertia weight ranging from 0 to 1, \( C_1 \) and \( C_2 \) are called cognitive and social learning factors, \( r_1 \) and \( r_2 \) are random numbers taking values in the interval \([0, 1]\), \( p_j^i \) is the \( j \)-th element in the personal best position vector of the \( i \)-th particle, and \( g_j \) is the global best solution. The updated position of the \( i \)-th particle relies on the information of its previous velocities (inertia term), current velocity, individual best solution, and global best solution.

In our case, the value of the time delay is an integer, and the parameter is multi-dimensional. So the updating equation corresponding to the position should be integer constrained, i.e., the updating equations are

\[
\begin{align*}
x_j^i(t+1) &= \lfloor x_j^i(t) + v_j^i(t+1) \rfloor, \\
v_j^i(t+1) &= wv_j^i(t) + C_1 p_j^i(t) - x_j^i(t) + C_2 r_2(g_j(t) - x_j^i(t)),
\end{align*}
\]

where \( \lfloor * \rfloor \) means taking the integer part of *. The algorithm aims to find the parameter \( \tau \) to minimize the fitness function \( f \) defined in (8). The process to obtain the optimal parameters by ICPSO is illustrated as follows.

- **Step 1:** initialize the population. Set the initial population size \( Q \), taken from the range of the time delay, \([\tau_{\text{min}}, \tau_{\text{max}}]\), where \( \tau_{\text{min}} = 1 \) and \( \tau_{\text{max}} \) is indirectly affected by the dataset. In general, a small population size reduces the time of one-step searching but increases the number of steps, and a large population size increases the time of one-step searching but reduces the number of steps. In this paper we choose a moderate value \( Q = 10 \) in each simulations. Then, initialize the positions \( \{\tau^1, \tau^2, \ldots, \tau^Q\} \) and velocities \( \{v^1, v^2, \ldots, v^Q\} \) of the \( Q \) individuals. Here, \( \tau^i = (\tau^1_i, \tau^2_i, \ldots, \tau^d_i) \), \( \tau^i_j \) takes on an integer value randomly between 1 and \( \tau^\text{max}_j \), and \( v^i = (v^1_i, v^2_i, \ldots, v^d_i) \), \( v^i_j \) takes on random values between \(-1\) and 1.
- **Step 2:** initialize the local best and global best. The present position of each particle is set as the initial personal best position, \( p_i = \tau_i^1 \), \( 1 \leq i \leq Q \), and the corresponding fitness value is saved as \( f^i \). Assuming the minimum value of \( \{f^i\} \) to be \( f^k \), we have the global best position \( g = p^k \), and the corresponding fitness value \( F = f^k \).
- **Step 3:** update and iterate.
  1. Update the velocity and the position by (16);
  2. Evaluate the fitness function at the new position;
  3. Update the local best and global best. If the fitness function value at the \( i \)-th new position is smaller than \( f^i \), replace \( f^i \) by this new value, and replace \( p^i \) by its new position. Find the minimum value of the fitness function at all the new positions, then update \( g \) and \( F \);
  4. If the evolution of global best value \( F \) is stationary (the fluctuation threshold is set to be \( 10^{-4}F \)), or the maximum of iteration is reached, go to (5), otherwise, return to (1);
  5. Obtain the optimal parameters.

The process diagram of ICPSO is shown in Figure 4.

### 4 Applications

In this section, we demonstrate the application of our proposed prediction scheme. Lorenz chaotic time series, real-world data including stock market index in finance science, influenza data in epidemiology are predicted. Descriptions about the data sets are shown in Appendix A. The selection of optimal parameters is based on the ICPSO algorithm. Moreover, the performance of our method quantified by MAE, MSE, and RMSR, is compared with that of the LSTM method [42–44]. LSTM was developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into
and out of the cell. In this paper, the prediction scheme by LSTM is implemented by the Matlab Deep Learning Toolbox\(^2\).

4.1 Prediction of the chaotic time series

The classical Lorenz system is introduced to show the process and performance of the prediction based on the DFNN scheme. The Lorenz chaotic time series are obtained by solving Eq. (2), where the initial point is defined as \([x_0, y_0, z_0] = [0.1, 0.1, 0.1]\), the sampling interval is fixed to 0.02, and the time span is set to [0, 200]. To ensure that the initial point and the time series are located in the chaotic attractor, the last point of the computed time series is set as the new initial point, and Eq. (2) is solved again. A total number of 10001 three-dimensional vectors are obtained, the data from locations 2001th to 2500th are used to train the network, and the data located after the 2500th vector are applied to test the prediction performance.

The fitness function in this example is defined as

\[
f = \frac{1}{3} \sum_{k=1}^{3} \frac{\text{mean}(|s'_k(\cdot) - s_k(\cdot)|)}{\text{std}(|s'_k(\cdot) - s_k(\cdot)|)},
\]

(17)

where \(s_k, 1 \leq k \leq 3\), correspond to the normalized \(x, y,\) and \(z\) signals, \(s'_k, 1 \leq k \leq 3\), represent the predicted ones, mean and std are defined as in (9) with \(N\) the length of the time series and \(M\) the number of points in the reconstructed phase space. Note that, this fitness function is different from (8), since some values of the calculated time series are close to zero and cannot be used in the denominator. With this well defined fitness function, the ICPSO algorithm combining DFNN is employed to select the optimal time delay.

A two-layer feed-forward neural network is built to train the data. The input matrix \(P\) is constituted by 3-dimensional vectors, and the output matrix is the reconstructed phase space. Then, 70% of the samples are used to train the network, 15% of the samples are applied to validate the trained network, the remained 15% of samples are for testing, and MSE is used to measure the training performance. The number of hidden neurons is fixed to 20 in this example. The parameters in ICPSO algorithm are fixed as follows: the initial population size \(Q = 10\), the inertia weight \(w = 0.8\), the self learning factor \(C_1 = 0.5\), the group learning factor \(C_2 = 0.5\), and the maximum number of iterations is 50. The LSTM layer has 300 hidden units, and the solver is set to ‘adam’, training for 250 epochs. To prevent the gradients from exploding, set the gradient threshold to 1. Specify the initial learn rate \(0.005\), and drop the learn rate after 125 epochs by multiplying by a factor of 0.2.

---

\(^2\) Time series forecasting using deep learning. [https://www.mathworks.com/help/deeplearning/index.html]
Figures 5(a) and (b) show the MSE and error histogram of training, validation, test corresponding to the obtained network. The optimal parameters depend not only on the time series, but also on the method of training for the network. The optimal time delay in this example is $[\tau_x, \tau_y, \tau_z] = [2, 3, 1]$ by DFNN-BP (training the network by BP neural network), but equals to $[4, 3, 1]$ by DFNN-RBF (training the network by RBF neural network). Figures 5(c) and (d) compare the real data, the training values, and the forecasting values. Figure 5(e) provides a comparison between the real trajectory of attractor and the predicted orbit. Figure 5(f) shows the prediction errors of the corresponding $x$, $y$, and $z$ signals. These results indicate that our proposed DFNN prediction scheme can well forecast the following 120 points of the Lorenz chaotic system. The corresponding MSE, MAE, RMSE calculated by DFNN-BP and LSTM on predicting the Lorenz system are listed in Table 1. The performance of DFNN scheme is much better than that of LSTM method.

4.2 Prediction of the stock market index

Now consider an application of our prediction scheme on the American stock market. The Standard & Poor’s 500 (S&P500) as an index of stock market is employed, and the daily K-line map (candlestick chart) from May 13, 2015 to May 12, 2016, displayed in Figure 6(a), are downloaded from Yahoo Finance$^3$. The high price ($x_1$), opening price ($x_2$), low price ($x_3$), and closing price ($x_4$) are selected as the variables in this system. Since there are 252 or 253 trading days one year in American stock market, 253 data from May 13, 2015 to May 12, 2016 are used to train the network, and 8 trading days from May 13, 2016 to May 24, 2016 are predicted.

The variables $x_i$, $1 \leq i \leq 4$, in the system are first normalized to $s_i = x_i / (\max(x_i) - \min(x_i))$. The original phase space is defined as $P = [s_1, s_2, s_3, s_4]^T$, the embedding dimension of the reconstructed space is set equal to 4 (the dimension of the variables), and the time delay $\tau$ is a parameter. The $k$-th reconstructed phase space is $Y_k = [s_k(j), s_k(j + \tau), s_k(j + 2\tau), s_k(j + 3\tau)]^T$, $1 \leq k \leq 4$.

\[
P = \begin{pmatrix}
    s_1(1) & s_1(2) & \cdots & s_1(M) & \cdots & s_1(N) \\
    s_2(1) & s_2(2) & \cdots & s_2(M) & \cdots & s_2(N) \\
    s_3(1) & s_3(2) & \cdots & s_3(M) & \cdots & s_3(N) \\
    s_4(1) & s_4(2) & \cdots & s_4(M) & \cdots & s_4(N)
\end{pmatrix},
\]

(18)

\[
Y_k = \begin{pmatrix}
    s_k(1) & s_k(2) & \cdots & s_k(M) & \cdots & s_k(N) \\
    s_k(1 + \tau) & s_k(2 + \tau) & \cdots & s_k(M + \tau) & \cdots & s_k(N + \tau) \\
    s_k(1 + 2\tau) & s_k(2 + 2\tau) & \cdots & s_k(M + 2\tau) & \cdots & s_k(N + 2\tau) \\
    s_k(1 + 3\tau) & s_k(2 + 3\tau) & \cdots & s_k(M + 3\tau) & \cdots & s_k(N + 3\tau)
\end{pmatrix},
\]

where $N$ is the length of time series and $M = N - 3\tau$ is the number of points in phase space, which is reconstructed by the time series. The first $M$ columns of $S$ and $Y_k$ are used to train the network. The fitness function of (8) is

\[
f = \frac{1}{4} \sum_{k=1}^{4} \text{mean} \left( \left| \frac{s_k'(i) - s_k(i)}{s_k(i)} \right| \right) / \text{std} \left( \left| \frac{s_k'(i) - s_k(i)}{s_k(i)} \right| \right),
\]

where $s_k'$, $1 \leq k \leq 4$, are the predicted ones, and mean and std are defined as in (9). This fitness function is used to identify the optimal time delay by the ICPSO algorithm. The minimizer of the fitness function corresponds to the optimal time delay. In this example, the parameters in the ICPSO algorithm are set to be the same as those in forecasting the Lorenz system. The optimal time delay $[\tau_1, \tau_2, \tau_3, \tau_4]$ equals to $[1, 6, 9, 6]$ by DFNN-BP, and equals to $[6, 6, 6, 6]$ by DFNN-RBF. The number of hidden units of LSTM layer is set to 300, and the solver is set to ‘adam’. Then we train for 250 epochs. The gradient threshold equals to 1, initial learn rate is 0.005, and we drop the learning rate after 125 epochs by multiplying by a factor of 0.2.

The training performance of the opening price and closing price by DFNN-BP and DFNN-RBF are shown in Figures 6(b) and (c), respectively. Figure 6(d) displays the error histogram by DFNN-BP.
Figure 5 (Color online) Prediction of the Lorenz system. (a) The MSE of training, validation and test corresponding to the obtained network, where the insert figure is the graphical diagram of the obtained neural network; (b) error histogram of training, validation and test corresponding to the obtained network; (c) comparison between the real data and the training values; (d) comparison between the real data and the predicted values; (e) comparison between the real trajectory of attractor and the predicted orbit; (f) prediction errors of the corresponding $x$, $y$, and $z$ signals.

Table 1 Selected time delays and prediction MSE, MAE, RMSE calculated by DFNN-BP and LSTM methods on predicting the Lorenz system

| Time delay | $x$  | $y$  | $z$  | $x$  | $y$  | $z$  |
|------------|------|------|------|------|------|------|
| MSE        | 1.3608E−04 | 4.404E−04 | 1.8830E−04 | 0.2209 | 0.5871 | 0.0979 |
| MAE        | 0.0059 | 0.0150 | 3.8975E−04 | 0.1826 | 0.6857 | 0.0116 |
| RMSE       | 0.0015 | 0.0027 | 4.0603E−07 | 0.7186 | 8.9457 | 0.0004 |
Figure 6 (Color online) Prediction of S&P 500. (a) The daily K-line map and the trading volume of S&P 500 from May 13, 2015 to May 24, 2016; (b) comparison between the real opening price and closing price and the training values by DFNN-BP; (c) comparison between the real opening price and closing price and the training values by DFNN-RBF; (d) error histogram of training, validation and test corresponding to the obtained network by DFNN-BP; (e) comparison between the real opening price and closing price and the predicted values by DFNN-RBF; (f) prediction errors by DFNN-RBF; (g) comparison between the real opening price and closing price and the predicted values by DFNN-BP; (h) prediction errors by DFNN-BP.

The columns $M + 1$ to $N$ of $S$ are used to predict the unknown data of the time series according to the obtained network. The predicted values of $s_i'$ are renewed to $x_i' = s_i' \times (\max(x_i) - \min(x_i))$. The forecasting performance of the opening price and closing price by DFNN-RBF and DFNN-BP are shown.
in Figures 6(e) and (g). In addition, Figures 6(f) and (h) show the corresponding prediction errors. We can see that the trends of the opening price and closing price of the next eight trading days are well predicted. The values MSE, MAE, RMSE calculated by DFNN-BP, DFNN-RBF and LSTM on predicting the opening price and closing price are listed in Table 2. The corresponding error values of DFNN are smaller than that of LSTM method.

### 4.3 Prediction of the influenza data

Influenza poses a significant risk to public health, as evidenced by the 2009 H1N1 pandemic which caused up to 203000 deaths worldwide. Predicting the disease in the incubation period is crucial because the primary goal is to provide guidance on preparing a response and avoid presumably adverse impact caused by a pandemic.

Influenza-like illness (ILI), known as acute respiratory infection and flu-like syndrome/symptoms, is a medical diagnosis of possible influenza or other illness causing a set of common symptoms. US Centers for Disease Control and Prevention (CDC) continuously monitors the level of ILI circulation in the US population by gathering information from physicians’ reports, which record the percentage of patients who seek medical attention exhibiting ILI symptoms. The national, regional, and state level ILI are available through the website 4). From this web site, we obtained the weekly data ranging from the 1st week in 2010 to the 17th week in 2020, on weighted ILI of ten regions (defined by health and human services). Thus, a 10-dimensional time series can be selected as the variables, \( \{x_i : 1 \leq i \leq 10\} \) in this system.

From the obtained time series (Figure 7), it is evident that the outbreaks of ILI are seasonal and the ILI reaches peak prevalence in winter. For convenience of extracting the information, we set the 30th week in a year as an initial point of the segment, and the 29th week in the following year as an end point. The data from the 30th week in 2010 to the 29th week in 2018 are used to train the network, and the following data after the 30th week in 2018 are used to test the performance of prediction. In this application, BP neural network is applied to train the network. The input matrix \( P \) is constituted by 10-dimensional vectors, the number 10 being the dimension of observations, i.e., the number of regions. The output matrix is the reconstructed phase space, \( \{Y_k : 1 \leq i \leq 10\} \).

\[
P = \begin{pmatrix}
  x_1(1) & x_1(2) & \cdots & x_1(M) & \cdots & x_1(N) \\
  x_2(1) & x_2(2) & \cdots & x_2(M) & \cdots & x_2(N) \\
  \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\
  s_{10}(1) & s_{10}(2) & \cdots & s_{10}(M) & \cdots & s_{10}(N)
\end{pmatrix},
\]

(19)

\[
Y_k = \begin{pmatrix}
  x_i(1) & x_i(2) & \cdots & x_i(M) & \cdots & x_i(N) \\
  x_i(1+\tau) & x_i(2+\tau) & \cdots & x_i(M+\tau) & \cdots & x_i(N+\tau) \\
  x_i(1+2\tau) & x_i(2+2\tau) & \cdots & x_i(M+2\tau) & \cdots & x_i(N+2\tau) \\
  \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\
  x_i(1+5\tau) & x_i(2+5\tau) & \cdots & x_i(M+5\tau) & \cdots & x_i(N+5\tau)
\end{pmatrix}.
\]

Note that the embedding dimension of \( Y_k \) equals to 6, which is the number of adjoined regions. In other words, samples in output matrix are 6-dimensional vectors. Of all the samples, 80% are used to train the network, 10% are applied to validate the trained network, and the remaining 10% are for testing the

---

4) http://gis.cdc.gov/fluview/fluportal/dashboard.html.
network. The number of hidden neurons is fixed to 120 in this application. The training performance of the weighted ILI in each region is listed in Figure 7.

As some values of weighted ILI are near zero, the fitness function is defined following the form of (17):

\[ f = \frac{1}{10} \sum_{k=1}^{10} \text{mean}(|x_k^*(\cdot) - x_k(\cdot)|) / \text{std}(|x_k^*(\cdot) - x_k(\cdot)|), \]

where mean and std are defined by (9). The optimal time delay with respect to each region, determined by the ICPSO algorithm in the training process, is listed in Table 3. Here, the parameters in the ICPSO algorithm are as follows: the initial population size \( Q = 10 \), the inertia weight \( w = 0.9 \), the self learning factor \( C_1 = 0.5 \), the group learning factor \( C_2 = 0.8 \), and the maximum number of iteration 50. After the network is obtained, the \( M \)-th to \( N \)-th samples in the matrix \( P \) are employed to predict the unknown data in \( Y_k, 1 \leq k \leq 10 \). The predicted values of the weighted ILI of each region compared with the real data are shown in Figure 8. For the LSTM method, the number of hidden units is set to \( 20 \times 3 \), and the solver is set to ‘adam’; then we train for 250 epochs. The gradient threshold equals to 1, initial learn rate is 0.005, and we drop the learn rate after 125 epochs by multiplying by a factor of 0.2. The prediction errors measured by MSE, MAE, RMSE corresponding to DFNN and LSTM methods are listed in Table 3. The corresponding values of Region 3, Region 5, and Region 8 by DFNN are somewhat larger than that of LSTM method. The reason of this situation is that the complex dynamics of the corresponding region makes the dynamical information insufficient in the scheme of prediction.

From the prediction performance, we find that the trend of weighted ILI in the following year (52 weeks) can be well predicted. The fluctuation on prediction values attributes much to the unstable complex dynamics of the time series. Deep learning can help to discover intricate structure in large data sets by using the back-propagation algorithm. It indicates how a machine should change the internal parameters which are used to compute the representation in each layer from the representation in the previous layer [23, 45]. Therefore, future work based on deep learning methods will hopefully improve the fluctuation problem.

Results from the above experiments show that our proposed prediction scheme is more effective and adaptive. It is proved by the generalized embedding theorem [14, 17] that there exists a correlation function between observations and the reconstructed phase space. Our scheme makes use of the neural network combined with the integer constrained particle swarm optimization algorithm to reveal this correlation function. Although the result from training is a black box, the obtained network can be a good approximation of the correlation function. The prediction can be achieved by the proposed network.
Table 3  The selected time delays and prediction MSE, MAE, RMSE by DFNN-BP and LSTM on predicting the ILI

| Region | Time delay (τ) | DFNN-BP | LSTM |
|--------|---------------|---------|------|
|        | MSE           | MAE     | RMSE | MSE  | MAE  | RMSE |
| Region 1 | 11            | 0.2928  | 0.2267 | 0.0711 | 0.6265 | 0.3243 | 0.1352 |
| Region 2 | 11            | 0.7720  | 0.2789 | 0.1152 | 1.1801 | 0.2667 | 0.0927 |
| Region 3 | 11            | 0.4682  | 0.5726 | 0.6955 | 1.6554 | 0.3292 | 0.1622 |
| Region 4 | 11            | 0.6523  | 0.3421 | 0.1921 | 2.1267 | 0.3674 | 0.2260 |
| Region 5 | 11            | 0.3130  | 0.3055 | 0.1499 | 0.6140 | 0.2835 | 0.1308 |
| Region 6 | 11            | 2.0717  | 0.2996 | 0.1490 | 8.1851 | 0.4357 | 0.3177 |
| Region 7 | 11            | 0.8493  | 0.5031 | 0.4008 | 3.2718 | 0.5244 | 0.4102 |
| Region 8 | 11            | 1.7187  | 0.5881 | 0.4380 | 4.4192 | 0.5544 | 0.3673 |
| Region 9 | 11            | 0.1870  | 0.1568 | 0.0400 | 0.9049 | 0.2570 | 0.1158 |
| Region 10| 11            | 0.4709  | 0.3863 | 0.2315 | 1.8013 | 0.5353 | 0.5116 |

Figure 8  (Color online) Prediction of weighted ILI for each region.

and the corresponding structural feature of the reconstructed phase space matrix. In our proposed prediction scheme in this paper, the utilizing of dynamical information embedded into the network is responsible for its excellent performance. Other prediction schemes, such as the LSTM, can keep track of the historical dynamics of the time series. However, the dynamical information applied in the LSTM scheme is abstract. Our scheme provides the dynamical information by the delay coordinates technique, which indicates a specific route that is topologically equivalent to the dynamical evolution of system. The dynamical information can be better utilized in our scheme, and that explains why the performance of our prediction scheme is better than that of the LSTM method.

The highlight of our proposed scheme is the application of the dynamical information as the output signal of the network. That is, the matrix consisting of multi-dimensional observed time series is set as input and the reconstructed phase space matrix is defined as output in the prediction scheme. Other neural networks, such as the RNN, CNN, can also make use of this idea for time series prediction.
## 5 Conclusion and future work

In summary, in this paper, we design and analyze a prediction system based on the dynamical feed-forward neural network. The dynamic evolution of the time series is exploited during the prediction process. Our research draws on the system mentioned above as integral parts of the theoretical apparatus needed to integrate meteorology, epidemiology, economy, and statistics. The prediction of the signal can give early detection of the abnormal phenomenon, and provide guidance to prepare a timely response and avoid presumably adverse impacts. The classical Lorenz system is applied to test the prediction performance of our method. Moreover, the prediction of Standard & Poors 500 index is given to introduce the application of our prediction scheme. The trend of the opening price and closing price of the following eight trading days is well predicted. In the field of epidemics, the trend of weighted ILI in the following entire year (52 weeks) also can be well predicted.

For future work, we will endeavor to construct the deterministic-oriented system by sparse identification of nonlinear dynamics on exploring the black box in neural networks. Besides, the diffusion characteristics of the correlated time series can be investigated, and we would like to extract an intrinsic differential equation to describe the diffusion process. Furthermore, we plan to enhance the forecast accuracy by reinforcement learning of the system parameters for large-scale data. The application of our proposed system on a specific industrial process is also feasible and significant.

### Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 52071298), Zhong-Yuan Science and Technology Innovation Leadership Program (Grant No. 214200510010), and China Scholarship Council (Grant No. 201907040017).

### References

1. Liu Y, Liu Q L, Wang W, et al. Data-driven based model for flow prediction of steam system in steel industry. Inf Sci, 2012, 193: 104–114
2. Li R J, Xie X M, Xue W P, et al. Hybrid teaching-learning artificial neural network for city-level electrical load prediction. Sci China Inf Sci, 2020, 63: 159204
3. Xiong T, Li C G, Bao Y K, et al. A combination method for interval forecasting of agricultural commodity futures prices. Knowl-Based Syst, 2015, 77: 92–102
4. Lee R S T. Chaotic type-z transient-fuzzy deep neuro-oscillatory network (CTZTFDNN) for worldwide financial prediction. IEEE Trans Fuzzy Syst, 2020, 28: 731–745
5. Jiang F, He J Q, Zeng Z G. Pigeon-inspired optimization and extreme learning machine via wavelet packet analysis for predicting bulk commodity futures prices. Sci China Inf Sci, 2019, 62: 070204
6. Chang V. Towards data analysis for weather cloud computing. Knowl-Based Syst, 2017, 127: 29–45
7. Liu H, Mi X W, Li Y F. Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and Elman neural network. Energy Convers Manage, 2018, 156: 498–514
8. Hoo Y, Tian C S. A novel two-stage forecasting model based on error factor and ensemble method for multi-step wind power forecasting. Appl Energy, 2019, 238: 368–383
9. Ginsberg J, Mohebbi M H, Patel R S, et al. Detecting influenza epidemics using search engine query data. Nature, 2009, 457: 1012–1014
10. Xue H X, Bai Y P, Hu H P, et al. Influenza activity surveillance based on multiple regression model and artificial neural network. IEEE Access, 2018, 6: 563–575
11. Clauset A, Larremore D B, Sinatra R. Data-driven predictions in the science of science. Science, 2017, 355: 477–480
12. Farmer J D, Sidorowich J J. Predicting chaotic time series. Phys Rev Lett, 1987, 59: 845–848
13. Sugihara G, May R M. Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series. Nature, 1990, 344: 734–741
14. Doyle E R, Sugihara G. Generalized theorems for nonlinear state space reconstruction. PLoS ONE, 2011, 6: 18295
15. Ye H, Sugihara G. Information leverage in interconnected ecosystems: overcoming the curse of dimensionality. Science, 2016, 353: 922–925
16. Ma H F, Leng S Y, Aihara K, et al. Randomly distributed embedding making short-term high-dimensional data predictable. Proc Natl Acad Sci USA, 2018, 115: 9994–10002
17. Guo X X, Sun Y T, Ren J L. Low dimensional mid-term chaotic time series prediction by delay parameterized method. Inf Sci, 2020, 516: 1–19
18. Wei X S, Wang P, Liu L Q, et al. Piecewise classifier mappings: learning fine-grained learners for novel categories with few examples. IEEE Trans Image Process, 2019, 28: 6116–6125
19. Subramanian V S, Kumar S. Predicting human behavior: the next frontiers. Science, 2017, 355: 489
20. Jaeger H, Haas H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science, 2004, 304: 78–80
21. Nassif A B, Shahin I, Attili I, et al. Speech recognition using deep neural networks: a systematic review. IEEE Access, 2019, 7: 19143–19165
22. Wang H Y, Yamamoto N. Using a partial differential equation with Google mobility data to predict COVID-19 in Arizona. Math Biosci Eng, 2020, 17: 4891–4904
23. Zhu X L, Fu B F, Yang Y D, et al. Attention-based recurrent neural network for influenza epidemic prediction. BMC Bioinf, 2019, 20: 575
24. Zheng H F, Liu F, Feng X X, et al. A hybrid deep learning model with attention-based conv-LSTM networks for short-term traffic flow prediction. IEEE Trans Intell Transp Syst, 2021, 22: 6910–6920
25. Wang X, Yu F, Dou Z Y, et al. SkipNet: learning dynamic routing in convolutional networks. In: Proceedings of European Conference on Computer Vision, 2018. 409–424
Appendix A Data set description

Lorenz system. The Lorenz system is a system of 3 ordinary differential equations. It is notable for having chaotic solutions for certain parameter values and initial conditions. The Lorenz chaotic time series are obtained by solving Eq. (2), the initial point is first set as \([x_0, y_0, z_0] = [0.1, 0.1, 0.1]\), the sampling interval is fixed to 0.02, and the time span is set to \([0, 200]\). The last point of the obtained time series is reset as the new initial point and we solve the system again to make sure that the computed time series are located in the chaotic attractor. A total number of 10001 three-dimensional vectors are obtained, the data from locations 2001th to 2500th are used to train the network, and the data located after the 2500th vector are applied to test the prediction performance.

S&P500 Index. The S&P500 is a stock market index that measures the stock performance of 500 large companies listed on stock exchanges in the United States. The index can be used to forecast the direction of the economy. A K-line map (also called a candlestick chart), consisting of high price, opening price, low price, and closing price, is a style of financial chart used to describe price movements of a security, derivative, or currency. The opening price is the price at which a security first trades upon the opening of an exchange on a trading day; closing price generally refers to the last price at which a stock trades during a regular trading session. The high price and low price correspond to the highest and lowest price during the trading session. The charts including 4-dimensional time series from May 13, 2015 to May 24, 2016 are downloaded from Yahoo Finance. Since there are 252 or 253 trading days per year in American stock market, 253 data from May 13, 2015 to May 12, 2016 are used to train the network, and 8 trading days from May 13, 2016 to May 24, 2016 are used to test the prediction performance.

ILI data. Influenza-like illness (ILI), known as acute respiratory infection and flu-like symptoms, is a medical diagnosis of possible influenza or other illness causing a set of common symptoms. US Centers for Disease Control and Prevention (CDC) defines ILI as ‘fever (temperature of 100°F or greater) and a cough and/or a sore throat without a known cause other than influenza” (for detail, see the website5). CDC continuously monitors the level of ILI circulation in the US population by gathering information from physicians’ reports, which record the percentage of patients who seek medical attention exhibiting ILI symptoms. The national, regional, and state level ILI are available through the website6. From this website, we obtained the weekly data ranging from the 1st week in 2010 to the 17th week in 2020, on weighted ILI of ten regions (defined by Health and Human Services). These 10-dimensional observations as the variables of system are defined as input of our prediction scheme. The data from the 30th week in 2010 to the 29th week in 2018 are used to train the network, and the following data after the 30th week in 2018 are used to test the performance of prediction.

5) http://www.cdc.gov/flu/weekly/overview.htm.
6) http://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.