Universal framework for reconstructing complex networks and node dynamics from discrete or continuous dynamics data

Yan Zhang [*]
School of Systems Science, Beijing Normal University, Beijing 100875, China

Yu Guo [*]
Software Institute, Nanjing University, Nanjing 210093, China

Zhang Zhang
School of Systems Science, Beijing Normal University, Beijing 100875, China

Mengyuan Chen
China TravelSky Mobile Technology Co., Ltd, Beijing 100029, China

Shuo Wang
School of Systems Science, Beijing Normal University, Beijing 100875, China

Jiang Zhang [†]
School of Systems Science, Beijing Normal University, Beijing 100875, China;
Swarma Research, Beijing 102308, China

(Received 20 November 2021; accepted 15 June 2022; published 16 September 2022)

Many dynamical processes of complex systems can be understood as the dynamics of a group of nodes interacting on a given network structure. However, finding such interaction structure and node dynamics from time series of node behaviors is tough. Conventional methods focus on either network structure inference task or dynamics reconstruction problem, very few of them can work well on both. This paper proposes a universal framework for reconstructing network structure and node dynamics at the same time from observed time-series data of nodes. We use a differentiable Bernoulli sampling process to generate a candidate network structure, and we use neural networks to simulate the node dynamics based on the candidate network. We then adjust all the parameters with a stochastic gradient descent algorithm to maximize the likelihood function defined on the data. The experiments show that our model can recover various network structures and node dynamics at the same time with high accuracy. It can also work well on binary, discrete, and continuous time-series data, and the reconstruction results are robust against noise and missing information.

DOI: 10.1103/PhysRevE.106.034315

I. INTRODUCTION

Living cells, brains, human society, stock markets, global climate systems, and so forth are complex systems composed of many nonlinear interactive units [1–3]. By decomposing a complex system into a static network with dynamics on nodes, networked dynamical system models are powerful tools to describe complex systems, playing a paramount role in understanding their collective behaviors and controlling their functions [1,4]. However, building such models requires professional prior knowledge and modeling experience, which hinders the wide application of these methods. The reconstruction of such networked dynamical systems in a data-driven way remains a fundamental problem, i.e., to retrieve the interaction network structure and the node dynamics from time-series data of complex system behaviors without any subjective biases [5,6].

Historically, recovering network structure, and predicting node dynamical behaviors reconstruction or simulation for predictions for future states are different tasks that are studied separately. First, revealing network structure from time-series data is of great importance because the inferred network can not only help us to understand the behaviors of the system but also can provide information on inner causal structure of interactions within a complex system [3,7–10]. The interdependence relations or causal structure can be obtained by directly calculating some statistical measures [11], perturbing the system [12], optimizing a score function [13,14], or expanding the complex interaction dynamics on a set of basal functions [5,15,16], and other methods [17]. For example, ARNI (the algorithm for revealing network interactions) method can not only infer a network with high accuracy but also be adopted for various nonlinear dynamics. However, one disadvantage is that the performance of the model strongly

[*]These authors contributed equally to this work.
[†]zhangjiang@bnu.edu.cn

depends on the choice of the basal functions. If the prior biases on basal functions are missing, then this approach becomes very time-consuming, limiting its application to larger systems.

Another branch of study is dynamics reconstruction from time series data. According to Taken's theorem, any dynamical behaviors of a nonlinear system can be reconstructed from the time series data, once the observed sequence is long enough [18]. Thereafter, various techniques have been developed to build models to simulate the dynamical behaviors behind data and to predict future states of the system [19–21]. However, the techniques of conventional time series forecasting can hardly extend to complex systems because the task becomes very difficult as the number of variables to be predicted become very large and the long-range correlations between nodes widely exist. The problem can be alleviated when the interaction network structure between nodes are provided as inductive bias, graph (neural) network (GNN) models [22–28] are designed particularly on this task. By learning complex functions of information aggregation and propagation on a given network, GNNs can simulate any complex dynamics on such networks. However, a complete graph is always required for most GNN models, which hinders their wider applications [6,26,29–31].

Very few studies have been proposed to perform both network reconstruction and node dynamics reconstruction tasks together at the same time, although some network inference algorithms are capable of forecasting node states and the implicit and time-variant network structures can also be obtained from deep learning models for forecasting based on an attention mechanism [6,31–34]. The first framework to derive an explicit network is NRI (neural relation inference) [26], in which an encoder–decoder framework is used. However, the complicated encoding process to infer the connections from time series data has limited its scalability and accuracy on larger networks [29,30,35].

This paper proposes a unified framework for automated interactions and dynamics discovery (AIDD). This is a universal framework for learning both the interaction structure and dynamics of a complex system from time series data. The design of a lightweight network generator and a universal dynamics learning component based on Markov dynamics makes AIDD not only be applicable to various networks and dynamics, but also enables it to reconstruct very large networks with high accuracy and robustness. The entire framework is differentiable so that it can be optimized directly by automatic differentiation and machine learning techniques [36]. Beyond tasks of network inference and time series forecasting, we propose a new method to test a learned data-driven model based on control experiments. Finally, we test the validity of our framework on real gene regulatory networks under noisy, incomplete, and interrupted data, which is close to realistic situations. The results demonstrate that high performance can be obtained.

II. METHODS

A. Problem formulation

Suppose the complex system to be considered evolves under discrete time steps. Thus, the dynamics to be reconstructed can be described by

$$\mathbf{X}^{t+1} = f(\mathbf{X}^t, A) + \zeta^t, \tag{1}$$

where $\mathbf{X}^t = [X_1^t, X_2^t, \ldots, X_N^t] \in \mathbb{R}^{N\times D}$ is the state of the system at time t, N is the number of nodes, D is the dimension of the state space of every single node, A is the adjacency matrix of the interaction network to be reconstructed, and $\zeta^t \in \mathbb{R}^{N\times D}$ is the noise imposed on nodes. However, Eq. (1) can only describe the dynamical processes with explicit mathematical forms and cannot be applied to those defined by rule tables or transitional probabilities, such as cellular automata, Boolean dynamics, or Markov chains. Therefore, instead of Eq. (1), we use a more general form, a Markov chain $\{\mathbf{X}^t\}$, to describe the dynamics,

$$f(\mathbf{X}^{t+1} = \mathbf{x}^{t+1}|\mathbf{x}^t, A) \equiv P(\mathbf{X}^{t+1} = \mathbf{x}^{t+1}|\mathbf{x}^t, A), \tag{2}$$

where f is the dynamics to be discovered, $\mathbf{x}^t = ({x_1}^t, {x_2}^t, \ldots, {x_n}^t) \in V^N$ is the observed time series, where V is the state space of each single node and can be either $\mathcal{R}^D$ for continuous values or $\{0,1\}^D$ for discrete values. Here, we use D-dimensional one-hot vector to represent a finite set with D elements. A one-hot vector is a vector with all 0s except one position taking 1, and if the position that 1 appears is p, then the vector represents the pth element. For example, $(0,0,1)$, $(0,1,0)$ can represent a, b, respectively, which are the first and second elements in set $\{a,b,c\}$. P is the conditional probability. Equation (2) is compatible with Eq. (1) but more general [37]. It can even be extended to non-Markov random processes with finite histories by adding more hidden auxiliary variable variables [38].

However, it is difficult to infer the probabilities in Eq. (2), particularly when N is large. Fortunately, the interactions of complex systems are always localized, which means that $P(\mathbf{X}^{t+1} = \mathbf{x}^{t+1}|\mathbf{x}^t, A)$ can be factorized into local transitional probabilities [39]:

$$P(\mathbf{X}^{t+1} = \mathbf{x}^{t+1}|\mathbf{x}^t, A) = \prod_{i=1}^{N} P\big(X_i^{t+1} = x_i^{t+1}|\mathbf{x}^t \odot A_{\cdot i}\big), \tag{3}$$

where $\odot$ is the elementwise product, and $A_{\cdot i}$ represents the ith column of matrix A, and $\mathbf{x}^t \odot A_{\cdot i}$ is a vector representing the state combination of all neighbor nodes of i. Then

$$f_i\big(X_i^{t+1} = x_i^{t+1}|\mathbf{x}^t \odot A_{\cdot i}\big) \equiv P\big(X_i^{t+1} = x_i^{t+1}|\mathbf{x}^t \odot A_{\cdot i}\big) \tag{4}$$

represents the local dynamics of node i, which is also called a causal mechanism in the literature [39]. Where $\odot A_{\cdot i}$ can be regarded as a filter on all nodes' states $\mathbf{x}^t$, so $\mathbf{x}^t \odot A_{\cdot i}$ is the combination of the states of i's neighbors.

Notice, for most network dynamics, nodes always share a same node dynamic, that is, $f_i \equiv f$ for any i. Therefore, the dependence of f on $\mathbf{x}^t \odot A_{\cdot i}$ need be a set function. That is, for any node, f is permutation invariant on all its neighbors such that f can be applied on neighborhoods with different number of nodes [40,41].

Therefore, our task becomes the reconstruction of the network A and learning the local dynamics f_i according to the observed time series $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T)$ with T time steps on multiple samples.

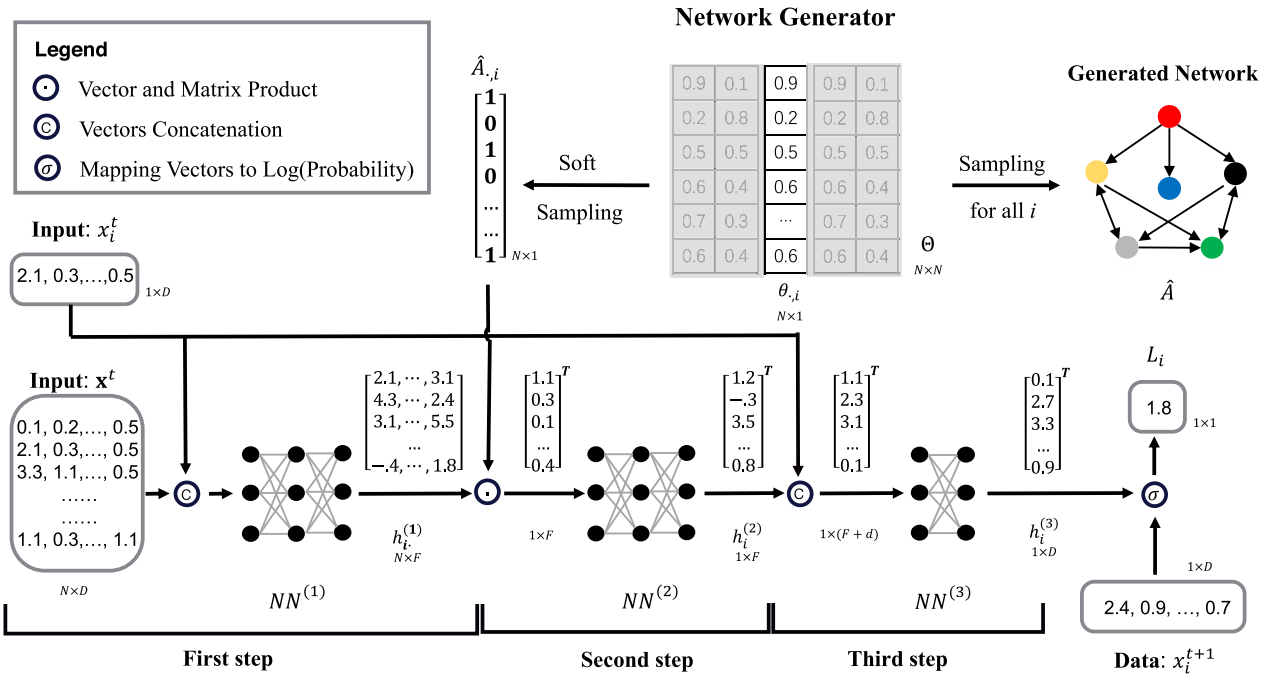

FIG. 1. The architecture of the framework automated interaction and dynamics discovery (AIDD). The upper row outlines how the network generator samples a column of the adjacency matrix for node i, and reconstructs the entire network according to a $N \times N$ dimensional parameter matrix Θ with ith N rows column vector representing the probabilities of $\hat{A}_{\cdot i}$ taking value 1 for node i; and the bottom row shows how the dynamics learner predicts the log-probability of node i's state at the next time step. We also show all the input and output vectors (matrices) and their sizes beside each unit. The first step of the dynamics learner converts the input data of all nodes into their node representations $\mathbf{h}^{(1)}$, the second step multiplies the ith column of the adjacency matrix $\hat{A}_{\cdot i}$ and the representations of all nodes $\mathbf{h}^{(1)}$ to feed into the second neural network $NN^{(2)}$ to generate a new representation $h_i^{(2)}$ with aggregated information of all neighbors of i; and the third step will concatenate $h_i^{(2)}$ and x_i^t to feed into the third neural network $NN^{(3)}$ to generate $h_i^{(3)}$ which represents the predicted state of i at the next step; after that, the log-likelihood will be calculated according to $h_i^{(3)}$ and x_i^{t+1}. Finally, the back-propagation algorithm will be automatically executed by the deep-learning framework(e.g., PyTorch). Each neural network $NN^{(1,2)}$ is a three-layered feed-forward network with $F = 256$ hidden units and ReLU activate function, while $NN^{(3)}$ has two layers.

B. Model

To finish the task, we build a parameterized model with neural networks to represent the reconstructed network structure and node dynamics.

Our model consists of two parts, as shown in Fig. 1. The first part is a network generator that can output a candidate network adjacency matrix $\hat{A}$ according to the parameters $\Theta = \{\theta_{ij}\}$. The second part then attempts to simulate the node dynamics f_i by using a set of neural networks $\hat{f}_{\phi_i}$ for any node i, which is parameterized by $\Phi = (\phi_1, \ldots \phi_N)$ to predict the future state $\hat{X}_i^{t+1}$ according to the candidate matrix $\hat{A}$ and the observed state of the previous time step $\mathbf{x}^t$.

1. Network generation

Instead of using the complicated neural network architecture to generate the candidate network as described in Ref. [26], we directly sample each element in the adjacency matrix $\hat{A}_{ij} \sim \text{Bernoulli}(\theta_{ij})$, where $\theta_{ij} \in [0, 1]$ represents the probability that the entry of the ith row and the jth column in $\hat{A}$ takes the value 1. To make the sampling process differentiable, we use the Gumbel-softmax technique [26,42] to generate the adjacency matrix [30]:

$$\hat{A}_{ij} = \frac{g(\theta_{ij}, \xi_{ij}, \tau)}{g(\theta_{ij}, \xi_{ij}, \tau) + g(1 - \theta_{ij}, \xi'_{ij}, \tau)}, \tag{5}$$

where

$$g(\theta, \xi, \tau) = \exp\left[\frac{\log(\theta) + \xi}{\tau}\right], \tag{6}$$

and $\xi_{ij}, \xi'_{ij} \sim \text{Gumbel}(0, 1)$ are random numbers following the standard Gumbel distribution, and τ is the parameter of temperature to adjust the evenness of distribution of the sampling results which is always 1 in this paper. The random numbers generated by Eq. (5) have a similar distribution as $\text{Bernoulli}(\theta_{ij})$, especially when τ is small. The simulated sampling process is differentiable such that the gradients can be passed by. When $\tau \to 0$, $\hat{A}_{ij}$ exactly equals 1 with probability θ_{ij}, or 0 with probability $1 - \theta_{ij}$.

In this way, our framework has a large improvement in flexibility and computational efficiency compared to the encoder-decoder frameworks such as Ref. [26] and can be applied to very large networks. Further, the introduction of noise ξ_{ij} can push the network generator to jump out of local minimums during optimization.

2. Node dynamics simulation

According to Eq. (3), the dynamics learning part can also be decomposed into local dynamics node by node. Each local node dynamics can be modeled by a feed-forward neural network.

To simulate very complex nonlinear node dynamics in various types for each node, we adopt graph neural network framework. Graph neural network (GNN) is a special kind of neural networks that act on graph-structured data [22]. It constructs a parameterized message passing process on a given graph structure, compares it with the real network dynamic process, and adjust the parameters by the back-propagation algorithm to simulate the real network dynamics process [26].

In our GNN framework, we decompose the message passing process into three steps as shown in Fig. 1. For any node i, the first step is to encode its interaction with any other node $j \in [0, N]$ to form an abstract representation vector:

$$h_{ij}^{(1)} = NN^{(1)}\left(x_i^t, x_j^t\right), \tag{7}$$

with dimension F (in this paper, we set $F = 128$). By iterating this computation for all j, we obtain a matrix:

$$h_{i\cdot}^{(1)} = \left(h_1^{(1)}, h_2^{(1)}, \ldots, h_N^{(1)}\right), \tag{8}$$

with dimension $F \times N$. This hidden variable represents all possible interactions with i.

The second step is to filter out the irrelevant information of nonadjacent nodes of node i according to the candidate adjacency matrix generated in the previous step $\hat{A}_{\cdot,i}$:

$$h_i^{(2)} = NN^{(2)}\left(h_{i\cdot}^{(1)} \cdot \hat{A}_{\cdot,i}\right), \tag{9}$$

where $h_i^{(2)}$ with dimension F is also the abstract representation vector of i after aggregating the information of all i's neighbors. The last step is to generate a new hidden representation vector prepared for output:

$$h_i^{(3)} = NN^{(3)}\left(x_i^t, h_i^{(2)}\right). \tag{10}$$

Here, we also feed x_i^t into $NN^{(3)}$ to promote prediction accuracy. In Eqs. (8), (9), and (10), $NN^{(k)}$ for $k = 1, 2, 3$ are feed-forward neural networks with parameters $\{\phi^{(k)}\}$ in the nearly identical structures (with F hidden units and ReLU activate function. $NN^{(1,2)}$ have three layers, $NN^{(3)}$ has two layers). Notice that, the parameters $\phi^{(k)}$ are shared for all nodes ($\phi_i^{(k)} = \phi^{(k)}$ for any node i and step k) which can reduce the number of parameters in a large sense. As shown in Refs. [40,41], Eqs. (8), (9), and (10) can simulate any network dynamics with the form of Eq. (3).

Finally, we will map the hidden representation of i: $h_i^{(3)}$ into the logarithm of the probability (log-likelihood) that the node state $\hat{X}_i^{t+1}$ taking the value x:

$$\sigma\left(x; h_i^{(3)}\right) \equiv \log \hat{f}_i \equiv \log P\left(\hat{X}_i^{t+1} = x | \mathbf{x}^t \odot \hat{A}_{\cdot,i}\right), \tag{11}$$

where σ is the function to map $h_i^{(3)}$ and x_i to the log-probability, and the functional form depends on the value range V of node state.

If $V = \mathbb{R}^D$ where D is the dimension of values, then σ has the form of logarithm of the standard normal distribution $\log \mathcal{N}(\mu, \mathbf{1})$ with the mean vector $\mu = h_i^{(3)}$ as the output of $NN^{(3)}$, and the variance as the value one (or we can also consider log-laplacian form $\log \mathcal{L}(\mu, \mathbf{1})$, and different distributions will affect the form of objective function as discussed in the next subsection).

If $V = \{0, 1\}^D$, $h_i^{(3)}$ is a one-hot vector representing for discrete values, then σ is a log-softmax function: $\sigma(x; h_i^{(3)}) = \log[\exp(h_i^{(3)}) / \sum_{j=1}^{D} \exp(h_j^{(3)})]$ and the output $h_i^{(3)}$ of $NN^{(3)}$ represents the logarithms of the probabilities that $\hat{X}_i^{t+1}$ taking each possible value.

To generate node states at time step $t + 1$, we can sample values according to the probability $P(\hat{X}_i)$ for node i independently. In this way, we can simulate one step dynamics of all nodes.

In general, the first step is to encode input information of each node and possible neighbors into representations, the second step is to aggregate the information from neighbors, and the last step is to generate output. This design has been verified to be suitable for learning various complex network dynamics [30].

C. Objective function

With the model which can generate candidate network and node dynamics, we can convert the network and node dynamics reconstruction problem as an optimization problem by minimising the following objective function:

$$\begin{aligned} &L(\Theta, \Phi) \\ &= \mathbb{E}_{\hat{A}\sim B(\Theta)}\left[-\sum_{t=1}^{T} \log P(\hat{\mathbf{X}}^{t+1} = \mathbf{x}^{t+1} | \mathbf{x}^t, \hat{A}, \Phi)\right] + \lambda \sum_{ij} \hat{A}_{ij} \\ &\approx -\frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T-1} \sum_{i=1}^{N} L_i + \lambda \sum_{ij} \hat{A}_{ij}, \end{aligned} \tag{12}$$

where

$$L_i = \log \hat{f}_i(x_i^{t+1}) \equiv \log P\left(\hat{X}_i^{t+1} = x_i^{t+1} | \mathbf{x}^t \odot \hat{A}_{\cdot,i}^k; \phi\right) \tag{13}$$

is the local log-likelihood, and K is the number of samples for matrix $\hat{A}$ under a given Θ, $\hat{A}^k$ is the kth sample of adjacency matrix, and x_i^t is the observational vector of state of node i at time t. B represents Bernoulli distribution. We sample K candidate networks under the same parameters Θ to evaluate dynamics simulator to improve accuracy. Thus, the objective function contains two terms, the former being the log-likelihood, which can be decomposed into local log-likelihood. The latter is the structural loss to conform the network to be sparse while avoiding over-fitting. The parameter λ can adjust the relative importance of the structural loss.

It can be easily derived that if the state space $V = \mathbb{R}^D$, and σ takes form of Normal or Laplacian distribution, then the local log-likelihood (13) is

$$L_i = \left\| x_i^{t+1} - h_i^{(3)} \right\|_p, \tag{14}$$

where $p = 2$ if Gaussian distribution is adopted and L_i is mean-squared error (MSE), and $p = 1$ if Laplacian distribution is used and L_i is mean-absolute error (MAE). And $h_i^{(3)}$ is the output of the last layer which represents the mean vector μ of the Gaussian (Laplacian) distribution.

Then, the network dynamics to fit the observational data can be obtained by tuning the parameters of $\theta_{\cdot i}$ in candidate

adjacency matrix and ϕ in dynamics simulators to optimize the objective functions (14) node by node [43]. We use the stochastic gradient descent algorithm to optimize.

D. Training

To train the model, we separate the time series data into data pairs with the following format $(\mathbf{x}^t, \mathbf{x}^{t+1})$ for any time step t. We feed $\mathbf{x}^t$ into the neural network model of any node i (selected one by one) to get the prediction of i's node state at next time $\hat{x}_i^{t+1}$, and this will be compared with the target x_i^{t+1}. The objective function will be evaluate and to train all the parameters. The computation can be parallel in batches, we group data pairs $(\mathbf{x}^t, \mathbf{x}^{t+1})$ into batches. In the experiments, we sample the candidate network at each train epoch, that is equivalent to set K as the total number of training steps. After forward computation, the parameters in the model will be updated according to stochastic gradient decent algorithm. We do not need to derive the explicit expression of the gradients, the platforms like PyTorch or Tensorflow can calculate the gradients automatically by using automatic differentiation technique. See Sec. 1 of the Supplemental Material (SM) [44] for more details about training algorithm.

E. Performance metrics

We separate all the time series data into training, validation, and test for training the model, selecting the hyper-parameters, and testing the performance of the model, respectively. See Sec. 4 of the Supplemental Material [44] for the ratios of these three data sets are various for dynamics.

To evaluate the effectiveness of our model in network inference problem and dynamics prediction task, we use the following indicators on the test set.

AUC (area under curve). It evaluates the accuracy of the reconstructed network. It is defined as the area under the ROC curve and is an index for comprehensive evaluation of accuracy and recall. We also use AUC to evaluate the performance of the dynamics reconstruction for binary dynamics.

MSE (mean-squared error). It measures the performance of the node dynamics reconstruction. It is the expectation of the square of the difference between the true value and the predicted value (estimated value).

MAE (mean-absolute error). It measures the performance of the node dynamics reconstruction. It is the expectation of the absolute of the difference between the true value and the predicted value (estimated value).

III. RESULTS

A. Experimental design

To test our method on various network structures and dynamics, we use different well-known dynamics on both real and modeled networks to generate time series data.

1. Networks

The networks we used are shown in Table I. These graphs are either generated by well-known models (ER [45] for Erdos Renyi, WS [2] for Watts-Strogatz, and BA [46] for Barabasi-Albert) or from empirical data including a gene network (for *Saccharomyces cerevisiae*, Gene) [47], an intercity traffic network (City) of China, three social networks (Email, Dorm, and Blog), and a road network (Road) [48]. See Sec. 3 of the Supplemental Material [44] for more details for empirical networks. In the experiments where the number of nodes is less than 1000, the weight λ [see Eq. (12)] is set to 0.0001, while λ is set to 0.001 for larger networks.

TABLE I. Network parameter of real network. For each network, N, E, $\langle k \rangle$, and $\langle c \rangle$ represent the number of nodes, the number of edges, average degree, and average clustering coefficient, respectively.

Network	N	E	$\langle k \rangle$	$\langle c \rangle$	Directed
Email	1133	10 902	4	0.220	No
Road	1174	2834	2	0.078	No
Blog	1224	19 025	15	0.210	Yes
Dorm	217	2672	24	0.399	Yes
City	371	2752	7	0.485	Yes
Gene	100	195	2	0.070	Yes

2. Test dynamics

We generate time series data by a set of network dynamics including continuous, discrete, and binary dynamics. As shown in Table II, Spring (spring dynamics) [26], SIR (an inter-city meta-population SIR epidemic model) [49,50], and Michaelis–Menten kinetics [51] all are examples of continuous dynamics. The coupled map network (CMN) [52] and the voter model [53] are representatives of discrete and binary dynamics, respectively. The equations for describing these dynamics are reported in Table II, and more details are referred to Sec. 4 of the Supplemental Material [44]. Notice that all the dynamics listed in the table can be converted into a Markovian dynamics.

B. Experimental results

To generate time series data, we ran the simulation for several time steps for each dynamical model and network. All the reported results are on the testing dataset. We compare our model to a series of baseline methods for both network inference and single step forecasting tasks. ARNI [15] and NRI [26] are both the state-of-the-art models for network inference and time series forecasting. The former is based on the block-orthogonal regression method, and the latter is based on deep learning and graph networks. Two other frequently used statistical metrics, partial correlation and mutual information, are also compared on the network inference task. In addition, the classic time series forecasting model, long short-term memory (LSTM) [38], is compared on the prediction task. See Sec. 2 of the Supplemental Material [44] for the details of compared methods. The comparison results on various networks and dynamics are shown in Table III.

The model can also work well on dynamics simulation. As shown in Table III, the MSE errors for continuous and discrete time series are always smaller than 10^{-3} for single step prediction. The AUC accuracy for binary state dynamics can also get high values. The model can also output multistep prediction results by feeding the result of the one-step prediction output

TABLE II. Functions for various dynamical rules, where X_i is the state of i and N_i represents the neighbors of i.

Type	Model	Dynamics	Description
Continuous	Spring	$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i$	where $X_i \equiv \{\mathbf{r}_i, \mathbf{v}_i\}$, $\mathbf{r_i}, \mathbf{v_i} \in \mathbb{R}^2$
		$\frac{d\mathbf{v}_i}{dt} = -k\sum_{j\in N_i}(\mathbf{r}_i - \mathbf{r}_j)$	$k = 0.1$
Continuous	Michaelis–Menten kinetics	$\frac{dX_i}{dt} = -X_i + \frac{1}{\lvert N_i\rvert}\sum_{j\in N_i}\frac{X_j}{1+X_j} + \xi_i$	$X_i \in \mathbb{R}$, $\xi_i \sim \mathcal{N}(0, 0.01)$
Continuous	SIR	$\frac{ds_n}{dt} = -as_ni_n + \omega\sum_{m\neq n}P_{nm}(s_m - s_n)$	where $X_n \equiv \{s_n, i_n, r_n\} \in [0, 1]^3$
		$\frac{di_n}{dt} = as_ni_n - bi_n + \omega\sum_{m\neq n}P_{nm}(i_m - i_n)$	$a = 0.27, b = 0.12, \omega = 0.02$
		$\frac{dr_n}{dt} = bi_n + \omega\sum_{m\neq n}P_{nm}(r_m - r_n)$	$P_{nm} \in [0, 1]$ (see Sec. 4 of SM)
Discrete	Coupled map network	$X_i^{t+1} = (1-\epsilon)f(X_i^t) + \frac{\epsilon}{\lvert N_i\rvert}\sum_{j=1}^{N}f(X_j^t)$	where $X_i \in [0, 1]$
			$f(X_i^t) = \eta X_i^t(1 - X_i^t)$, $\eta = 3.5, \epsilon = 0.2$
Binary	Voter	$P(X_i^{t+1} = 1 \lvert X_{j\in N_i}^t) = \frac{\sum_{j\in N_i}X_j^t}{\lvert N_i\rvert}$	$X_j \in \{0, 1\}$

back to the model as the input. Figure 2 shows the results for the selected dynamics.

C. Performance under different network parameters

In general, our model works very well on the large sparse networks, and the performance on both tasks decreases as the edge density increases, as shown in Fig. 3(b).

We can improve the accuracy by increasing the amount of data. Figure 3(a) shows how the AUC depends on both network size and data volume to be fed into the model systematically. There is a trade-off between network size and data volume under a given accuracy, as shown in Fig. 3(a). It is interesting to observe that data volume is sensitive to network size only when the number of nodes is between 300 and 500, and beyond that, a minimum amount of data volume is sufficient to obtain an acceptable accuracy (e.g., AUC = 0.7), and this almost does not depend on how large the network is. We suppose that the given total number of epochs is sufficient for training the model only for networks smaller than 300, which is the reason why a sudden increase in data volume is observed. When the size is larger than 500, the model can converge quickly when sufficient data volume is provided; therefore, the curves become insensitive again.

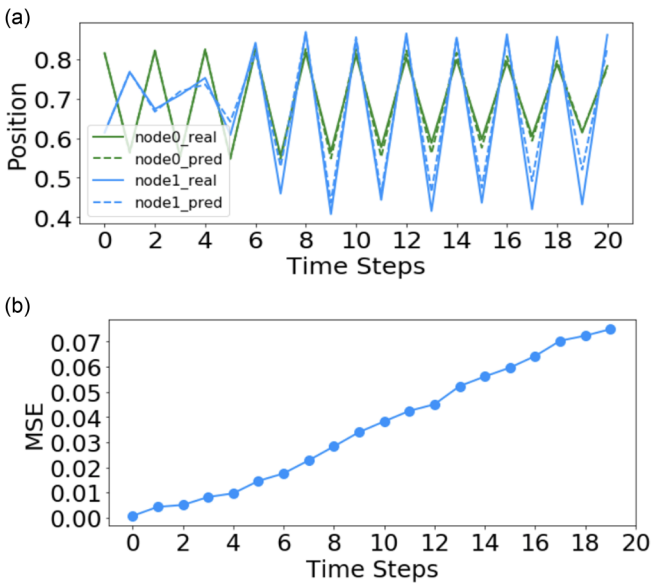

FIG. 2. Multistep prediction results of AIDD on CMN dynamics data in a 10-node ER network. In panel (a), we show the time series data of multistep predictions and the ground truths for two selected nodes. In panel (b), we show how the mean-squared error (MSE) increases with time for CMN dynamics. The parameters are the same as in Table III.

D. More experiments

To further test our method on different conditions, we design more experiments including network and dynamics reconstruction on incomplete networks, under the real scenario, and control experiment with learned network and node dynamics.

1. Robustness against noise and hidden nodes

A good data-driven model must be robust against noise and unobservable nodes such that it can be applied to the real world. To show the robustness against noise of AIDD, we plot changes in AUC with the magnitude of noise on Michaelis–Menten kinetics [51], which can describe the dynamics of Gene regulatory networks, as shown in Fig. 4. Our model can recover the network structure with 0.85 AUC when the mean magnitude of noise is 0.3.

In real applications, we can only obtain partial information of the entire system owing to the limitations of observation. Thus, a certain proportion of nodes are unobservable or hidden. This requires the inference algorithm to be robust to hidden nodes. Thus, we test the AIDD on an incomplete network. To generate the incomplete network data as the ground truth, we randomly select a certain percentage of nodes as the hidden nodes [Fig. 4(a)], and the time series data of these nodes are removed. AUC decreases and MSE increases as the fraction of the number of unobserved nodes increases on both spring and voice dynamics, as shown in Fig. 4(c); however, the sensitivity depends on various types of dynamics. It is found that when the proportion of missing nodes reaches 50%, the inference accuracy is still above 95%, which proves that our

TABLE III. Comparisons of performance on network inference and dynamics prediction tasks between AIDD and other selected methods (columns) on different dynamics and networks (rows). In the network column, we use network-size format. The networks marked with "D" represent directed graphs. The same data volume is shared for different methods in one row. The items marked by a dash (—) indicate that valid results of the model cannot be obtained due to the limitations of the specific method on dynamics, memory, or time consumption. The best results among all the compared algorithms in the same row are boldfaced, and the second-best results are marked by an asterisk (*). All networks generated by models share the same edge density value, which is 1% for large networks (size > 10), and it is 20% and 3% for small networks with sizes smaller than 10, and ER networks with size = 200, respectively, to avoid isolated nodes. All the results are the averages of five repeated experiments.

			ARNI		MI	PC	NRI		LSTM	OURS	
Type	Model	Network	AUC	MSE	AUC	AUC	MSE/AUC_Dyn	AUC	MSE/AUC_Dyn	MSE/AUC_Dyn	AUC
Continuous	Spring	ER-10	0.5853	1.33×10^{-3}	0.7500	0.8250	2.60×10^{-8}	0.9998*	2.98×10^{-4}	2.70×10^{-4}*	**1.0**
		WS-10	0.5125	1.58×10^{-3}	0.6875	0.7875	8.40×10^{-8}	0.9997*	3.35×10^{-4}	3.31×10^{-4}*	**1.0**
		BA-10	0.5169	1.10×10^{-3}	0.6422	0.6571	7.00×10^{-10}	0.9999*	2.14×10^{-4}	2.90×10^{-5}*	**1.0**
		ER-2000	—	—	0.4997	—	—	—	2.25×10^{-3}	1.18×10^{-5}	**0.9886**
		WS-2000	—	—	0.5002	—	—	—	5.89×10^{-3}	8.51×10^{-6}	**0.9933**
		BA-2000	—	—	0.5010	—	—	—	4.54×10^{-3}	2.09×10^{-3}	**0.9523**
	SIR	City-371(D)	0.5424*	8.25×10^{-3}	0.5027	0.5119	—	—	2.28×10^{-3}*	2.98×10^{-5}	**0.9156**
	Menten	Gene-100(D)	**1.0**	9.71×10^{-3}	0.5416	0.6574	—	—	2.29×10^{-3}*	4.37×10^{-5}	0.9960*
Discrete	CMN	ER-10	**1.0**	2.33×10^{-9}	0.5745	0.7804	1.40×10^{-5}	0.8850	2.60×10^{-4}	5.60×10^{-6}*	**1.0**
		WS-10	**1.0**	2.35×10^{-9}	0.6875	0.8375	9.40×10^{-6}	0.9331	2.40×10^{-4}	2.80×10^{-6}*	**1.0**
		BA-10	**1.0**	2.40×10^{-9}	0.4390	0.7439	1.30×10^{-5}	0.6753	9.21×10^{-5}	6.90×10^{-6}*	**1.0**
		ER-200	0.8441*	4.17×10^{-2}	0.5774	0.7648	—	—	5.91×10^{-5}*	2.04×10^{-6}	**0.9987**
		WS-200	**1.0**	2.36×10^{-9}	0.6969	0.7506	—	—	1.63×10^{-4}	1.95×10^{-6}*	0.9987*
		BA-200	0.8840*	2.45×10^{-2}	0.5533	0.7493	—	—	1.46×10^{-4}*	2.57×10^{-6}	**0.9874**
		WS-1000	—	—	0.5670	—	—	—	3.54×10^{-5}	2.92×10^{-6}	**0.9795**
		BA-1000	—	—	0.5290	—	—	—	3.46×10^{-5}	5.48×10^{-5}	**0.9105**
Binary	Voter	ER-10	—	—	0.4390	0.4552	0.9156*	0.5305*	0.5413	**0.9664**	**1.0**
		WS-10	—	—	0.4375	0.5250*	0.8301*	0.5196	0.6966	**0.9452**	**1.0**
		BA-10	—	—	0.4390	0.4607	0.9510*	0.5192*	0.5426	**0.9853**	**1.0**
		WS-1000	—	—	0.5470	—	—	—	0.5301	**0.6567**	**0.9984**
		BA-1000	—	—	0.5030	—	—	—	0.5236	**0.6746**	**0.9573**
		Email-1133	—	—	0.4999	—	—	—	0.5329	**0.7286**	**0.9737**
		Road-1174	—	—	0.5004	—	—	—	0.5448	**0.8829**	**0.9999**
		Dorm-217(D)	—	—	0.5219	—	—	—	0.5688	**0.6918**	**0.9920**
		Blog-1224(D)	—	—	0.4995	—	—	—	0.5300	**0.6366**	**0.8401**

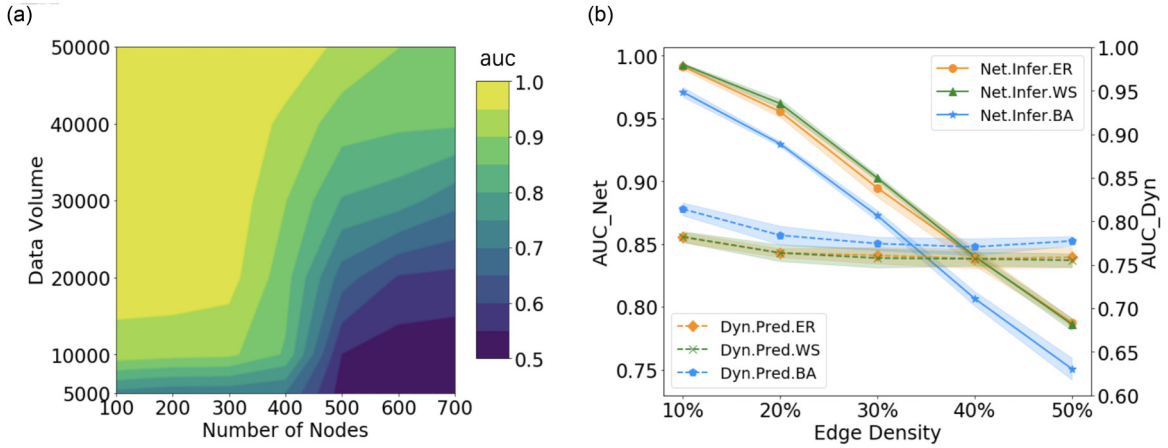

FIG. 3. The performance of AIDD under different factors in network inference and dynamics learning. Panel (a) shows how the number of nodes and the volume of data (the number of samples× the number of time steps, which was fixed to 100) jointly influence the network inference accuracy on WS networks under CMN dynamics. With the exception of 100 nodes with a 4% edge density, all nodes shared the same edge density value, which was 1%. Panel (b) shows how performance decreases with edge density. For the experiments in panel (b), we set the number of nodes to 100, and the sparse matrix parameter λ was set to 0.

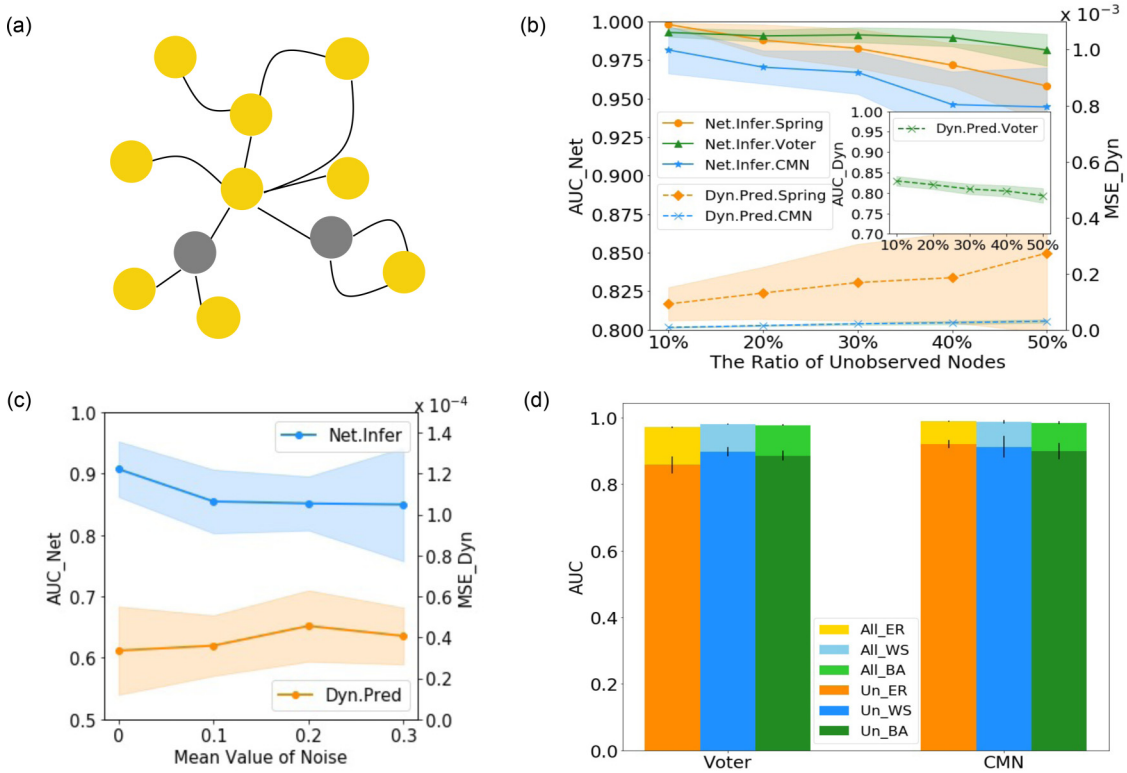

FIG. 4. The robustness evaluation of our model against noise and missing nodes. Panel (a) shows a schematic ground-truth network with missing information on the unobserved nodes (grey nodes). Panel (b) shows the influence of proportion of unobserved nodes on the accuracy of interaction inference on the partial network with observed nodes measured by AUC, and the accuracy of dynamic predictions (inset) measured by the MSE of the observable nodes on Spring, CMN, and the AUC of the observable nodes on Voter dynamics. All the experiments were conducted on an ER network with 100 nodes, and all networks generated by models share the same edge density value, which is 4%. Panel (c) shows the dependence of AUC and MSE on the mean of noise added on each node for the Michaelis–Menten kinetics (Gene dynamics) on the yeast *S. cerevisiae* gene network with 100 nodes. Panel (d) shows the ability to infer interactions on the entire network (the light color bars) and the unobserved partial networks (the dark color bars). All the experiments are conducted on CMN and Voter dynamics with ER, WS, and BA networks, and all networks contain 100 nodes with 10% unobservable nodes selected randomly, and all networks generated by models share the same edge density value, which is 4%.

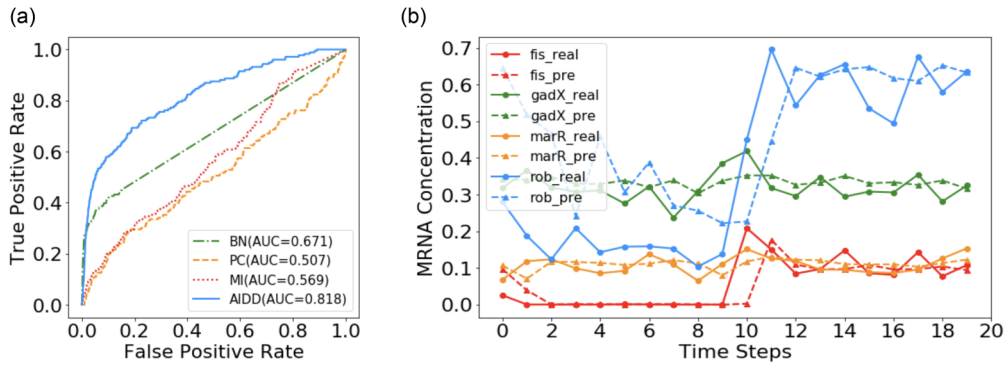

FIG. 5. Performances of AIDD and other compared methods on network and dynamics reconstruction for the gene regulatory network of yeast *S. cerevisiae* with 100 nodes. (a) ROC curves of different network inference methods. True positive rate (y axis) means the fraction of the actual links are classified correctly by our model, and the false positive rate (x axis) represents the fraction of the node pairs without actual link are classified as links by our model. The comparison methods include Bayesian network (BN), partial correlation (PC), mutual information (MI), and AIDD. The AUC for different methods is marked in the legend. (b) shows the comparison between the observed time series of the expression data (real) and the predicted data on selected genes. In this plot, the solid lines represent the predictions and the dotted lines represent the observed data.

model can achieve superior results in the absence of normally sufficient amounts of data.

Furthermore, we test the ability of AIDD to reveal unknown network structures of unobservable nodes on CMN and Voter dynamics, with only the number of hidden nodes available. We completed this task by performing the same interaction inference task, setting the states for unknown nodes to random values. Figure 4(d) shows the AUCs of the link structures of unknown networks on Voter and CMN dynamics. The results reveal that the network inference accuracy is robust for missing nodes. The algorithm can recover the interactions even for unobservable nodes with over 80% accuracy. See Ref. [54] and Sec. 5 of the Supplemental Material [44] for more details of the algorithm.

2. Reconstruction performance in actual scenarios

To verify that our algorithm can be applied to actual scenarios and not only on toy models, we attempt to infer the real subnetwork structure from the known transcriptional network of yeast *S. cerevisiae* according to the time series data of mRNA concentrations generated by GeneNetWeaver (GNW) [47], a famous simulator for gene dynamics.

The networks used by GNWs are extracted from known biological interaction networks (*Escherichia coli*, *S. cerevisiae*, etc.). On these networks, GNW uses a set of dynamical equations to simulate the transcription and translation processes, and it has considered many factors close to real situations. Therefore, GNW is a famous platform for benchmarking and performance assessment of network inference methods.

In the experiment, we used yeast *S. cerevisiae* gene network with 100 nodes as the benchmark gene network, and we used the default parameters of DREAM4_In-Silico in GeneNetWeaver software to generate data. For the dynamics simulator, we use different neural networks for each node because of the heterogeneity of node dynamics and the existence of latent variables, noise, and perturbations [47]. We compare our method with partial correlation, Bayesian network inference, and mutual information algorithms. Our method outperforms others on network inference [Fig. 5(a)] on the AUC(0.82). It can also predict the dynamics with a relatively high accuracy [the mean-absolute error (MAE) is 0.038; see Fig. 5]. This indicates that our method can perform well realistic gene regulatory dynamics.

3. Control experiments

To further verify that AIDD has the ability to learn the ground-truth dynamics and that the well trained adjacency matrix $\hat{A}$ and node dynamics model $\hat{f}_i(\hat{X}_i^{t+1}|\mathbf{x}^t, \hat{A})$ for all i can replace the original system (A and f_i), we design control experiments. The reasons why we choose the control problem as the test bed for our model include (1) the control problem in complex network is very important, and it has relevance to many engineering fields [4]; (2) the control problem is more difficult than predictions. This is true because to control a system means to intervene in it. As a result, we have stood at least on the second level of the causal hierarchy [55].

Here, our control problem is to find a control law based on the well trained system (learned network $\hat{A}$ and node dynamics $\hat{f}_i$) such that some control objective can be achieved. And after the optimized control law is found, we can further apply it on the ground-truth system (real network structure A and real node dynamics f_i). We hope that the same control objective on the ground-truth system can also be achieved. It is obvious that only if the learned network $\hat{A}$ and node dynamics $\hat{f}_i$ are very closed to the ground-truth system, the same control target can be realized. In this way, we can test if the learned system can reproduce the causal mechanisms of the ground-truth system but not merely the pseudocausal laws based on correlations.

For example, suppose the ground-truth system is a spring-mass dynamical system with 10 nodes (masses) and 18 edges(springs) to form the network A as shown in Fig. 6(a), and the node dynamic f_i is the spring-mass dynamics. We can train an AIDD model ($\hat{A}$, $\hat{f}_i$) to reconstruct this network and node dynamics. After a large number of steps of training, we can do the control experiment under the trained system ($\hat{A}$, $\hat{f}_i$). Suppose our control objective is to make all nodes to move in the same direction. However, we can only exert forces on the three driver nodes (0,1,3) as shown in Fig. 6(a).

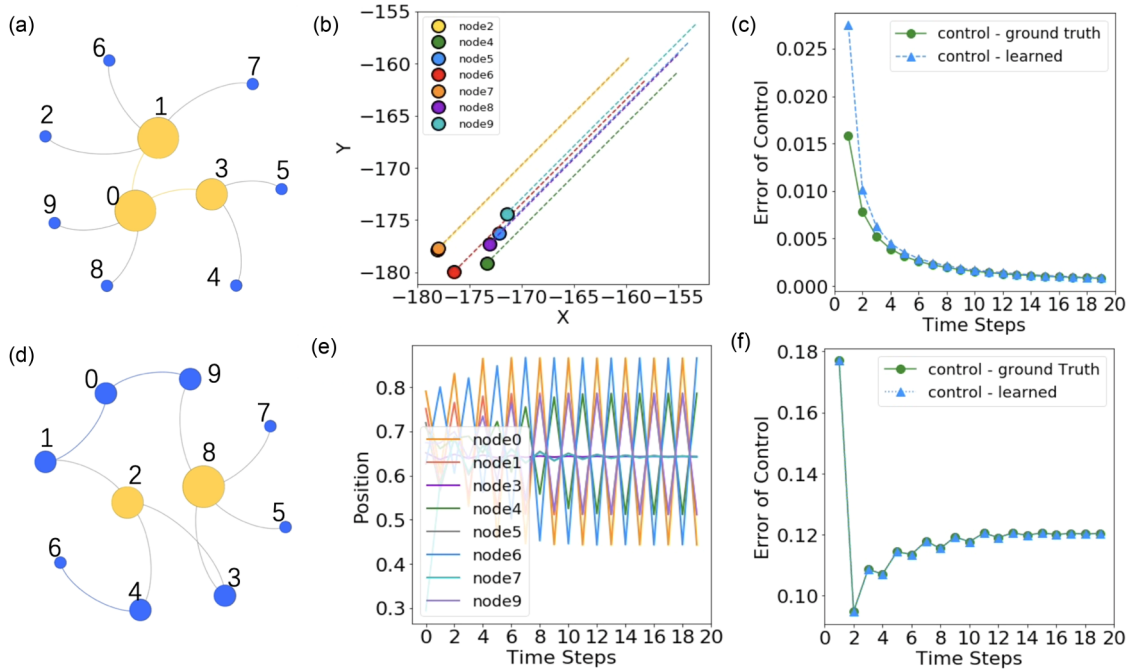

FIG. 6. The control experiments on learned models. Panel (a) shows the spring network that we studied for the first experiment. Three large nodes are driver nodes, and the others are target nodes. The control objective is to request all masses to have the same movement direction. Panel (b) shows the final movement states of all the target nodes under the controls. X and Y represents the two-dimensional coordinates of positions for all masses. Panel (c) shows two loss curves for evaluating goal achievement versus time steps of the controls. One represents the results of learned model, and the other is the ground truth. Panel (d) is a coupled mapping network that we studied in the second experiment. Two large nodes were selected as driver nodes. The control objective is to ask all oscillators to have the same value of 0.6, which is the mean of the value range for all nodes. Panel (e) shows the oscillations of all target nodes during control. Panel (f) shows two loss curves for evaluating goal achievement versus time steps of the controls. One is for the trained model, and the other for the ground truth.

We can train another neural network (the controller) to learn the control law, i.e., how to exert forces on drivers to let all nodes move in the same direction. The inputs to the controller are the states of all nodes on the learned system at each time, the outputs are the forces (including the magnitude and the direction) that will exert on the three drivers at that time, and the training loss function is the error in achieving the control objective. After training enough time steps, when the controller converges, we can transfer the controller optimized on the learned system $(\hat{A}, \hat{f}_i)$ onto the ground-truth system to achieve the same objective. Figures 6(b) and 6(c) show the trajectories of nodes and the errors in achieving the objectives on the ground-truth system and the learned system at each time step after the controller has exerted forces onto the drivers. The two curves collapse together after about six time steps of controls which means the learned system and the ground truth have similar behaviors under the control law. And the fact that the errors tend to zero means the control objectives have been realized.

The second example is a similar control experiment on CMN dynamics. In which the nodes are oscillators, the links are the connections, the network is a small world network generated by the WS model, and the node dynamic is the CMN dynamics. The control objective is to synchronize all nodes by exerting forces on the driver nodes (2 and 8). From Figs. 6(e) and 6(f), the controls are not well achieved for all nodes because the error curves do not converge to zeros. However, the two error curves overlap very well, indicating that the surrogate behaves identically to the ground-truth model.

These experiments show that the AIDD model can learn the mechanism of causality such that we can apply the control law optimized on the learned system onto the real system.

IV. CONCLUSION

In this paper, we propose a unified framework for network structure reconstruction and node dynamics simulation in the same time. We also propose a new standard based on control tasks to evaluate whether the true network dynamics can be learned.

Compared to our previous work in Ref. [30], we formulate our problem in a more rigor mathematical form which is based on Markov dynamics and localized networked interactions. We also propose a new mathematical framework for our model. A unified objective function based on log-likelihood is provided which is the theoretical foundation of why the model can be applied on different dynamics. We also describe how the optimization objective function can be decomposed into single nodes which is the basis for a new deep learning based node by node network reconstruction method. From experimental aspect, the main highlights of the model include scalability, universality, and robustness. The high scalability is reflected by the fact that the model can be applied to large networks with more than thousands of nodes with more than 90% accuracy because the training procedure can be taken

node by node. It is a universal framework because it can be applied to various types of dynamics, including continuous, discrete, and binary. Our model is robust not only on noisy input signals, but also on unobservable nodes. It can recover an entire network even when time series data is missing with more than 90% accuracy. It was also shown to work well on datasets generated by GeneNetWeaver, which emulates the real environment of gene regulatory network dynamics.

Furthermore, we propose a new method based on controls to test the validity of the learned model. Control experiments on spring and CMN dynamics of small networks have proved that well-trained models can replace the real systems.

This framework has many potential applications. For example, our method can be used to infer missing links according to the dynamics information. It can also be used in time series forecasting. In contrast to other forecasting models, a clear binary network can be output by the model, which can provide deeper insights into element interactions and potential causal links, increasing the explanability of the model.

However, some drawbacks are present in the current framework. First, a large amount of training data, especially the time series in diverse initial conditions, is required to obtain a good model. Nevertheless, it is difficult to obtain time series data in many cases. Hence, we may develop new models that are suitable for small data.

Second, all the dynamics considered in this paper are Markovian, but this property is hardly satisfied in real cases. New extensions and experiments on non-Markovian dynamics should be conducted. For example, we can use a recurrent neural network instead of a feed-forward network as the dynamics simulating component.

ACKNOWLEDGMENTS

We acknowledge Prof. Qinghua Chen, Dr. Lifei Wang, and the workshops in Swarma Club for the helpful discussion. We acknowledge the support of the National Natural Science Foundation of China (NSFC) under Grant No. 61673070.

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Complex networks: Structure and dynamics, Phys. Rep. **424**, 175 (2006).

[2] D. J. Watts and S. H. Strogatz, Collective dynamics of "small-world" networks, Nature (London) **393**, 440 (1998).

[3] J. Runge, P. Nowack, M. Kretschmer, S. Flaxman, and D. Sejdinovic, Detecting and quantifying causal associations in large nonlinear time series datasets, Sci. Adv. **5**, eaau4996 (2019).

[4] Y.-Y. Liu and A.-L. Barabási, Control principles of complex systems, Rev. Mod. Phys. **88**, 035006 (2016).

[5] W.-X. Wang, Y.-C. Lai, and C. Grebogi, Data based identification and prediction of nonlinear and complex dynamical systems, Phys. Rep. **644**, 1 (2016).

[6] S. Ha and H. Jeong, Deep learning reveals hidden interactions in complex systems, arXiv:2001.02539

[7] J. Pearl, *Causality* (Cambridge University Press, Cambridge, UK, 2009).

[8] A. Tank, I. Covert, N. Foti, A. Shojaie, and E. Fox, Neural Granger causality for nonlinear time series, Stat **1050**, 16 (2018).

[9] S. Löwe, D. Madras, R. Zemel, and M. Welling, Amortized causal discovery: Learning to infer causal graphs from time-series data, Proc. Mach. Learn. Res. **177**, 509 (2022).

[10] C. Glymour, K. Zhang, and P. Spirtes, Review of causal discovery methods based on graphical models, Front. Genet. **10**, 524 (2019).

[11] J. Peng, P. Wang, N. Zhou, and J. Zhu, Partial correlation estimation by joint sparse regression models, J. Am. Stat. Assoc. **104**, 735 (2009).

[12] M. Nitzan, J. Casadiego, and M. Timme, Revealing physical interaction networks from statistics of collective dynamics, Sci. Adv. **3**, e1600396 (2017).

[13] H. Liu, J. Kim, and E. Shlizerman, Functional connectomics from neural dynamics: Probabilistic graphical models for neuronal network of caenorhabditis elegans, Philos. Trans. Roy. Soc. B: Biol. Sci. **373**, 20170377 (2018).

[14] J. Runge, Causal network reconstruction from time series: From theoretical assumptions to practical estimation, Chaos **28**, 075310 (2018).

[15] J. Casadiego, M. Nitzan, S. Hallerberg, and M. Timme, Model-free inference of direct network interactions from nonlinear collective dynamics, Nat. Commun. **8**, 2192 (2017).

[16] L. Li, D. Xu, H. Peng, J. Kurths, and Y. Yang, Reconstruction of complex network based on the noise via QR decomposition and compressed sensing, Sci. Rep. **7**, 15036 (2017).

[17] C. W. Granger, Investigating causal relations by econometric models and cross-spectral methods, Econometrica **37**, 424 (1969).

[18] F. Takens, Detecting strange attractors in turbulence, in *Dynamical Systems and Turbulence, Warwick 1980* (Springer, Berlin, 1981), pp. 366–381.

[19] P. J. Brockwell, R. A. Davis, and S. E. Fienberg, *Time Series: Theory and Methods* (Springer Science & Business Media, Cham, 1991).

[20] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note, Bonn, Germany: German National Research Center for Information Technology GMD Technical Report **148**, 13 (2001).

[21] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, An overview of reservoir computing: Theory, applications and implementations, in *Proceedings of the 15th European Symposium on Artificial Neural Networks* (2007), pp. 471–482.

[22] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. **20**, 61 (2008).

[23] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, Relational inductive biases, deep learning, and graph networks, arXiv:1806.01261.

[24] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. **32**, 4 (2021).

[25] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, Graph networks as learnable physics engines for inference and control, in *Proceedings of the International Conference on Machine Learning* (PMLR, 2018), pp. 4470–4479.

[26] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, Neural relational inference for interacting systems, in *Proceedings of the International Conference on Machine Learning* (PMLR, 2018), pp. 2688–2697.

[27] Z. Zhang, L. Wang, S. Wang, R. Tao, J. Xiao, M. Mou, J. Cai, and J. Zhang, Neural gene network constructor: A neural based model for reconstructing gene regulatory network, bioRxiv, 842369 (2019).

[28] C. Zheng, X. Fan, C. Wang, and J. Qi, Gman: A graph multi-attention network for traffic prediction, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, Menlo Park, CA, 2020), Vol. 34, pp. 1234–1241.

[29] F. Alet, E. Weng, T. Lozano-Pérez, and L. P. Kaelbling, Neural relational inference with fast modular meta-learning, Adv. Neural Info. Process. Syst. **32**, 11827 (2019).

[30] Z. Zhang, Y. Zhao, J. Liu, S. Wang, R. Tao, R. Xin, and J. Zhang, A general deep learning framework for network reconstruction and dynamics learning, Appl. Netw. Sci. **4**, 110 (2019).

[31] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, Evolvegcn: Evolving graph convolutional networks for dynamic graphs, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, Menlo Park, CA, 2020), Vol. 34, pp. 5363–5370.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, Graph attention networks, in *Proceedings of the International Conference on Learning Representations* (OpenReview.net, 2018).

[33] D. Wang, Z. Zhang, Y. Ma, T. Zhao, T. Jiang, N. V. Chawla, and M. Jiang, Learning attribute-structure co-evolutions in dynamic graphs, arXiv:2007.13004.

[34] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)* (International Joint Conferences on Artificial Intelligence Organization, 2019), pp. 1907–1913.

[35] I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari, Learning dynamical systems from partial observations, arXiv:1902.11136.

[36] A. G. Baydin, P. Barak, A. A. Radul, and J. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res. **18**, 1 (2018).

[37] C. W. Gardiner *et al.*, *Handbook of Stochastic Methods*, Vol. 3 (Springer, Berlin, 1985).

[38] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[39] B. Schölkopf, Causality for machine learning, arXiv:1911.10500.

[40] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, Deep sets, Adv. Neural Info. Process. Syst. **30** (2017).

[41] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, How powerful are graph neural networks? in *Proceedings of the 7th International Conference on Learning Representations* (ICLR, New Orleans, LA, 2019).

[42] E. Jang, S. Gu, and B. Poole, Categorical reparameterization with Gumbel-Softmax, in *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)* (OpenReview.net, 2017).

[43] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, Deep neural networks as gaussian processes, in *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)* (OpenReview.net, 2018).

[44] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.106.034315 for more details about the training algorithm, compared methods, empirical networks, dynamics, and network inference under incomplete information.

[45] P. Erdős and A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci **5**, 17 (1960).

[46] A.-L. Barabási and R. Albert, Emergence of scaling in random networks, Science **286**, 509 (1999).

[47] T. Schaffter, D. Marbach, and D. Floreano, Genenetweaver: *In silico* benchmark generation and performance profiling of network inference methods, Bioinformatics **27**, 2263 (2011).

[48] R. Rossi and N. Ahmed, The network data repository with interactive graph analytics and visualization, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI Press, Palo Alto, CA, 2015), Vol. 29, pp. 4292–4293.

[49] D. Brockmann and D. Helbing, The hidden geometry of complex, network-driven contagion phenomena, Science **342**, 1337 (2013).

[50] J. Zhang, L. Dong, Y. Zhang, X. Chen, G. Yao, and Z. Han, Investigating time, strength, and duration of measures in controlling the spread of Covid-19 using a networked meta-population model, Nonlinear Dyn. **101**, 1789 (2020).

[51] G. Karlebach and R. Shamir, Modelling and analysis of gene regulatory networks, Nat. Rev. Mol. Cell Biol. **9**, 770 (2008).

[52] P. García, A. Parravano, M. G. Cosenza, J. Jiménez, and A. Marcano, Coupled map networks as communication schemes, Phys. Rev. E **65**, 045201(R) (2002).

[53] A. Campbell, G. Gurin, and W. E. Miller, *The Voter Decides* (Row, Peterson, and Co., New York, NY, 1954).

[54] M. Chen, J. Zhang, Z. Zhang, L. Du, Q. Hu, S. Wang, and J. Zhu, Inferring network structure with unobservable nodes from time series data, Chaos: An Interdis, J. Nonlin. Sci. **32**, 013126 (2022).

[55] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect* (Basic Books, New York, NY, 2018).