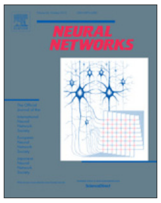
Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



SDARE: A stacked denoising autoencoder method for game dynamics network structure reconstruction

Keke Huang [a,e], Shuo Li [a], Penglin Dai [b], Zhen Wang [c], Zhaofei Yu [d,e,*]

[a] School of Automation, Central South University, Changsha 410083, China
[b] School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China
[c] Center for Optical Imagery Analysis and Learning, Northwestern Polytechnical University, Xi'an 710072, China
[d] Department of Computer Science and Technology, Peking University, Beijing 100871, China
[e] Peng Cheng Laboratory, Shenzhen 518055, China

ARTICLE INFO

Article history:
Received 22 September 2019
Received in revised form 17 January 2020
Accepted 9 March 2020
Available online 14 March 2020

Keywords:
Complex network
Network structure reconstruction
Deep learning
Stacked denoising autoencoder
Compressive sensing

ABSTRACT

Complex network is a general model to represent the interactions within technological, social, information, and biological interaction. Often, the direct detection of the interaction relationship is costly. Thus, network structure reconstruction, the inverse problem in complex networked systems, is of utmost importance for understanding many complex systems with unknown interaction structures. In addition, the data collected from real network system is often contaminated by noise, which makes the network structure inference task much more challenging. In this paper, we develop a new framework for the game dynamics network structure reconstruction based on deep learning method. In contrast to the compressive sensing methods that employ computationally complex convex/greedy algorithms to solve the network reconstruction task, we introduce a deep learning framework that can learn a structured representation from nodes data and efficiently reconstruct the game dynamics network structure with few observation data. Specifically, we propose the denoising autoencoders (DAEs) as the unsupervised feature learner to capture statistical dependencies between different nodes. Compared to the compressive sensing based method, the proposed method is a global network structure inference method, which can not only get the state-of-art performance, but also obtain the structure of network directly. Besides, the proposed method is robust to noise in the observation data. Moreover, the proposed method is also effective for the network which is not exactly sparse. Accordingly, the proposed method can extend to a wide scope of network reconstruction task in practice.

1. Introduction

Complex networks are ubiquitous in social, biological and engineering, such as communication networks, biological networks and WWW networks (Boccaletti, Latora, Moreno, Chavez, & Hwang, 2006; Girvan & Newman, 2002; Li, Bu, Li, Liu, & Shi, 2016; Tanimoto, Brede, & Yamauchi, 2012; Wu, Zhao, Lü, Tang, & Lu, 2016; Yu et al., 2009). In recent years, considerable attention has been paid to the analysis of complex systems. For example, biological scientists made use of gene regulatory networks to understand certain chemical reaction mechanisms (Cussat-Blanc, Harrington, & Pollack, 2015; Tang, Yu, & Kocarev, 2007). Social scientists utilized the social network to study opinion evolution and group behavior. Engineers used the complex network to analyze the dynamics, such as information diffusion, epidemic spreading and evolutionary game dynamics (Wang, Lai, Grebogi, & Ye, 2011; Wang & Wu, 2018; Xu, Su, Zhang, Ren, & Shen, 2015). It should be noted that these analyses rely on precise knowledge of network structure, but we are often incapable of measuring the network structures directly. The data-driven method, as an alternative method that can infer the network structure from some limited observable data, is raising (He, She, & Wu, 2013; Perc & Grigolini, 2013).

Although the data-driven network structure reconstruction is now becoming increasingly important in interdisciplinary fields, it faces several major challenges. Firstly, the structural information is implicit, which cannot be observed from the measurable data. Secondly, as the complex network often consists of many nodes, the dimension of the solution space of all possible structural configurations is extremely high. Thirdly, the data collecting from the complex network is limited, which makes the inverse problem, in general, an ill-posed problem. Fourthly, some notable factors, such as non-linearity, large noise, and lack of data, increase the difficulty of the network structure reconstruction task.

* Corresponding author.
E-mail address: yuzf12@pku.edu.cn (Z. Yu).

https://doi.org/10.1016/j.neunet.2020.03.008
0893-6080/© 2020 Elsevier Ltd. All rights reserved.

So far, some approaches have been proposed to address the inverse problem to some extent (Han, Shen, Wang, & Di, 2015). Roughly, these methods can be divided into three categories: compressive sensing methods (Wang et al., 2011), phase space methods (Napoletani & Sauer, 2008) and delay-coordinate embedding methods (Packard, Crutchfield, Farmer, & Shaw, 1980). Although the above-mentioned methods pave the way of network structure reconstruction to some extent, accurate and robust reconstruction of large complex networks is still a challenging problem, especially given limited measurements contaminated by noise or unexpected outliers.

Deep learning is an emerging field mainly concerned with learning multiple levels of representation of data and coming up with higher levels of abstraction in it (Deng, Yu, et al., 2014; Goodfellow, Bengio, & Courville, 2016; LeCun, Bengio, & Hinton, 2015; Schmidhuber, 2015). The pioneering work of solving compressive sensing with deep learning is proposed by Mousavi, Patel, and Baraniuk (2015). Inspired by these works, here we proposed a stacked denoising autoencoder to incorporate the underlying features of complex network, and then reconstruct the complex network based on state data of each node, i.e. the game strategy (cooperation or defection) of each agent at different times and the overall payoff (fitness) of each agent after each round of dynamic game. Hereafter, we named the proposed method as SDARE.

To be specific, we first design the stacked denoising autoencoder framework and bring the sparse constraint of the network into the loss function. Then, we train the network with the observation data collected from complex network system. Lastly, we feed the learned framework with observation data to obtain the original network structure. In order to verify the efficiency of the proposed method, we use the artificial small-world network and scale-free network, as well as an empirical network as benchmarks to test the reconstruction performances. Here, the empirical network mainly refers to the real complex network in daily life, which is derived from artificial abstractions, such as the Zachary's karate club network, the Dolphin social network and so on. Compared with the most commonly used compressive sensing method, the proposed method shows high accuracy and reliability, especially when there is little observation data. In addition, the proposed method is robust to contaminated noise. Moreover, it is also effective when the network structure is not exactly sparse, which further extends the scope of application.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries of game dynamics in the network as well as the compressive sensing based network structure reconstruction. In Section 3, we present the proposed method for network structure reconstruction. The experimental results are presented in Section 4 and concluding remarks are discussed in Section 5.

2. Preliminaries

In this section, we first introduce the game dynamics in the network as well as the problem formulation of data-driven network structure reconstruction. Then, we will give an outline of compressive sensing. Finally, we explain how to solve the compressive sensing problem with deep learning framework.

2.1. Game dynamics in networks

Here we introduce the game dynamics in networks. Typically, the game dynamics in complex network consist of three basic factors: the network structure model, the game model and the strategy updating model. After defining these factors, the game dynamics can evolve according to three steps. Firstly, all the nodes are arranged in the networked system, and each node occupies a node of the network. Then, all the nodes in the network system interact with their direct neighbors according to the structure of the network, and obtain the fitness based on the game model. Lastly, all the nodes update their strategies according to the strategy updating model.

For the complex network model, two artificial networks: the scale-free network and the small-world network as well as some empirical networks often used to study the game dynamics (Boccaletti et al., 2014; Huang, Zhang, Li, Yang and Wang, 2018). For the game model, there have two common evolutionary game models: the prisoner's dilemma and snowdrift game (Christoph & Michael, 2004; Perc, Gómez-Gardeñes, Szolnoki, Floría, & Moreno, 2013; Perc & Szolnoki, 2008, 2010). In this paper, we will use the prisoner's dilemma as the game model. In each round of the evolutionary game, each node of network can choose one of the two strategies: cooperation or defection, and obtain the fitness based on the payoff matrix. Here, the payoff matrix of prisoner's dilemma is defined as follows:

$$\mathbf{P} = \begin{bmatrix} p_{cc} & p_{cd} \\ p_{dc} & p_{dd} \end{bmatrix}, \tag{1}$$

where p_{cc} and p_{dd} represent the payoff of mutual cooperation and mutual defection respectively, and the mixed choice of strategies gives the cooperator the sucker's payoff p_{cd} and the defector the temptation p_{dc}. In addition, according to the definition of prisoner's dilemma, we have $p_{dc} > p_{cc} > p_{dd} > p_{cd}$. For example, in the tth step, if node 1 and node 2 choose cooperation and defection respectively, i.e. $\mathbf{s}_1(t) = [1, 0]^T$ and $\mathbf{s}_2(t) = [0, 1]^T$, then the payoffs of node 1 and node 2 in the tth step can be calculated as follows:

$$\begin{aligned} u_1 &= \mathbf{s}_1^T(t)\mathbf{P}\mathbf{s}_2(t) = [1, 0] \begin{bmatrix} p_{cc} & p_{cd} \\ p_{dc} & p_{dd} \end{bmatrix} [0, 1]^T = p_{cd} \\ u_2 &= \mathbf{s}_2^T(t)\mathbf{P}\mathbf{s}_1(t) = [0, 1] \begin{bmatrix} p_{cc} & p_{cd} \\ p_{dc} & p_{dd} \end{bmatrix} [1, 0]^T = p_{dc} \end{aligned}. \tag{2}$$

Similarly, we can get the overall fitness of node x in the tth step as follows:

$$u_x(t) = \sum_{y \in \Omega_x} \mathbf{s}_x^T(t)\mathbf{P}\mathbf{s}_y(t), \tag{3}$$

where node y is one of the neighbors of node x and Ω_x is the set of all neighbors of node x including itself, $\mathbf{s}_x(t)$ and $\mathbf{s}_y(t)$ are the strategy of node x and node y at the tth step, and $\mathbf{P}$ is the payoff matrix. During the strategy evolution, each node randomly selects one of its direct neighbors for strategy updating. In other words, there is no preference for the node to select a neighbor for strategy imitation. The possible strategy-update rules include the best-take-over rule, the Fermi rule, the proportional imitation rule, and others (Huang, Liu, Zhang, Yang and Wang, 2018; Liu, Yang, Huang, & Wang, 2019; Szabó & Fath, 2007). Since many artificial networks and empirical networks are heterogeneous, we used the proportional imitation rule in this paper (Santos & Pacheco, 2005). Specifically, node x randomly chooses one of its neighbors y and then adopts its strategy $\mathbf{s}_x(t+1)$ with probability:

$$p(\mathbf{s}_x(t+1) = \mathbf{s}_y(t)) = \frac{u_y(t) - u_x(t)}{Dk_>}, \tag{4}$$

where $\mathbf{s}_x(t)$ is the strategy of x in the tth step, $u_x(t)$ is the fitness of x in the tth step. $D = p_{dc} - p_{cd}$ represents the largest difference in the payoff matrix, and $k_> = \max\{k_x, k_y\}$ with k_x represents the degree of node x. In general, given a typical complex network structure, the game model as well as the strategy updating rule model, we can record the time-series evolutionary game data, which include the strategies and the fitness of all the nodes.

Unfortunately, due to the high cost of measurement and the limitation of measurement technology, the structure of complex network is often unknown. For example, DNA microarrays and mass spectroscopy have made the analysis of gene networks more feasible. However, it is not obvious how the data acquired through such methods can be assembled into unambiguous and predictive models of these networks (Bansal, Gatta, & Di Bernardo, 2006). Thus, how to infer the complex network structure based on the record time-series evolutionary game data is meaningful. As Ω_x is often unknown in the network structure inference problem, we rewrite Eq. (3) as:

$$u_x(t) = \sum_{y\in\Omega_x} a_{xy}\mathbf{s}_x^T(t)\mathbf{P}\mathbf{s}_y(t). \tag{5}$$

Here y represents all nodes in complex network. Typically, $a_{xy} = 1$ means there exist an edge between node x and node y, otherwise, $a_{xy}= 0$. By recording the evolutionary game data from the t_1 step to the t_M step, the network structure inference problem can be converted to the problem of solving linear equations that is defined as follows:

$$\begin{bmatrix} u_x(t_1) \\ \vdots \\ u_x(t_M) \end{bmatrix} = \begin{bmatrix} \mathbf{s}_x^T(t_1)\mathbf{P}\mathbf{s}_1(t_1) & \dots & \mathbf{s}_x^T(t_1)\mathbf{P}\mathbf{s}_N(t_1) \\ \vdots & \vdots & \vdots \\ \mathbf{s}_x^T(t_M)\mathbf{P}\mathbf{s}_1(t_M) & \cdots & \mathbf{s}_x^T(t_M)\mathbf{P}\mathbf{s}_N(t_M) \end{bmatrix} \times \begin{bmatrix} a_{x1} \\ \vdots \\ a_{xN} \end{bmatrix} \tag{6}$$

For the sake of simplicity, Eq. (6) can be briefly described as follows:

$$\mathbf{u}_x^{all} = \boldsymbol{\Phi}_x \mathbf{a}_x^{all}, \tag{7}$$

where $\mathbf{M}$ and $\mathbf{N}$ denote the length of the game dynamics time series and the number of nodes in the whole complex network respectively. $\mathbf{u}_x^{all} \in \mathbf{R}^{\mathbf{M}}$ and $\boldsymbol{\Phi} \in \mathbf{R}^{\mathbf{M}\times\mathbf{N}}$ are the game dynamic data and augmentation of observation matrix of node x, and they can be expressed as follows:

$$\mathbf{u}_x^{all} = \begin{bmatrix} u_x(t_1) & \dots & u_x(t_M) \end{bmatrix}^T, \tag{8}$$

$$\boldsymbol{\Phi}_x = \begin{bmatrix} \mathbf{s}_x^T(t_1)\mathbf{P}\mathbf{s}_1(t_1) & \dots & \mathbf{s}_x^T(t_1)\mathbf{P}\mathbf{s}_N(t_1) \\ \vdots & \vdots & \vdots \\ \mathbf{s}_x^T(t_M)\mathbf{P}\mathbf{s}_1(t_M) & \cdots & \mathbf{s}_x^T(t_M)\mathbf{P}\mathbf{s}_N(t_M) \end{bmatrix}, \tag{9}$$

$$\mathbf{a}_x^{all} = \begin{bmatrix} a_{x1} & \dots & a_{xN} \end{bmatrix}^T. \tag{10}$$

Apparently, the vector $\mathbf{u}_x^{all}$ can be obtained directly from the payoff data. In addition, the strategies time-series $\mathbf{s}_x(t_j)$, $\forall x, j$ and the payoff matrix $\mathbf{P}$ are known, the matrix $\boldsymbol{\Phi}_x$ can be calculated directly. Thus, the only unknown in Eq. (7) is $\mathbf{a}_x^{all}$, which represents the adjacent vector of node x. Since complex network often sparse, namely, the adjacent vector $\mathbf{a}_x^{all}$ is a sparse vector, so the compressive sensing based method can be used to address this problem (Wang et al., 2011). In a similar way, the structure of the whole network $\mathbf{A}_{adj} = (\mathbf{a}_1^{all}, \mathbf{a}_2^{all}, \dots, \mathbf{a}_N^{all})$ can be obtained by solving $\mathbf{u}_x^{all} = \boldsymbol{\Phi}_x\mathbf{a}_x^{all}(x = 1, \dots, N)$ respectively.

2.2. Compressive sensing

In this section, we will give an outline of compressive sensing. Compressive sensing is mainly concerned with the inverse problem of recovering a K-sparse signal $\mathbf{x} \in \mathbf{R}^{\mathbf{N}}$ from a set of under-sample linear measurements:

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{x} + \mathbf{e}, \tag{11}$$

where $\mathbf{y} \in \mathbf{R}^{\mathbf{M}}$ is the measurement vector. $\boldsymbol{\Phi} \in \mathbf{R}^{\mathbf{M}\times\mathbf{N}}$ is a sensing matrix with $M \le N$. $\mathbf{x} \in \mathbf{R}^{\mathbf{N}}$ is a K-sparse vector. Here, K-sparse means that the non-zero elements in the vector is less than K. $\mathbf{e}$ is the measurement error term.

Accordingly, the main problem of compressive sensing is to recover the sparse vector $\mathbf{x}$ given the observation data $(\mathbf{y}, \boldsymbol{\Phi})$. If the measurement error term $\mathbf{e}$ is equal to 0, the mathematical formulation of compressive sensing is defined as follows (Mei et al., 2018):

$$\begin{aligned} &\min_{\mathbf{x}} \|\mathbf{x}\|_0 \\ &s.t.\ \mathbf{y} = \boldsymbol{\Phi}\mathbf{x}, \end{aligned} \tag{12}$$

where $\|\mathbf{x}\|_0$ denotes the L_0 norm of vector $\mathbf{x}$, which represent the sparsity of vector x. If the measurement error term $\mathbf{e}$ is not equal to 0, the Lasso method can be used, and the objective function of Lasso is defined as follows (Wang & Rahnavard, 2013):

$$\min_{\mathbf{x}} \left(\|\mathbf{x}\|_0 + \gamma\ \|\mathbf{y} - \boldsymbol{\Phi}\mathbf{x}\|_2^2\right), \tag{13}$$

where γ is a non-negative regularization parameter to balance the weight of sparsity and least square term. The introduction of least square term can make the solution more robust against measurement noise.

Since the L_0 norm was not convex, the optimization of Eqs. (12)–(13) is NP-hard. Typically, two methods can be used to get approximate solutions. The first one is the greedy method such as the OMP (Orthogonal Matching Pursuit) algorithm (Blumensath & Davies, 2009). The second method is the convex relaxation method (Candes & Tao, 2006), of which the L_1 norm is the commonly used. To be specific, one can use $\|\mathbf{x}\|_1$ to replace $\|\mathbf{x}\|_0$ in Eqs. (12)–(13). Although the convex/greedy algorithms can solve these problems to some extent, they may confront the difficulties of high computational complexity and inaccurate when applied to the network structure inference problem as the measurement data is rare or the network is not exactly sparse.

2.3. Deep learning

Deep learning is an emerging field mainly based on constructing a multiple layer hidden neural network model, and using a large amount of data to train the network model to extract the most representative feature. Until now, the deep learning method has been applied widely in many researches, such as image processing (He, Zhang, Ren, & Sun, 2016; Huang, Liu, Van Der Maaten, & Weinberger, 2017; Krizhevsky, Sutskever, & Hinton, 2012), computer vision (Chollet, 2017), speech recognition (Amodei et al., 2016; Graves, Mohamed, & Hinton, 2013), and natural language processing (Devlin, Chang, Lee, & Toutanova, 2018; Mikolov, Chen, Corrado, & Dean, 2013) and so on. More recently, deep learning has been used to recover sparse signals, the main idea of which is to learn a representation from the training data use the stacked denoising autoencoders (SDA) (Mousavi et al., 2015). In addition, the recurrent neural networks was also used to deal with the problem of block-sparsity recovery (Lyu, Liu, & Yu, 2019). Compared to the greedy iterative method, these deep learning based methods have better performance. However, they have to confront the following problems when they are used to reconstruct network structure:

1. Is there a way to reconstruct the adjacent matrix of the network globally, instead of reconstructing the adjacent matrix column-by-column locally?
2. How to learn the nonlinear mapping from the game dynamics data of the network to the network adjacent matrix $\mathbf{A}_{adj}$ from the training data?
3. Training a neural network often needs a lot of data, how to produce a large amount of training data with good quality to ensure the network reconstruction performance?

In the following section, we will propose the SDARE method, which can learn the mapping from the game dynamics data of all nodes to the network adjacent matrix $\mathbf{A}_{adj}$ directly and solve the three problems discussed above.

3. SDARE: SDA for network structure reconstruction

In this section, the SDA method is proposed to reconstruct the network structure. We will address the above three problems sequentially.

First, we discuss the framework for global network structure reconstruction problem. With the deep learning based signal recovery method, the trained network can map the observation data $\mathbf{y}$ to the signal $\mathbf{x}$. As Eq. (7) indicates, one deep network is needed to learn the relationship between the game dynamics data $\mathbf{u}_x^{all}$ and the adjacent vector $\mathbf{a}_x^{all}$ of node x. As the complex network contains N nodes, we need to train N deep networks to reconstruct the complex network. In order to globally reconstruct the network structure, we should convert the adjacent vector reconstruction problem to the adjacent matrix reconstruction problem. Inspired by the work (Huang, Wang, and Jusup, 2018), we flatten the adjacent matrix $\mathbf{A}_{adj}$ into a vector $\mathbf{vec}\left(\mathbf{A}_{adj}\right)$, and the relationship between the game dynamic data and the vector can be rewritten as:

$$\mathbf{u^{all}} = \Psi\,\mathbf{vec}\left(\mathbf{A}_{adj}\right), \tag{14}$$

where the game dynamic data $\mathbf{u^{all}} \in \mathbf{R^{MN}}$ and augmentation of observation matrices $\Psi \in \mathbf{R}^{\mathbf{MN}\times\mathbf{N^2}}$ are expressed as follows:

$$\mathbf{u}^{all} = [(\mathbf{u}_1^{all})^T \quad \dots \quad (\mathbf{u}_N^{all})^T]^T, \tag{15}$$

$$\Psi = \begin{bmatrix} \Phi_1 & 0 & \dots & 0 \\ 0 & \Phi_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \Phi_N \end{bmatrix}. \tag{16}$$

Based on these definition, we can infer the adjacent matrix globally with the data $\left(\mathbf{u}^{all}, \Psi\right)$.

Now we discuss how to learn a nonlinear mapping from the game dynamics data $\mathbf{u}^{all}$ of the network to the network adjacent matrix $\mathbf{A}_{adj}$ from the training data. We propose a three-layers stacked denoising autoencoder to address this problem. The schematic diagram of SDA method is shown in Fig. 1. Specifically, the SDA consists of three denoising autoencoder. It takes the benefits vector of each node during all rounds of evolutionary game as the input, thus the number of neurons in the input layer is $N_{um} = L_D N$, where L_D is the length of the recorded time series and N is the number of nodes in the network. The second layer is a scale layer, which can scale the data of the input layer and speed up gradient descent. The scale layer is followed by two hidden layers and an output layer, each of which is a hidden layer of a denoising autoencoder (see the left part of Fig. 1). These hidden layers can be formulated as:

$$\mathbf{y}_i = \sigma(\mathbf{W}_i\mathbf{x}_i^{in} + \mathbf{b}_i), \tag{17}$$

where $\mathbf{x}_i^{in}$ is the input layer of the ith denoising autoencoder, $\mathbf{W}_i$ and $\mathbf{b}_i$ is the weight matrix and bias vector of the hidden layer, and $\sigma(\cdot)$ is the activation function, which is a sigmoid function. To ensure consistency between the input layer and output layer of a denoising autoencoder (DA), the output layer can be represented as:

$$\widetilde{\mathbf{x}}_i = \sigma(\mathbf{W}_i^T\mathbf{y}_i + \mathbf{b}_i^T), \tag{18}$$

with $\widetilde{\mathbf{x}}_i$ denoting the output layer of DA, $\mathbf{y}_i$ is the output of the hidden layer. For each denoising autoencoder, we need to pre-train it in unsupervised manner with the loss function of mean squared error (MSE):

$$L(\Theta) = \arg\min_{\Theta} \left\|\widetilde{\mathbf{x}}_i - \mathbf{x}_i^{in}\right\|, \tag{19}$$

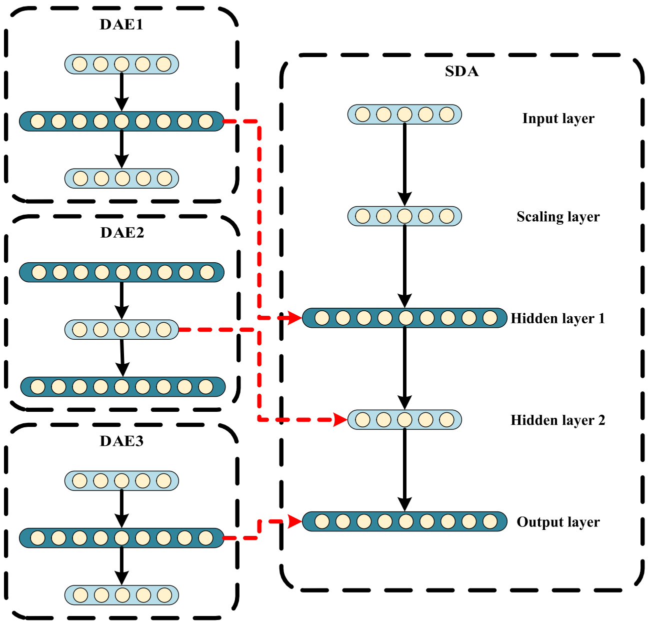


Fig. 1. The schematic diagram of SDA for reconstructing the network based on the game dynamic data, which consists of three DAE. The parameters of the coding layer of the three trained DAEs are assigned to hidden layers and output layer of SDA.

where $\Theta = \left\{\mathbf{W}_i, \mathbf{b}_i, \mathbf{W}_i^T, \mathbf{b}_i^T\right\}$ is the set of parameters.

After the unsupervised pretraining, we are able to stack the three DAs and fine-tune them to train the final SDA. First, we use the hidden layer of the first denoising autoencoder as the input to the second autoencoder. Then we take the hidden layer of the second autoencoder as the input to the third autoencoder. Finally, we take the hidden layer of the third autoencoder as the output layer of the entire stacked denoising autoencoder. One should note that each layer of the SDA used for sparse recovery either has input size of N^2 (the ambient dimension of the original network) and output size of N_{um} (the dimension of the measurement vector) or vice versa (Mousavi et al., 2015). Therefore, these three hidden layers can be formulated as:

$$\begin{aligned} \mathbf{y}_1 &= \sigma(\mathbf{W}_1\mathbf{x}_{in} + \mathbf{b}_1), \\ \mathbf{y}_2 &= \sigma(\mathbf{W}_2\mathbf{y}_1 + \mathbf{b}_2), \\ \widetilde{\mathbf{x}}_{out} &= \sigma(\mathbf{W}_3\mathbf{y}_2 + \mathbf{b}_3), \end{aligned} \tag{20}$$

where $\mathbf{x}_{in} \in \mathbf{R^{Num}}$ is the output of the scale layer, $\mathbf{W}_1 \in \mathbf{R}^{\mathbf{N^2}\times\mathbf{Num}}$, $\mathbf{W}_2 \in \mathbf{R}^{\mathbf{Num}\times\mathbf{N^2}}$, $\mathbf{W}_3 \in \mathbf{R}^{\mathbf{N^2}\times\mathbf{Num}}$ and $\mathbf{b}_1 \in \mathbf{R}^{\mathbf{N^2}\times\mathbf{1}}$, $\mathbf{b}_2 \in \mathbf{R}^{\mathbf{Num}\times\mathbf{1}}$, $\mathbf{b}_3 \in \mathbf{R}^{\mathbf{N^2}\times\mathbf{1}}$ are the paraments of the stacked denoising autoencoder. As with the autoencoder, we need to ensure that the output of the neural network is as much as possible the desired goal, so the loss function of the SDA was defined as follows:

$$L(\Theta') = \arg\min_{\Theta'} \left\{\left\|\widetilde{\mathbf{x}}_{out} - \mathbf{x}_{pre}\right\|_2^2 + \lambda\|\widetilde{\mathbf{x}}_{out}\|_1\right\}, \tag{21}$$

where $\Theta' = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ is the set of parameters, $\widetilde{\mathbf{x}}_{out}$ is the output of the SDA, $\mathbf{x}_{pre}$ is the expected target result, and λ is the penalty term parameter. On the basis of pre-training, we fine-tune the whole network by using Adamoptimizer (Kingma & Ba, 2014) to minimize the loss function in Eq. (21).

The last question is how to generate training data and test data to ensure the performance of network reconstruction. According to Section 2, given a fixed measurement matrix Ψ and an adjacent matrix $\mathbf{A}_{adj}$, we can obtain the observation data $\mathbf{u}^{all}$ according to Eq. (14). Consequently, we can use the k pairs data consisting of the original networks and their benefits matrices as the training

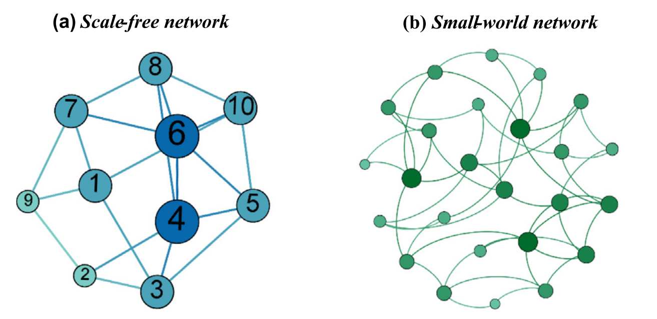


Fig. 2. A schematic diagram of two artificial network, (a) the scale-free network, (b) the small-world network.

data $\mathbf{D}_{train} = \{ \left((\mathbf{u}^{all})_1, (\mathbf{vec}(\mathbf{A}_{adj}))_1\right), \left((\mathbf{u}^{all})_2, (\mathbf{vec}(\mathbf{A}_{adj}))_2\right), \ldots, \left((\mathbf{u}^{all})_k, (\mathbf{vec}(\mathbf{A}_{adj}))_k\right) \}$. Based on the training data, we can learn a nonlinear mapping from the game dynamic data $\mathbf{u}^{all}$ to the vector $\mathbf{vec}(\mathbf{A}_{adj})$. Then we use the test data $(\mathbf{u}^{all})_{test}$ of the network that needs to be reconstructed as the input. To sum up, the generation of training data and test data was stated in detail in Algorithm 1.

Algorithm 1. Generation of training data and test data

```
Input: The original network $\mathbf{A}_{adj}$; the number of training sample $k$; the payoff matrix $\mathbf{P}$;
the length of measure time $M$; the network size $N$
Initialization: The initial measure epoch $m = 1$; the initial iteration $l = 1$
While $m \le M$ do
    Step 1: Initialize the different strategy for each node in network $\mathbf{A}_{adj}$
    Step 2: Using Eqs. (2) and (3) to calculate the current game benefits $\mathbf{u}_x(m)$ for each node
    Step 3: Update the strategy of each node in the next round game dynamic according to Eq. (4)
end while
Step 4: Calculate the observation matrix $\Psi$ and the benefit vector $\left(\mathbf{u}^{\mathbf{all}}\right)_{test}$ of the network $\mathbf{A}_{adj}$ according to Eqs. (5)–(10)
While $l \le k$ do
    Step 5: Generate network $\left(\mathbf{A}_{adj}\right)_l$ with the same degree of distribution as the original
    network $\mathbf{A}_{adj}$
    Step 6: Convert the network matrix $\left(\mathbf{A}_{adj}\right)_l$ to the vector $\left(\mathbf{vec}\left(\mathbf{A}_{adj}\right)\right)_l$ through straigh-
    tening
    Step 7: Calculate the benefits vectors $\left(\mathbf{u}^{\mathbf{all}}\right)_l$ of network $\left(\mathbf{vec}\left(\mathbf{A}_{adj}\right)\right)_l$ through Eq. (14)
end while
Output:The test set $\mathbf{D}_{test} = \left\{\left(\mathbf{u}^{all}\right)_{test}, \left(\mathbf{vec}\left(\mathbf{A}_{adj}\right)\right)_{test}\right\}$; the training set
$\mathbf{D}_{train} = \{\left((\mathbf{u}^{all})_1, (\mathbf{vec}(\mathbf{A}_{adj}))_1\right), \left((\mathbf{u}^{all})_2, (\mathbf{vec}(\mathbf{A}_{adj}))_2\right),$
$\ldots, \left((\mathbf{u}^{all})_k, (\mathbf{vec}(\mathbf{A}_{adj}))_k\right)\}$
```

According to these steps, a collection of game dynamics data $\tilde{U} = \left\{(\mathbf{u}^{all})_1, (\mathbf{u}^{all})_2, \ldots, (\mathbf{u}^{all})_k\right\}$ and the corresponding networks $\tilde{A} = \{\left(\mathbf{vec}\left(\mathbf{A}_{adj}\right)\right)_1, \left(\mathbf{vec}\left(\mathbf{A}_{adj}\right)\right)_2, \ldots, \left(\mathbf{vec}\left(\mathbf{A}_{adj}\right)\right)_k\}$ are obtained. The whole training set can be obtained by combining $\mathbf{u}^{all}$ and the corresponding $\mathbf{vec}\left(\mathbf{A}_{adj}\right)$. After that we can train a SDA with the game dynamics data as the input and adjacent vector as the output. Apparently, we can reconstruct the entire network $\mathbf{A}_{adj}$ directly with the SDA. The specific algorithm process is shown in Algorithm 2.

Algorithm 2. Structure reconstruction of the game dynamics network based on the SDA

```
Input: The testing data set $\left(\mathbf{u}^{all}\right)_{test}$; the training data set
$\mathbf{D}_{train} = \{ \left((\mathbf{u}^{all})_1, (\mathbf{vec}(\mathbf{A}_{adj}))_1\right), \left((\mathbf{u}^{all})_2, (\mathbf{vec}(\mathbf{A}_{adj}))_2\right), \ldots,$
$\left((\mathbf{u}^{all})_k, (\mathbf{vec}(\mathbf{A}_{adj}))_k\right) \}$;
the training epoch $E$; the pre-train learning rate $l_{pre}$; the fine-tune learning rate $l_{fine}$
Initialization: parameters of three denoising autoencoder; the initial training epoch $e = 1$
Step 1:Pre-trained three denoising autoencoder
Using $\left(\mathbf{u}^{all}\right)_{train}$ as the input $\mathbf{x}_in$ of the first denoising autoencoder to obtain the output $\mathbf{y}_1$;
then using $\mathbf{y}_1$ as the input of the second denoising autoencoder to obtain the output $\mathbf{y}_2$; fina-
lly, using $\mathbf{y}_2$ as the input of the last denoising autoencoder to obtain the output $\tilde{\mathbf{x}}_{out}$
While $e \le E$ do
    Using Eq. (19) to minimize the mean square error between the output of the denoising autoencoder and the output of
    the denoising autoencoder for every denoising autoencoder to train three denoising autoencoder
end while
Step 2: Assign the encoder layer parameters of the three trained denoising autoencoder to
the first three layers of the SDA
Step 3: Fine-tuning
Using $\left(\mathbf{u}^{all}\right)_{train}$ as the input of the SDA to obtain the output $\tilde{\mathbf{x}}_{out}$ of the entire SDA
While $e \le E$ do
    Using Eq. (21) to minimize the mean square error between the output $\tilde{\mathbf{x}}_{out}$ of the entire SDA
    and the train data $\left(\mathbf{vec}(\mathbf{A}_{adj})\right)_{train}$ to train the SDA;
end while
Step 4: Using $\left(\mathbf{u}^{all}\right)_{test}$ as the input of the trained SDA to obtain the output $\mathbf{vec}\left(\mathbf{A}_{\mathbf{adj}}\right)'$, and convert the vector
$\mathbf{vec}\left(\mathbf{A}_{\mathbf{adj}}\right)'$ into the adjacent matrix $\mathbf{A}_{adj}'$
Output:The adjacent matrix $\mathbf{A}_{adj}'$ of complex network that need to be inferred
```

4. Simulation results

In this section, we first test the performance of the proposed method with two artificial networks: a scale-free network (Santos & Pacheco, 2005) shown in Fig. 2(a) and a small world network (Watts & Strogatz, 1998) shown in Fig. 2(b). Then the measurement noise is taken into account to verify the robustness of the method. At last, we scale up the model to solve the structure reconstruction problem of an empirical network.

4.1. Evaluation metrics for network reconstruction

Network structure reconstruction can be viewed as a binary classification problem with extremely unbalanced classes, that is, judging whether the edge exists or not. As the network is commonly sparse (most of the edges do not exist), a naive algorithm predicting all edges non-existed can still achieve high accuracy without reconstructing any edge. Thus two standard performance measures, the receiver operating characteristic (ROC) curve and the precision–recall (PR) curve (Davis & Goadrich, 2006), are used here to evaluate the performance for different methods. Besides, the area under ROC curve (AUROC) and the area under PR curve (AUPR) are used as aggregate indicators of the performance of the proposed algorithm. In addition, since the length of the recorded time series (L_D) is an important index for evaluating the efficiency

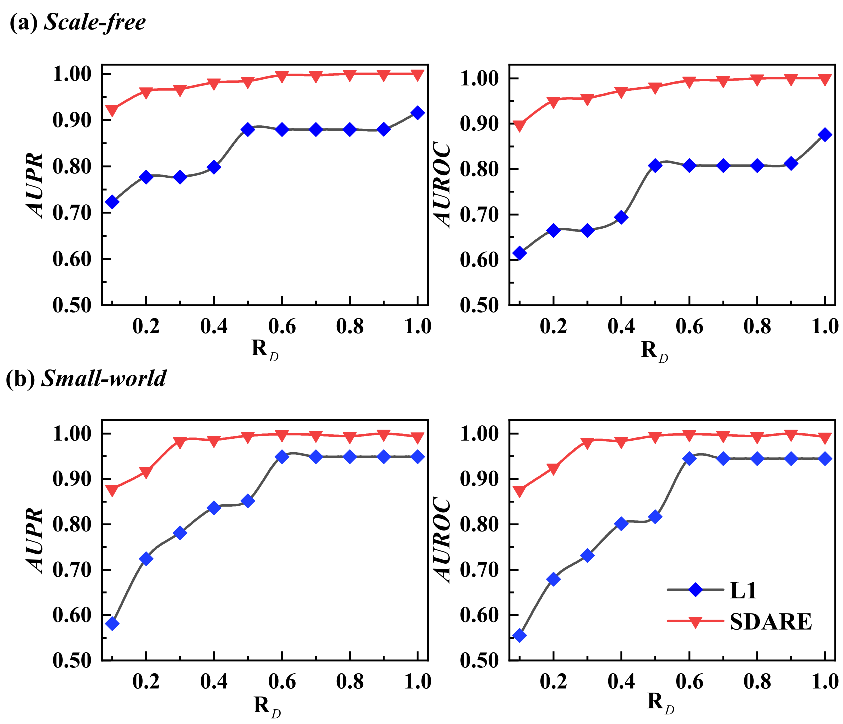


Fig. 3. AUPR and AUROC curves evaluating the performance of SDARE for a scale-free network (a) and a small-world network (b) without noise. Network size $N = 10$. High indexes indicate a strong capability of the proposed method in reconstructing the 10-nodes scale-free network and small world network without noise.

of the proposed method, we added a new parameter data rate, the ratio of the length of the time-series data to the number of nodes $R_D = \frac{L_D}{n}$, to evaluate the performance of the proposed method.

4.2. Reconstruct artificial networks without noise

We first test our method on the scale-free network and small-world network of 10-nodes respectively, and then scale up to 25-nodes. Here the proposed method is compared with the most commonly used compressive sensing method, which is called L_1 for short. For networks with different nodes, we use different numbers of training data to train the networks. Specifically, for the network of 10 nodes, 2000 pieces of data are used to train the SDA. Here the learning rates for pre-training and fine-tuning are set to 0.01, and the number of epochs is set to 100. For the network of 25 nodes, 10000 training data are used to train the SDA. The learning rates for pre-training and fine-tuning are set to 0.01 and 0.001 respectively, and the number of epochs is set to 200.

Fig. 3 illustrates the experimental results of reconstructing the scale-free network and the small-world network with 10 nodes. When the data rate $R_D = 0.1$, the AUPR and AUROC of SDARE are 20% and 28% higher than that of the L_1 method in the scale-free network reconstruction respectively. Similarly, for the small-world network reconstruction, the AUPR and AUROC of SDARE are 29% and 32% higher than that of the L_1 method. More importantly, when the data rate $R_D = 0.6$, the AUPR and AUROC of SDARE both reached 1.0 for the small-world network and the scale-free network reconstruction. This means the SDARE can reconstruct the networks accurately, which is much better than the L_1 method.

Then we scale up the results to the networks with 25-nodes, which is shown in Fig. 4. The reconstruction results are similar to that of the networks with 10-nodes. SDARE is much more accurate and efficient than the L_1 method for all the data rate. Even when the data rate is very small (such as 0.1), the SDARE can get good performance (the AUPR is above 0.7 and the AUROC is above 0.8).

Fig. 5 shows how the structure identification accuracy of a scale-free network changes with respect to the number of training data. Although an unstable phenomenon occurs due to the randomness of the experiment, it can be seen from Fig. 5 that the performance of SDARE is getting better and better as the amount of data increases. This means if we have enough training data, we can learn the accurate mapping from the game dynamics data to the adjacent vector.

In order to test the stability of our method to different training/test set partitioning, we reconstruct the scale-free network of 10 nodes by dividing the data into different training set and test set. Here we generate 2000 pieces of data for the scale-free network with $R_D = 0.5$. In each round of experiment, we randomly choose 1900 pieces of data as the training set and the rest data as the test set. The experiment is repeated six times and the results are shown in Fig. 6. It can be seen that the AUROC is basically stable at about 0.90, and the AUPR is stable at about 0.95. Therefore, our method has stable performance for different training/test set partitioning. In order to further verify the stability of the proposed method under different data rate, we selected 10 samples as the test set and 1990 samples as the training set. Fig. 7 shows that our method has good stability. AUROC and AUPR increase when the data rate increases. It also can be seen that both AUROC and AUPR are above 0.8. Therefore, the proposed method has stable performance for different training/test set partitioning and data rate.

4.3. Reconstruct artificial networks with noise

In practice, the measurements usually contain noise, here we also carried out experiments on the data with noise. The observational noise is assumed to follow Gaussian distributed $n \sim N\left(0, \sigma_N^2\right)$, with σ_N denoting the standard deviation of the noise. We add the Gaussian noise to the measurement data, and then reconstruct the network through the SDA trained by the

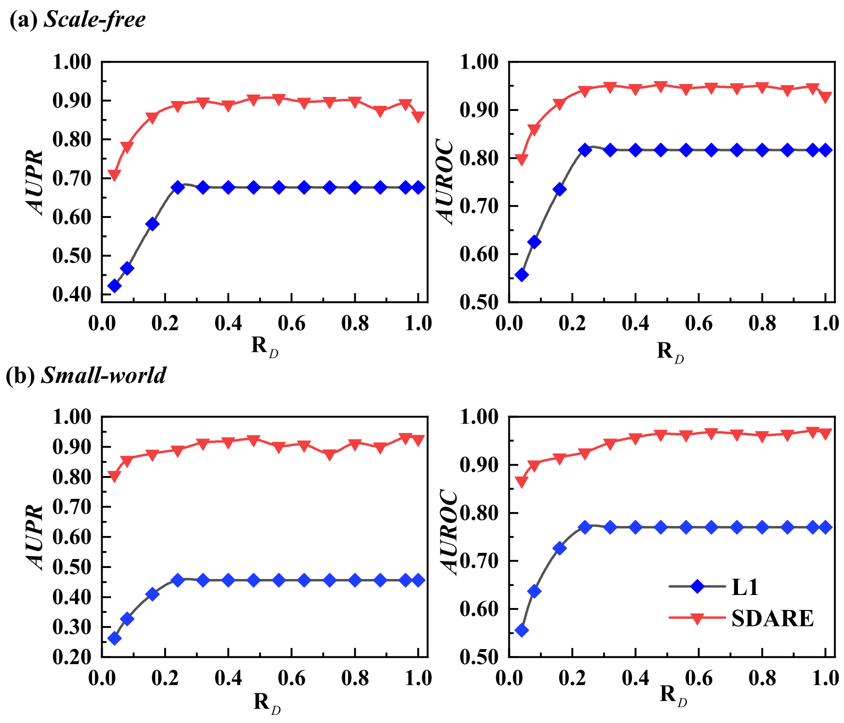


Fig. 4. AUPR and AUROC curves evaluating the performance of SDARE for a scale-free network (a) and a small-world network (b) without noise. Network size $N = 25$. The proposed method shows good performance for 25-nodes scale-free networks and small world networks.

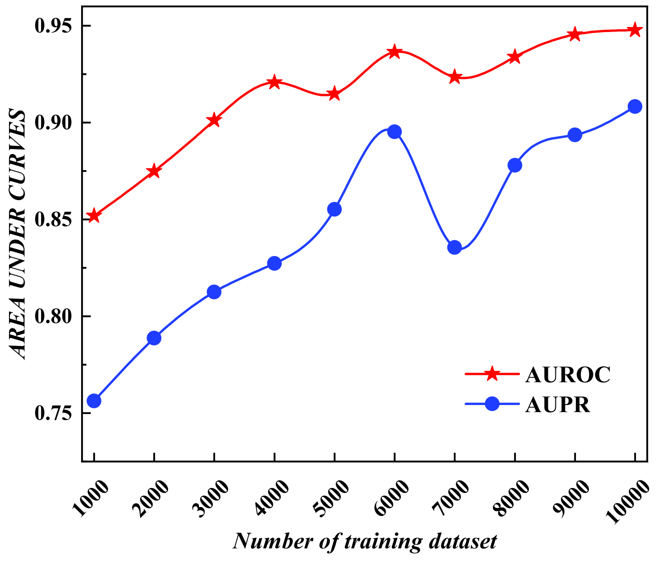


Fig. 5. The AUPR and AUROC of the SDARE method with respect to the number of training dataset without noise. Network size $N = 25$, data rate $R_D = 0.4$. As the number of training dataset increases, the reconstruction performance of the proposed method increases.

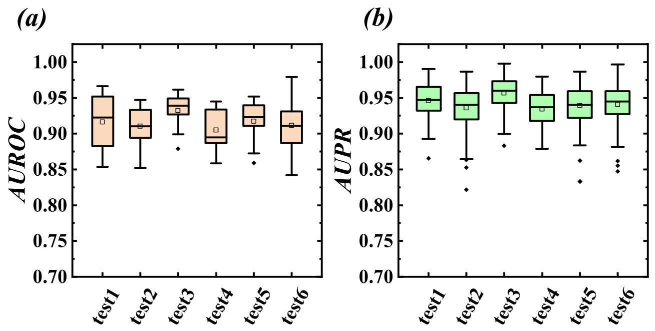


Fig. 6. The boxplots of scale-free network with different partitions. Network size $N = 10$, $R_D = 0.5$. The proposed method has stable performance for different training/test set partitioning.

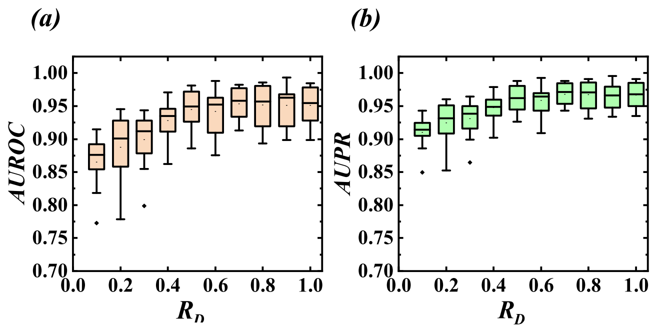


Fig. 7. The boxplots of scale-free network with different data rate. Network size $N = 10$. The proposed method has stable performance for different data rate.

noiseless data. Fig. 8 shows how the area under curve changes with the increase of noise amplitude σ_N. One can find that even when the noise amplitude $\sigma_N = 0.8$, the AUROC is still higher than 0.85 and the AUPR is still higher than 0.73. It is worth noting that the SDARE model is trained by the noiseless data, thus we can conclude that our method is robust against the strong observational noise.

We also compare the proposed method to the Lasso method on the evolutionary game data that are contaminated with noise. Here the network structure models include scale-free network and small-world network. Different from the last experiment, we trained the network with noisy data and test on noisy data. As shown in Figs. 9 and 10 , the AUPR and AUROC of SDARE

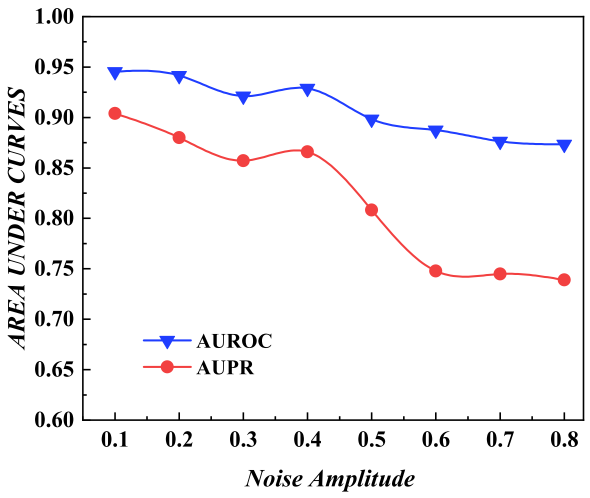


Fig. 8. SDARE is robust against strong noise in a scale-free network. Network size $N = 25$, data rate $R_D = 0.4$. High indexes of network reconstruction against strong noise perturbations indicate the robustness of the proposed method.

are higher than the Lasso method for both scale-free and small-world networks. When the data rate is above 0.3, the AUPR is larger than 0.75 for both the scale-free network and small-world network, and the AUROC is larger than 0.9. All these results show that our method can also get good performance even when the measurement data are contaminated with noise.

4.4. Reconstruct empirical networks

Lastly, the proposed SDARE method is testified for the structure reconstruction of an empirical network: the Zachary's karate club network (Zachary, 1977). Here, the Zachary's karate club network captures 34 members of a karate club, documenting pairwise links between members who interacted outside the club. One should note that there is only one actual network that needs to be tested. In order to train the SDA, we generate 15000 networks with the same degree distribution as Zachary's karate club network. Here the learning rates for pre-training and fine-tuning are set to 0.001 and the number of training epoch is set to 200. We carry out our experiments for both noisy and noiseless measurements data.

The results of structure identification for Zachary's karate club network are shown in Fig. 11. Although our method has similar performance to L_1 and Lasso when data rate is high, it is better than these two methods when data rate is low. Besides, it still has high reconstruction performance for noisy data. All these simulations indicate that our method can reconstruct the empirical network with satisfactory results.

5. Conclusion

In this paper, the SDA is proposed to reconstruct the network structure based on the evolutionary game data. It can learn the mapping between the local game dynamic data and the global network structure. Through extensive experiments, we prove that the proposed method outperform the previous methods based on compressive sensing, and it can also maintain good performance when the data rate is small. It is worth noting that, although the proposed method is tested for the game dynamics on the small-world network, the scale-free network and the Zachary's karate club network, it can also be applied to both more complex network structures and other temporal dynamics (Li, Wu, Liu, Lu, & Guo, 2015; Shen, Wang, Fan, Di, & Lai, 2014). In the future, we will explore how to absorb distributed learning into our framework to reconstruct large-scale complex network.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

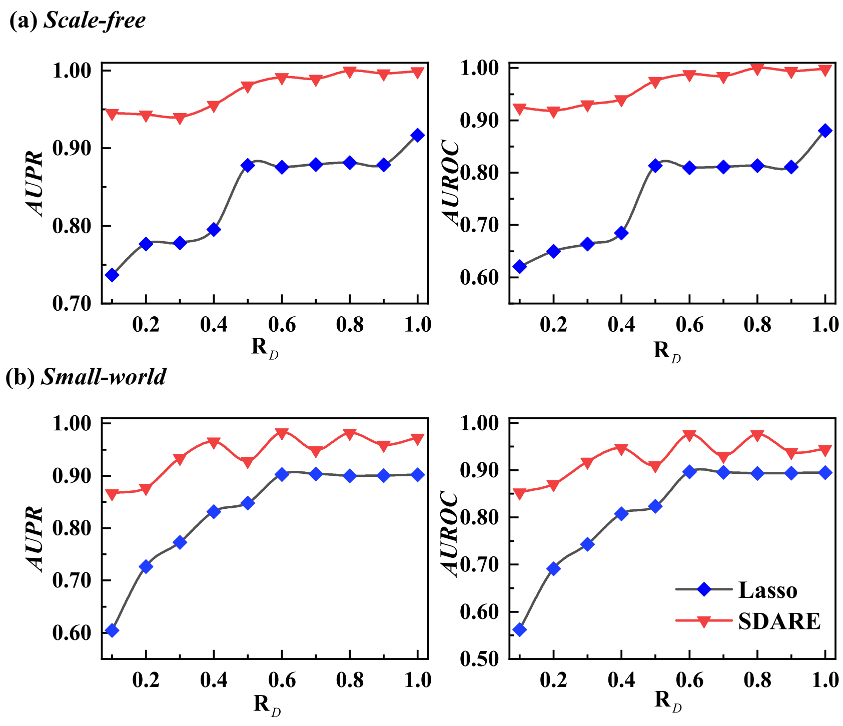


Fig. 9. The proposed method is robust against strong noise. PR and ROC curves for reconstructing a scale-free network (a) and a small-world network (b). Network size $N = 10$, standard deviation of noise $\sigma_N = 0.10$. High indexes of network reconstruction against strong noise perturbations indicate the robustness of the proposed method in 10-nodes scale-free and small world networks.

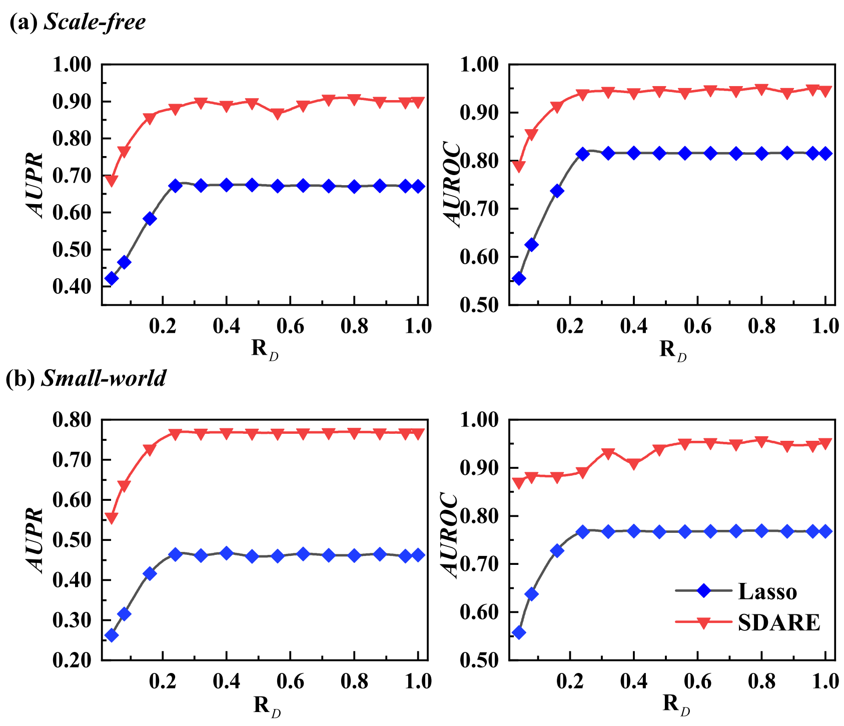


Fig. 10. The proposed method is robust against strong noise. PR and ROC curves for reconstructing a scale-free network (a) and a small-world network (b). Network size $N = 25$, standard deviation of noise $\sigma_N = 0.10$. The proposed method maintains good performance and robustness for 25-nodes scale-free and small-world networks with noise.

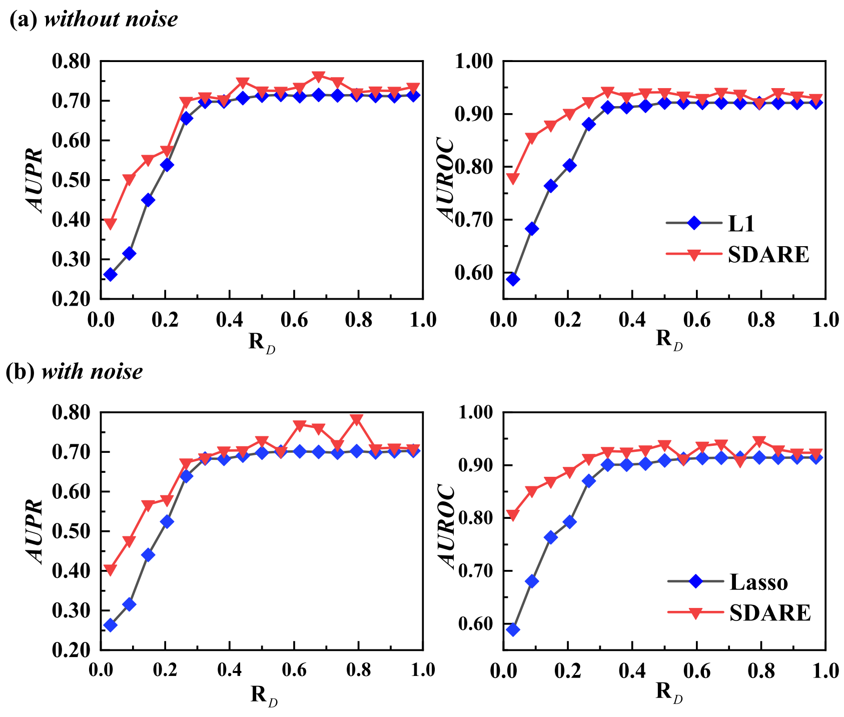


Fig. 11. AUPR and AUROC curves for structure identification of Zachary's karate club network without noise (a) and with noise (b), Noise amplitude $\sigma_N = 0.10$. The proposed method shows good reconstruction performance and robustness to noise.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grants No. 61703439, 61806011, 61802319, 61860206014, 61621062, U81961138010), in part by the Natural Science Foundation of Hunan Province, China (Grant No. 2019JJ50777), in part by the Innovation-Driven Project of Central South University, China (2019CX020) and in part by the 111 Project, China (B17048).

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 173–182).

Bansal, M., Gatta, G. D., & Di Bernardo, D. (2006). Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics*, *22*(7), 815–822.

Blumensath, T., & Davies, M. E. (2009). Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, *27*(3), 265–274.

Boccaletti, S., Bianconi, G., Criado, R., Del Genio, C. I., Gómez-Gardenes, J., Romance, M., et al. (2014). The structure and dynamics of multilayer networks. *Physics Reports*, *544*(1), 1–122.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., & Hwang, D.-U. (2006). Complex networks: Structure and dynamics. *Physics Reports*, *424*(4–5), 175–308.

Candes, E. J., & Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies?. *IEEE Transactions on Information Theory*, *52*(12), 5406–5425.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1251–1258).

Christoph, H., & Michael, D. (2004). Spatial structure often inhibits the evolution of cooperation in the snowdrift game. *Nature*, *428*(6983), 643.

Cussat-Blanc, S., Harrington, K., & Pollack, J. (2015). Gene regulatory network evolution through augmenting topologies. *IEEE Transactions on Evolutionary Computation*, *19*(6), 823–837.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 233–240). ACM.

Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197–387.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(12), 7821–7826.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649). IEEE.

Han, X., Shen, Z., Wang, W.-X., & Di, Z. (2015). Robust reconstruction of complex networks from sparse data. *Physical Review Letters*, *114*(2), 028701.

He, Y., She, Y., & Wu, D. (2013). Stationary-sparse causality network learning. *Journal of Machine Learning Research (JMLR)*, *14*(1), 3073–3104.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4700–4708).

Huang, K., Liu, Y., Zhang, Y., Yang, C., & Wang, Z. (2018). Understanding cooperative behavior of agents with heterogeneous perceptions in dynamic networks. *Physica A. Statistical Mechanics and its Applications*, *509*, 234–240.

Huang, K., Wang, Z., & Jusup, M. (2018). Incorporating latent constraints to enhance inference of network structure. *IEEE Transactions on Network Science and Engineering*.

Huang, K., Zhang, Y., Li, Y., Yang, C., & Wang, Z. (2018). Effects of external forcing on evolutionary games in complex networks. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *28*(9), 093108.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.

Li, H.-J., Bu, Z., Li, A., Liu, Z., & Shi, Y. (2016). Fast and accurate mining the community structure: integrating center locating and membership optimization. *IEEE Transactions on Knowledge and Data Engineering*, *28*(9), 2349–2362.

Li, G., Wu, X., Liu, J., Lu, J.-a., & Guo, C. (2015). Recovering network topologies via taylor expansion and compressive sensing. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *25*(4), 043102.

Liu, Y., Yang, C., Huang, K., & Wang, Z. (2019). Swarm intelligence inspired cooperation promotion and symmetry breaking in interdependent networked game. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *29*(4), 043101.

Lyu, C., Liu, Z., & Yu, L. (2019). Block-sparsity recovery via recurrent neural network. *Signal Processing*, *154*, 129–135.

Mei, G., Wu, X., Wang, Y., Hu, M., Lu, J.-A., & Chen, G. (2018). Compressive-sensing-based structure identification for multilayer networks. *IEEE Transactions on Cybernetics*, *48*(2), 754–764.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mousavi, A., Patel, A. B., & Baraniuk, R. G. (2015). A deep learning approach to structured signal recovery. In *The 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (pp. 1336–1343). IEEE.

Napoletani, D., & Sauer, T. D. (2008). Reconstructing the topology of sparsely connected dynamical networks. *Physical Review E*, *77*(2), 026103.

Packard, N. H., Crutchfield, J. P., Farmer, J. D., & Shaw, R. S. (1980). Geometry from a time series. *Physical Review Letters*, *45*(9), 712.

Perc, M., Gómez-Gardeñes, J., Szolnoki, A., Floría, L. M., & Moreno, Y. (2013). Evolutionary dynamics of group interactions on structured populations: a review. *Journal of the Royal Society Interface*, *10*(80), 20120997.

Perc, M., & Grigolini, P. (2013). Collective behavior and evolutionary games-an introduction. arXiv preprint arXiv:1306.2296.

Perc, M., & Szolnoki, A. (2008). Social diversity and promotion of cooperation in the spatial prisoner's dilemma game. *Physical Review E*, *77*(1), 011904.

Perc, M., & Szolnoki, A. (2010). Coevolutionary games—a mini review. *BioSystems*, *99*(2), 109–125.

Santos, F. C., & Pacheco, J. M. (2005). Scale-free networks provide a unifying framework for the emergence of cooperation. *Physical Review Letters*, *95*(9), 098104.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Shen, Z., Wang, W.-X., Fan, Y., Di, Z., & Lai, Y.-C. (2014). Reconstructing propagation networks with natural diversity and identifying hidden sources. *Nature Communications*, *5*, 4323.

Szabó, G., & Fath, G. (2007). Evolutionary games on graphs. *Physics Reports*, *446*(4–6), 97–216.

Tang, W. K., Yu, M., & Kocarev, L. (2007). Identification and monitoring of biological neural network. In *IEEE International Symposium on Circuits and Systems, 2007* (pp. 2646–2649). IEEE.

Tanimoto, J., Brede, M., & Yamauchi, A. (2012). Network reciprocity by coexisting learning and teaching strategies. *Physical Review E*, *85*(3), 032101.

Wang, W.-X., Lai, Y.-C., Grebogi, C., & Ye, J. (2011). Network reconstruction based on evolutionary-game data via compressive sensing. *Physical Review X*, *1*(2), 021021.

Wang, S., & Rahnavard, N. (2013). Binary compressive sensing via sum of l1-norm and l (infinity)-norm regularization. In *Military communications conference, MILCOM 2013–2013 IEEE* (pp. 1616–1621). IEEE.

Wang, L., & Wu, J. T. (2018). Characterizing the dynamics underlying global spread of epidemics. *Nature Communications*, *9*(1), 218.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440.

Wu, X., Zhao, X., Lü, J., Tang, L., & Lu, J.-a. (2016). Identifying topologies of complex dynamical networks with stochastic perturbations. *IEEE Transactions on Control of Network Systems*, *3*(4), 379–389.

Xu, Q., Su, Z., Zhang, K., Ren, P., & Shen, X. S. (2015). Epidemic information dissemination in mobile social networks with opportunistic links. *IEEE Transactions on Emerging Topics in Computing*, *3*(3), 399–409.

Yu, W., Cao, J., Chen, G., Lu, J., Han, J., & Wei, W. (2009). Local synchronization of a complex network model. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *39*(1), 230–241.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*(4), 452–473.