Machine learning dynamical phase transitions in complex networks

Qi Ni,[1] Ming Tang,[2,3,*] Ying Liu,[4,5] and Ying-Cheng Lai[6]

[1]*School of Information Science and Technology, East China Normal University, Shanghai 200241, China*
[2]*School of Mathematical Sciences, Shanghai Key Laboratory of PMMP, East China Normal University, Shanghai 200241, China*
[3]*Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University, Shanghai 200241, China*
[4]*School of Computer Science, Southwest Petroleum University, Chengdu 610500, China*
[5]*Big Data Research Center, University of Electronic Science and Technology of China, Chengdu 610054, China*
[6]*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, Arizona 85287, USA*

(Received 2 April 2019; revised manuscript received 19 August 2019; published 26 November 2019)

Recent years have witnessed a growing interest in using machine learning to predict and identify critical dynamical phase transitions in physical systems (e.g., many-body quantum systems). The underlying lattice structures in these applications are generally regular. While machine learning has been adopted to complex networks, most existing works concern about the structural properties. To use machine learning to detect phase transitions and accurately identify the critical transition points associated with dynamical processes on complex networks thus stands out as an open and significant problem. Here we develop a framework combining supervised and unsupervised learning, incorporating proper sampling of training data set. In particular, using epidemic spreading dynamics on complex networks as a paradigmatic setting, we start from supervised learning alone and identify situations that degrade the performance. To overcome the difficulties leads to the idea of exploiting confusion scheme, effectively a combination of supervised and unsupervised learning. We demonstrate that the scheme performs well for identifying phase transitions associated with spreading dynamics on homogeneous networks, but the performance deteriorates for heterogeneous networks. To strive to meet this challenge leads to the realization that sampling the training data set is necessary for heterogeneous networks, and we test two sampling methods: one based on the hub nodes together with their neighbors and another based on k-core of the network. The end result is a general comprehensive machine learning framework for detecting phase transition and accurately identifying the critical transition point, which is robust, computationally efficient, and universally applicable to complex networks of arbitrary size and topology. Extensive tests using synthetic and empirical networks verify the virtues of the articulated framework, opening the door to exploiting machine learning for understanding, detection, prediction, and control of complex dynamical systems in general.

DOI: 10.1103/PhysRevE.100.052312

I. INTRODUCTION

A research frontier across many disciplines of science and engineering is machine learning [1]. In physics, machine learning has attracted a great deal of attention because of its demonstrated ability to detect, predict, and uncover various phases of matter in quantum many-body systems [2–13]. Not only can neural-network based machine learning generate phases or states of matter that are already known [4,5,7,12] or uncover phase transitions [2,3,6], it can also predict out-of-equilibrium phases of matter that have not been previously known [13]. In most existing studies, the paradigmatic setting where machine learning, supervised or unsupervised, has been demonstrated to be effective and powerful is the Ising type of spin dynamics on a lattice. For example, the principal component analysis (PCA) [14] for dimension reduction of data set was exploited to uncover phase transitions with unsupervised learning, where samples with low and high temperatures were found to concentrate or distribute in different regions of the learning space [2]. It was also demonstrated that the threshold or the critical phase transition point of the Ising model can be predicted through deep learning [15], an important class of machine learning, where feed-forward neural networks or convolutional neural networks were employed to extract the necessary structural information [6,7]. All these accomplishments benefited greatly from the regular topology of the underlying Ising spin lattice.

Since the late 1990s, research on complex networks has yielded unprecedented insights into the working of a large variety of natural, social, and engineering systems [16]. A complex network, by definition, has a complex topology and, as a natural phenomenon associated with network dynamics, the emergence of distinct phases and phase transitions are ubiquitous [17]. The main question to be addressed in this paper is *Can phase transitions associated with dynamical processes in complex networks be "machine learned"?*

While there were recent efforts in incorporating machine learning into complex networks [18–22], the studies were limited to learning the *structural information* of the network. For example, network representation learning in which a complex network is dimensionally reduced with most of its structure information intact has found applications in problems such as link prediction [23], clustering [24], and node

*tangminghan007@gmail.com

classification [25]. Based on random-walk and graph search algorithms, the algorithm named "node2vec" can embed the network topology into a lower-dimensional space by integrating macroscopic and microscopic structural information about the network [19]. The algorithm "Deepwalk" finds a way to unite skipgram, a natural language processing technique, with random-walk sequences on graphs [18], providing a concise solution to graph embedding. Furthermore, it was proved that the performances of deep learning models such as graph convolutional neural networks [20] and structural deep network embedding [21] can be equivalent to those of traditional, random-walk-based methods. The key aspect that distinguishes our present work from the previous works is that we exploit machine learning to deal with *dynamics* responsible for phase transitions on complex networks.

To be concrete, we exploit machine learning to predict phase transitions associated with a fundamental type of dynamics on complex networks: epidemic spreading [26–33]. The dynamical process typically exhibits a second-order phase transition as in the Ising model, and to accurately identify the threshold or critical point of the phase transition has been an active research topic. A widely studied method is the "degree-based mean field" approach [17,26], which gives the theoretical threshold as $\langle k\rangle/\langle k^2\rangle$, where $\langle k\rangle$ and $\langle k^2\rangle$ are the first and second moments of the degree distribution, respectively. Strictly, the theoretical prediction is valid only in the limit of infinite network size, so for any real-world networks, there is always a discrepancy between the predicted and simulated threshold values, where the latter can be obtained, e.g., by using Monte Carlo simulations through measures such as network susceptibility [34], variability [35], or the average lifetime [36]. The basic idea and working principle of our machine-learning-based approach differ fundamentally from those of the existing methods.

To apply machine learning to predicting the epidemic threshold, a difficulty must be overcome: A complex network has a hierarchy of structural irregularities and contains rich features such as hubs, k-cores, and communities, making it challenging for machine learning to grasp the structural and dynamical information. For example, the PCA method that is effective for regular lattices usually fails to exhibit any clustering behavior for complex networks, on which the effectiveness of the learning algorithm depends. Compounding this difficulty is the complicated interplay between network structure and dynamics. To meet the challenge, we develop a systematic learning framework. In particular, we adopt and combine two different learning methods—supervised and unsupervised learning—to identify the relation between the configuration data of all nodal states and the epidemic phase of the system. We demonstrate that, while supervised learning works well in ideal cases, there is lack of robustness in identifying the threshold when some labeling information about the training data set is incorrect or missing. The unsupervised learning method we adopt is *confusion scheme* [6], which can be used to identify the threshold without requiring any prior knowledge about the labels. We find that, while this scheme works well for homogeneous networks, it is largely ineffective for heterogeneous networks. The origin of this difficulty can be understood by a physical analysis of the relative roles of the hub nodes and the small-degree nodes in the spreading dynamics. Aided by this understanding, we articulate two distinct sampling methods to render the confusion scheme applicable to heterogeneous networks: hub-and-neighbors and max-k-core sampling. We show that incorporating either sampling method can greatly improve the performance of deep learning in identifying the epidemic threshold for heterogeneous networks. For example, with sampling the algorithm underlying the confusion scheme is robust against noise and asymmetry of labeling information.

Overall, our deep learning framework is effective for different types of network topology, is robust, and is computationally efficient and consequently applicable to large networks. Our work has thus demonstrated that machine learning can be powerful for identifying the phase transition associated with epidemic *dynamics on complex networks of any topology*. Our framework combining supervised and unsupervised learning as well as incorporating sampling goes beyond the existing works on learning-based identification of matter phases in regular lattices in physics and those applicable only to detecting the structural information of complex networks in computer science. In fact, our deep learning framework provides the base for potential generalization to broad applications such as predicting the phases for more diverse types of dynamical processes on complex networks and identifying the matter phases in complex physical materials.

II. EPIDEMIC SPREADING AND DEEP LEARNING FRAMEWORK

We describe the basics of our machine learning framework for identifying phase transitions on complex networks, which include the epidemic spreading model, the structure of the training set, and the neural network model underlying deep learning.

A. Epidemic spreading and measure of susceptibility

We consider the classical susceptible-infected-susceptible (SIS) process on complex networks [16]. During the spreading, each infected node transmits the disease to its susceptible neighbors at the infection rate β, and the infected nodes return to the susceptible state at the recovery rate μ. There are two distinct phases associated with SIS dynamics: active and absorbing, where in the former there are both susceptible and infected nodes in the network but, for the latter, there is no longer any infected node. The effective infection rate is defined as the ratio of the infection rate to the recovery rate, $\lambda = \beta/\mu$, and the critical value of the effective infection rate is denoted as λ_c. For $\lambda < \lambda_c$, the system approaches the absorbing state after long time evolution. In this case, the system is in the absorbing phase. For $\lambda > \lambda_c$, asymptotically the system will enter into an endemic state where the density of the infected nodes reaches a stable value. In this case, the whole networked dynamical system is in the active phase.

We use the synchronous updated Monte Carlo method to simulate epidemic spreading processes in networks. In the absorbing phase, the nodal states are all identical, rendering them improper for training. Near the phase transition point,

i.e., when the value of λ is in the vicinity of λ_c ($|\lambda-\lambda_c| \gtrsim 0$), there is a high probability for the system to be trapped in the absorbing state. To overcome this difficulty, we use the quasistationary method [37] to prevent the system from entering the absorbing state. Especially, whenever the system tends to the absorbing state, we change its state to that of the previous time step.

The epidemic threshold of the SIS process can be conveniently characterized by the measure of susceptibility [34] defined as

$$\chi = N\frac{\langle\rho^2\rangle - \langle\rho\rangle^2}{\langle\rho\rangle}, \tag{1}$$

where ρ is the density of the infected nodes (i.e., the order parameter), N is the network size, $\langle\rho\rangle$ and $\langle\rho\rangle^2$ are the first and second moments of ρ, respectively. The order parameter associated with a second-order phase transition typically exhibits a power-law distribution near the critical point. As a function of the effective infection rate λ, the susceptibility measure reaches its maximum value at λ_c^{χ}, the threshold of the epidemic process or the phase transition point.

B. Training data set

For the underlying neural network to learn the system dynamics, a proper training data set is needed. To be specific, we assume that our data set is ordered along a tuning parameter λ, which is also the control parameter of the phase transition. The training data set with a certain value of λ is represented by $\mathbf{s}(\lambda)$. It consists of a set of binary vectors, expressed as $\mathbf{s}^{m\times N}(\lambda) = [a_1, a_2, \ldots, a_N]^{m\times 1}$, where m is the number of configurations including microcosmic dynamical states for all nodes at a time for a given λ, and a_i is the binary dynamical state of node i in one configuration. In particular, $a_i = 1$ means that node i is in the infected state while $a_i = 0$ indicates that i is susceptible. The system phase of every training data configuration is labeled by the function $l(\lambda) = H(\lambda - \lambda_c)$, where λ_c is the ground truth (or fake truth) that is preset in advance and H represents the unit step function $H(x) = \frac{d}{dx}\max\{x, 0\}, x \neq 0$. For any finite size network, we use label $l(\lambda) = 0$ to denote a system configuration being in absorbing state for $\lambda \leqslant \lambda_c$ and $l(\lambda) = 1$ for an active sate with $\lambda > \lambda_c$.

In the training process, a large data set of different $\mathbf{s}(\lambda)$ and corresponding labels $\mathbf{L}(\lambda)$ are fed into the neural network $\mathcal{F}$. The output of the neural network gradually approaches the preset label $\mathbf{L}(\lambda)$ through the training process called "back propagation" (see Appendix for detailed information). That is, we try to make $\mathcal{F}[\mathbf{s}(\lambda)] \to \mathbf{L}(\lambda)$ by optimizing the parameter set of the neural network $\mathbf{P}_{\mathcal{F}}$ through "back propagation" to minimize a cost function $C(\mathcal{F}[\mathbf{s}(\lambda)], \mathbf{L}(\lambda))$. The cost function is used to describe the mismatch between the network's outcome and the true answer quantitatively. It can be minimized by optimization methods such as stochastic gradient descent (SGD) [38] and adaptive moment estimation (ADAM) [39]. Actually the neural network is a high-level mapping function that takes data $\mathbf{s}(\lambda)$ to infer the probability distribution $\mathcal{F}[\mathbf{s}(\lambda)] = p_0$, where p_0 $(1 - p_0)$ represents the inferred probability that $\mathbf{s}(\lambda)$ is in the phase of absorbing (active) state.

C. Neural network architecture

The gist of our deep learning framework is binary classification in machine learning, adapted to dynamical processes on complex networks. The training data set is a set of feature-label pairs $(\mathbf{s_1}, l_1)$, $(\mathbf{s_2}, l_2)$, …, $(\mathbf{s_m}, l_m)$, where $\mathbf{s_i}$ is the feature vector and l_i is the corresponding ground truth (either 0 or 1 to represent two classes) label. A classifier $\mathcal{F}$ can be constructed by learning the hidden information in the data set, and is used to classify the new unlabeled data $\mathbf{s_{m+1}}$, which is also called "test data set."

The learning process is carried out by a feed-forward neural network (FFNN) that receives a set of labeled training data and passes the data from one layer to the next—a forward propagation process. The output layer generates results that can be compared with the given labels of the training set, triggering a back propagation process to minimize the cost function, which enables the weights (and biases) of each layer to be updated. More specifically, we construct our learning model with TensorFlow [40]. As illustrated in Fig. 1, the neural network consists of three dense layers. The input layer contains N neurons, where N is exactly the size of the complex network on which the epidemic dynamical process occurs. As the input goes in, the value in each input neuron is the epidemic state of a node (either 0 or 1). The hidden layer has 100 neurons, each being fully connected with the input layer. We impose the ReLU activation function on the hidden layer to achieve a nonlinear mapping of the input data to the layer [41,42]. The general advantage of ReLU is that it can prevent the detrimental phenomenon of gradient vanishing in learning and expedite the convergent process as compared with the sigmoid function. We use L2 regularization to avoid overfitting. The output layer has only one neuron whose output is constrained between zero and one by the sigmoid function, which represents the probability that the original complex networked system is in a certain phase. Specifically, if the output is 0 (1), then the neural network regards the state of the input data as belonging to the absorbing (active) phase with probability one. We use ADAM optimizer [39] to improve the learning efficiency. Some hyperparameters for the neural network are as follows: batch size $N_b = 128$, learning rate $\alpha = 0.001$, and regularization parameter $l2 = 0.01$ (see Appendix for details).

III. IDENTIFYING THRESHOLD THROUGH SUPERVISED LEARNING AND ISSUES

The principle to identify the threshold value through supervised learning can be described, as follows. Suppose a training data set in the matrix form as described in Sec. II B is available, where each row records the states of all nodes in the network at a given time (i.e., a given configuration). Suppose further that the corresponding label vector characterizing the state of each network configuration is available, as illustrated in Fig. 1. The aim of supervised learning is to identify the relation between data and labels. In particular, the training set is fed into the neural network in Fig. 1 to reveal some hidden patterns in the data. For $\lambda \ll \lambda_c$, after multiple times of training, the neural network can correctly classify the test set even without the labels. In this case, there is little confusion

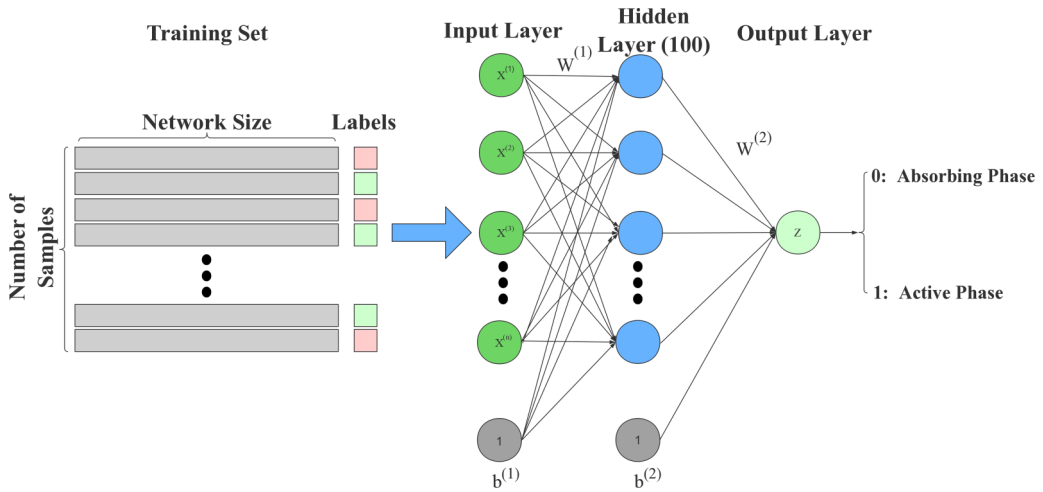

FIG. 1. Illustration of the neural network structure in our deep learning framework. Labeled data set is used as input and the learning process as illustrated is supervised. There is a hidden layer of 100 neurons and a single-neuron output layer to generate the probability of the identified phase of the dynamical process on a complex network.

for the neural network to recognize the phase associated with the dynamical process on the complex network, which can then be correctly recognized. For $\lambda = \lambda_c$, the average output of the last layer will be about 0.5, meaning that only half of the data are correctly classified. This corresponds to the case where the maximum amount of confusion arises, providing a criterion for the neural network to identify the threshold.

To gain insights, we test supervised learning on random regular networks. Figure 2(a) shows that, when the output of the neural network reaches the value of 0.5, the corresponding value of λ is quite close to the threshold value λ_c^X. Note that the identified threshold value depends on the label information in a given training data set. In an actual situation, we may not know all the label information of the training set, especially when the state of the underlying dynamical network is near the threshold. To simulate this situation, we deliberately change the boundary (i.e., a preset threshold) between the labels zero and one so as to make incorrect the label information of the training data between the preset and the true threshold values. For example, there are wrong labels for the training data between the green and blue dot-dashed lines in Fig. 2(a), where the preset threshold is 0.1025 while the true threshold is about 0.1050. As shown in Fig. 2(b), the identified threshold value via supervised learning deviates from the true threshold and increases with the preset threshold value. This means that wrong labeling information near the threshold can render supervised learning ineffective at identifying the threshold.

To overcome the difficulty associated with missing labels and/or the network spreading dynamics being near the threshold, we truncate the training data set. In particular, we remove the data points near the critical area and retain only those far away from the threshold value whose labels are less confused. We find that the neural network can still infer the information pertinent to missing data points by learning the retained data and identifying the threshold, as shown by the middle curve in Fig. 3(a). Another issue that may arise with data set in real applications is asymmetry in the truncated data set. To study how asymmetry affects the effectiveness of supervised learning, we shift the data set so that the threshold is no longer the center of the truncated area. Figure 3(b) shows that the identified threshold deviates markedly from the true value, indicating that supervised learning is sensitive to asymmetry in the truncated data set.

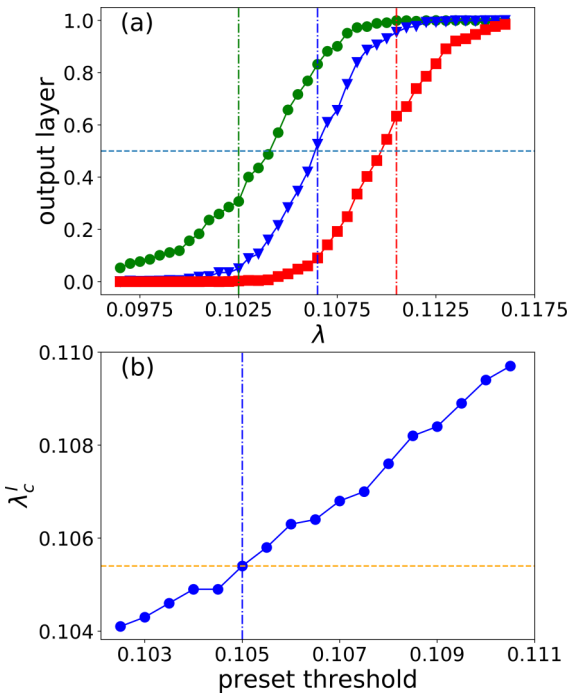

FIG. 2. Identifying phase transition on a random regular network through supervised learning. (a) The output of the neural network averaged over a test set versus the effective infection rate λ for different preset threshold values of the SIS spreading dynamics. For relatively smaller (larger) value of λ than the preset threshold value as marked by a vertical dashed line, the corresponding label will be zero (1). (b) The relationship between the identified and preset threshold values. The neural network's prediction depends on the preset threshold value. The structural parameters of the random regular network are $N = 1000$ and $\langle k \rangle = 10$.

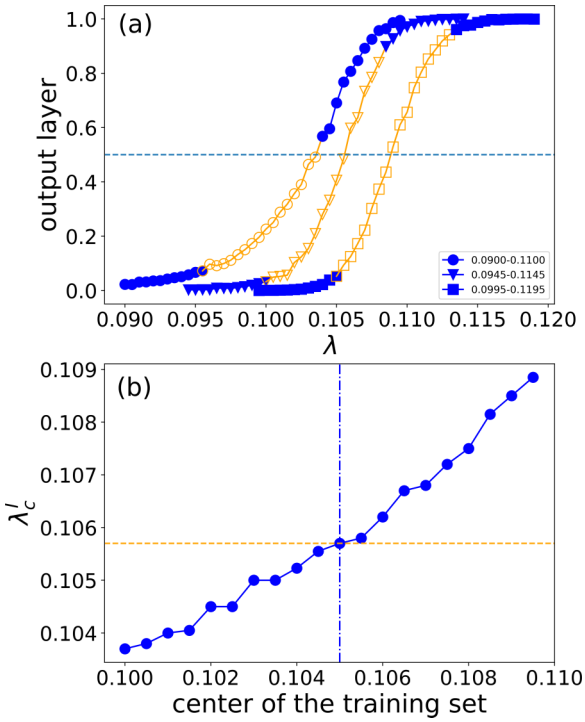

FIG. 3. Identification of phase transition through supervised learning with truncated training data set on a random regular network. (a) Threshold of phase transition identified via supervised learning with truncated data sets. Data points in the yellow area are artificially removed from the training set and the neural network makes the judgment only by learning the data in the blue area. The three curves show that the output shifts when the center λ value of the training set changes from 0.1 to 0.1095. (b) Robustness against asymmetry of data set. Shown is the relationship between the predicted threshold value and the center of the training set. When the range of the training set is shifted while keeping the number of data points in the training set unchanged, the identified threshold will change.

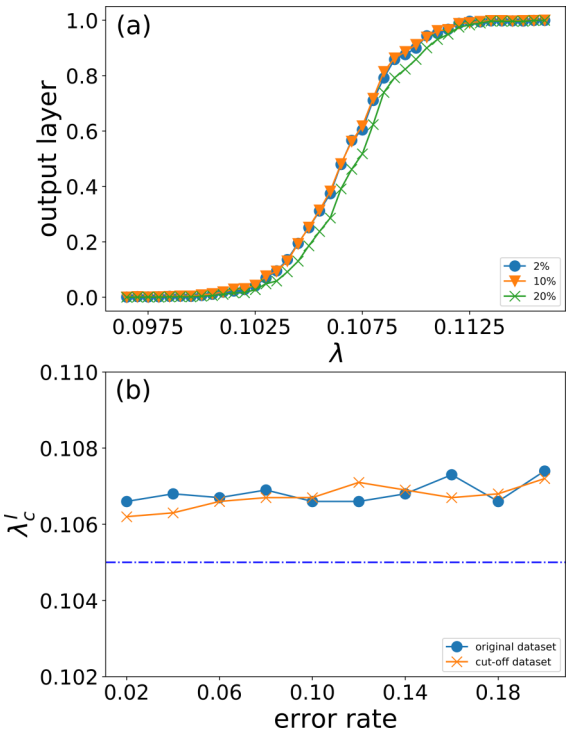

FIG. 4. Robustness of threshold identification via supervised learning against noise. For a random regular network, (a) the output curve (output of the neural network versus the effective infection rate) under different noisy inputs. Noise has little effect on the curve. (b) The relationship between the identified threshold and the error rate of the training labels. The predicted value oscillates about the output value with zero noise and exhibits a slightly upward trend as the labeling error rate is increased.

Is identification of phase transition through supervised learning robust against noise? To address this issue, we deliberately invert a proportion of the labels for the training data set and investigate whether the "artificial" noise can affect the predicted threshold value. Figures 4(a) and 4(b) demonstrate that the performance of supervised learning is robust against noise. Especially, as the labeling error rate increases, the output of the neural network oscillates but within a relatively small range.

The results in Figs. 2–4 indicate that supervised learning for identifying phase transition on random regular networks works well but only when the data near the threshold are removed. The resulting inevitable asymmetry in the truncated data set is detrimental, but noise has little effect on the performance of supervised learning.

IV. LEARNING BY CONFUSION SCHEME AND THE NECESSITY OF INCORPORATING SAMPLING

As demonstrated in Sec. III, the main deficiency of supervised learning is that it requires information about the labels to be accurate and complete and the training data set be symmetric with respect to the true threshold. To overcome these difficulties, we propose a general framework combining both supervised and unsupervised learning for accurate and efficient identification of dynamical phase transition in complex networks. We exploit the method of confusion scheme that can make precise prediction without any prior knowledge about the labels [6]. The confusion scheme is a supervised learning method incorporating some ingredients of unsupervised learning. The only difference between confusion scheme and supervised learning is that the labels of the confusion scheme method are human-made guesses instead of the ground truth. As our data are indexed by parameter λ, we can avoid using any prior knowledge by making a "good guess" of the true critical point, and the number of candidates in total is merely $\mathcal{M}+1$, where $\mathcal{M}$ is the number of different λ values existing in our data set. It is feasible to perform a "brute force" type of guess because the data set has been sorted, and the result can thus be inferred by the output accuracy of our neural network.

Suppose we have a set of unlabeled network configuration data ranging from $\lambda_{\min}$ to $\lambda_{\max}$. Let λ_c be the unknown true threshold value, where $\lambda_{\min} \leqslant \lambda_c \leqslant \lambda_{\max}$. We assign tentative labels to the data set by assuming that the threshold is λ_c', where $\lambda_{\min} \leqslant \lambda_c' \leqslant \lambda_{\max}$. Especially, we assign label zero to all configurations for $\lambda \leqslant \lambda_c'$ and label 1 to those with $\lambda > \lambda_c'$. We then choose a number of closely spaced values of λ_c'. For each value of λ_c', we conduct the training and obtain the

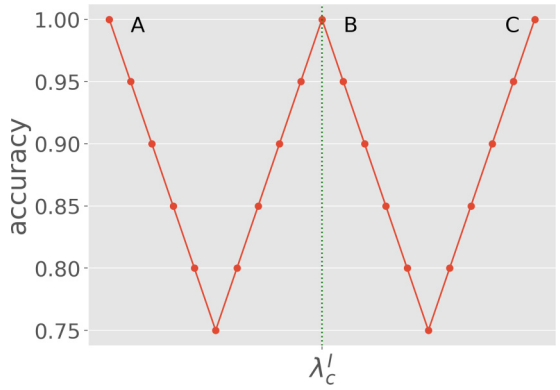

FIG. 5. Schematic illustration of the working of confusion scheme to detect a phase transition and to identify the transition point. For a range of tentatively assigned threshold values, the curve of classification accuracy of supervised learning versus the threshold value will exhibit a W shape if there is a phase transition associated with epidemic dynamics on the network. The location of the middle peak gives an accurate prediction of the true threshold value. If the network dynamics do not exhibit a phase transition, then the curve would exhibit a U shape (see text for a detailed reasoning).

classification accuracy. Ideally, we expect the accuracy to exhibit a W-shaped kind of behavior versus λ_c', as schematically illustrated in Fig. 5, which can be argued as follows. For $\lambda_c{}' = \lambda_{\min}$, every row of the training set is labeled as 1, so the neural network regards the data of every pattern as in the activation phase, giving rise to 100% accuracy of prediction. A similar result is expected for $\lambda_c' = \lambda_{\max}$. For $\lambda_c{}' = \lambda_c$, the method reduces to supervised learning because the tentative or "fake" labels happen to be correct under the circumstance. As described in Sec. III, the neural network can yield a high classification accuracy in this case. For $\lambda_{\min} < \lambda_c{}' < \lambda_c$ or $\lambda_c < \lambda_c{}' < \lambda_{\max}$, the neural network will be "confused" for some data whose labels are exactly opposite to the true values, thereby leading to a decrease in the accuracy. Overall, a W shape of the dependence of the accuracy on the value of λ_c' arises, where the location of the peak in the middle corresponds to the identified threshold. If there is no phase transition in the threshold range $[\lambda_{\min}, \lambda_{\max}]$, the accuracy versus λ_c' would exhibit a universal U shape. The emergence of a W-shaped curve is thus unequivocal indication that there is a phase transition in the system and the correct transition point (or threshold) can be identified accordingly without requiring any prior knowledge about the labels.

We test the confusion scheme on both homogeneous and heterogeneous complex networks. Figure 6(a) shows that the scheme can successfully identify the threshold on random regular networks. For this network topology and parameters, direct simulation of the SIS dynamics reveals a second-order phase transition at $\lambda \approx 0.1050$. When the "fake" threshold value λ_c' is varied in a range that contains the transition point, e.g., [0.096,0.116], the confusion scheme indeed yields an overall W-shaped type of behavior and the position of the middle peak occurs at $\lambda_c' \approx 0.1067$, which is indistinguishable from the actual threshold value. In general, the scheme is quite effective at detecting phase transition with an accurate identification of the transition point for homogeneous networks.

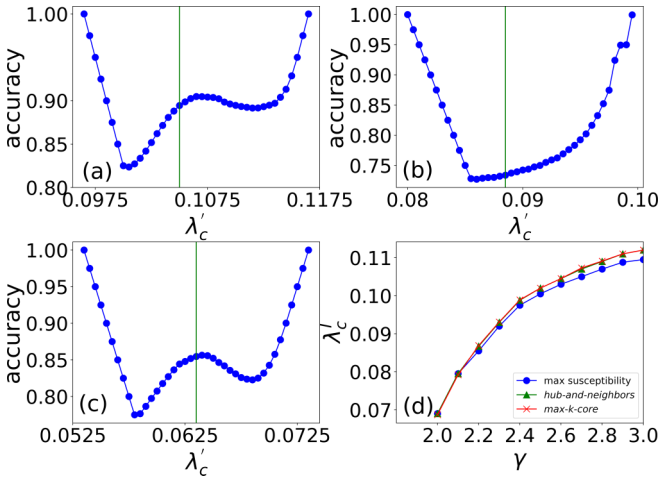

FIG. 6. Learning phase transition by confusion scheme on homogeneous and heterogeneous networks. (a) For a random regular network of size $N = 1000$ and average degree $\langle k \rangle = 10$, the confusion scheme generates an overall W-shaped curve of the classification accuracy versus the assumed threshold value. The peak of susceptibility is at 0.105, as indicated by the green vertical line (the same legend holds below). (b) For a scale-free network of size $N = 1000$ and structural parameter $m = m_0 = 3$ generated by the preferential attachment rule, the neural network fails to yield a W-shaped accuracy curve: The accuracy has a U shape. (c) For a scale-free network of parameters $N = 10\,000$ and $m = m_0 = 3$, the accuracy curve obtained by incorporating a hub sampling procedure into the confusion scheme. In this case, the accuracy curve exhibits a W shape, rendering detectable the phase transition. The transition point can also be identified accurately. (d) For scale-free networks generated by the uncorrelated configuration model of size $N = 10\,000$ with the value of the power-law exponent in the range $\gamma \in [2.0, 3.0]$, output of the confusion scheme, where the solid circles correspond to the maximum susceptibility, the triangles and crosses indicate the predictions given by neural network incorporating two sampling methods: hub-and-neighbors and max-k-core, respectively.

For heterogeneous networks such as scale-free networks, a direct application of the confusion scheme turns out not to be effective. As demonstrated in Fig. 6(b), the accuracy curve does not exhibit an apparent W shape in the chosen threshold range, whereas an actual phase transition occurs at $\lambda \approx 0.0885$. A possible reason is that, in a typical heterogeneous network, nodes of relatively large degrees are more prone to infection. Near the epidemic threshold, there are many nodes of small degrees which can hardly be infected. These nodes make the data set sparse by assuming the zero value most of the time and force the neural network to learn with an unbalanced data set. This will affect the learning process and prevent it from making the right decision.

To enhance the applicability of the confusion scheme for heterogeneous networks, we articulate to incorporate some proper, nodal importance-based sampling procedure into the scheme. Specifically, we aim to extract the state information of the important nodes and disregard that from the less important nodes. In the context of epidemic spreading, a straightforward criterion to determine the nodal importance can be obtained by addressing the key question: Who are the major spreaders? Intuitively, nodes with large degrees

are relatively more important in the spreading process [43]. To provide a physical reasoning, we consider the hub node with the largest degree in the network. If it is infected, then all of its neighbors are likely to be infected subsequently. However, importance of this sort will be greatly reduced if the hub node is located at the periphery of the network, implying that the hub nodes in the central area or core of the network would have greater importance [44,45], where the max k-core [46] can be used to define the core of the network. We henceforth propose two sampling methods: (1) to extract the state information of the hub node with a maximum degree and its neighbors (hub-and-neighbors sampling) and (2) to extract the information of the max-k-core subgraph (max-k-core sampling). As shown in Fig. 6(c), when the hub-and-neighbors sampling procedure is applied to the same scale-free network in Fig. 6(b), the neural network can detect the phase transition and identify the transition point. For the scale-free network, there are 314 nodes in the star graph that consists of the hub node and its neighbors. The peak value of susceptibility is about 0.0635 and the middle peak of the accuracy curve in Fig. 6(c) as generated by the confusion scheme is about 0.0642, which is the predicted transition point λ_c. In fact, after incorporating the sampling procedure, the learning results from the scale-free network are better than those for the homogeneous network as shown in Fig. 6(a).

To compare the performances of the two sampling methods, we generate an ensemble of scale-free networks using the uncorrelated configuration algorithm [16] whose degree exponent ranges from 2.0 to 3.0. As shown in Fig. 6(d), both sampling methods perform well, making the neural network powerful at identifying the epidemic threshold accurately. As both sampling methods give essentially the same performance, it suffices to focus on the sampling size (i.e., the size of the subgraph) to reduce the cost of computation time, which is desired for large networks.

To address the issue of asymmetric labels associated with supervised learning, we carry out a symmetry analysis of the confusion scheme. As shown in Figs. 7(a) and 7(b), the middle peak of the accuracy curve remains unchanged when the range of the training data set is shifted, indicating that introducing sampling into the confusion scheme can overcome the difficulty associated with asymmetry arising when no sampling is performed. Insofar as there is a phase transition in the given range, the threshold can be accurately identified. Figures 7(c) and 7(d) demonstrate the robustness of the confusion scheme against noise or mislabeling. As the proportion of the erroneous labels increases from 2% to 20%, the accuracy curve exhibits only small fluctuations, and the location of the middle peak does not change and can still be unequivocally identified.

V. APPLICATIONS TO REAL-WORLD NETWORKS

We test the performance of our framework combining supervised learning and the unsupervised confusion scheme on a number of real-world networks. Figures 8(a) and 8(b) illustrate the outputs of the neural network under supervised learning with two real-world data sets incorporating the two sampling methods. (From a computational standpoint, max-k-core sampling is often preferred because the hub-and-neighbors sampling method relies on the hub-and-neighbors star graphs whose sizes are typically much greater than the max-k-core.) Figures 8(c) and 8(d) demonstrate the accuracy of the confusion scheme on the same networks. The accuracies resulting from the two sampling methods are essentially the same. Performances of our framework on nine

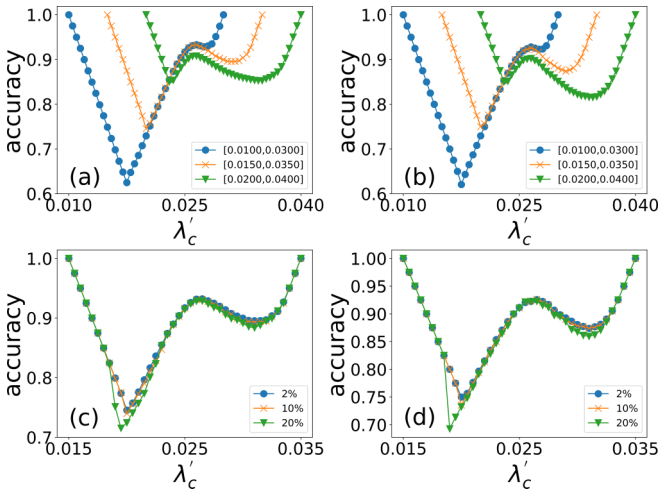

FIG. 7. Performance analysis of the confusion scheme for spreading dynamics on heterogeneous networks. The network is scale free with the degree exponent $\gamma = 2.2$. Symmetry analysis of accuracy incorporating (a) hub-and-neighbors and (b) max-k-core sampling methods. Different markers indicate different ranges of the training set. [(c) and (d)] Robustness analysis of the two sampling methods. Solid circles, crosses, and triangles represent the results for error rates of 2%, 10%, and 20%, respectively.

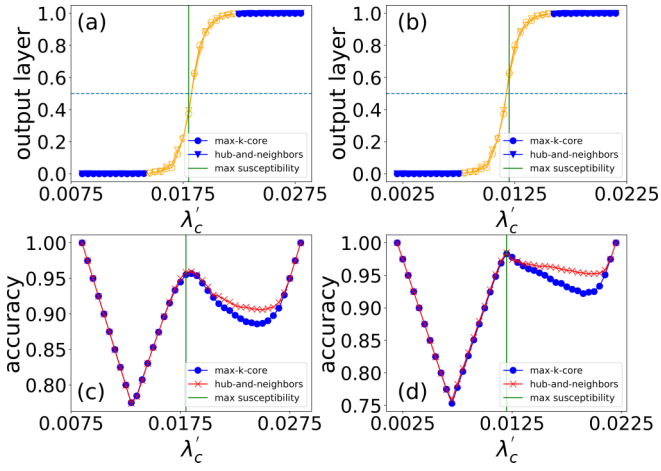

FIG. 8. Test on two representative real-world networks. [(a) and (b)] Output of the neural network under supervised learning incorporating max-k-core sampling for the corporative association for internet data analysis (CAIDA) and Brightkite data sets, respectively. Circles and triangles correspond to the results with max-k-core and hub-and-neighbors sampling, respectively. [(c) and (d)] Accuracy curves of executing the confusion scheme for the CAIDA and Brightkite data sets, respectively. For the CAIDA (Brightkite) network, the size of the subgraph sampled is 2628 (1135) with hub-and-neighbors sampling and 64 (154) with max-k-core sampling. Both sampling methods give essentially the same accuracy result.

TABLE I. Summary of performance results with nine real-world networks. The sampling method used is max-k-core and the star * indicates the threshold determined by this method.

Network	Size	Suscept.*	Supervised*	Confusion*
CAIDA	26475	0.0180	0.0183	0.0185
Brightkite	58228	0.0120	0.0118	0.0120
Astro-Ph	18771	0.0115	0.0117	0.0120
PGP	10680	0.0180	0.0318	0.0315
RV	6474	0.0180	0.0310	0.0310
Facebook	4039	0.0070	0.0070	0.0070
Gnutella4	10876	0.0635	0.0643	0.0645
Gnutella5	8846	0.0485	0.0497	0.0495
Gnutella6	8717	0.0530	0.0534	0.0540

real-world networks are summarized in Table I. The results with real-world networks thus confirm that our deep learning framework based on combined supervised and unsupervised learning and incorporating proper sampling is fully capable of ascertaining phase transition and identifying the transition point associated with spreading dynamics on complex networks with high accuracy and fidelity.

VI. DISCUSSION

To summarize, we have developed a deep learning framework to detect phase transition and to accurately identify the critical (transition or threshold) point associated with epidemic spreading dynamics on complex networks. The main motivations are twofold. First, in recent years there has been a great deal of attention in the physics community to exploiting machine learning for detecting phase transitions in many-body quantum systems, but the lattice structures underlying all existing works in this area are regular [2–13], raising the question of whether deep learning can be effective for identifying phase transitions in complex networks. Second, while deep learning has been introduced into the field of complex networks [18–22], the existing studies dealt exclusively with the structural properties. Physically, phase transitions in complex networks are often associated with certain dynamical processes. To apply machine learning to probe dynamical phase transitions in complex networks in terms of detection, prediction, and identification was then an unexplored territory, yet the problem is significant and challenging, especially from a physical point of view. Our work represents an initial effort in addressing this problem.

From the standpoint of methodological development, the innovative aspect of our framework is a combination of supervised and unsupervised learning, coupled with sampling methods tailored to complex networks. For a concrete dynamical process on complex networks, we focus on epidemic spreading. A straightforward application of supervised learning can be quite successful in detecting phase transition and predicting the critical threshold, provided that the labeling information is complete. When there is missing information about the labels, the performance of supervised learning tends to deteriorate, often significantly. Truncating the data set to remove those near the critical point helps to certain extent, but then the inevitable asymmetry in the data set can make the prediction unstable. To overcome these difficulties with supervised learning, we exploit the confusion scheme, a type of unsupervised learning, by which no prior knowledge about the labels is required. Operated on a systematically chosen set of threshold values, the confusion scheme constitutes essentially a series of supervised learning with different assumed knowledge and aims to locate the threshold value that leads to the maximum amount of "confusion." This combination of supervised and unsupervised learning performs well and is robust against data asymmetry and noise but only for homogeneous networks. For heterogeneous complex networks, the confusion scheme tends to fail due to the bias in the data set caused by the intrinsic structure of the network. We find that this difficulty can be overcome by incorporating a proper sampling schemes to prevent the training data set from being unbalanced. Two sampling methods have been tested: one based on hub nodes and their neighbors and another based on k-core, with both being quite effective at mitigating the problem of biased data set for heterogeneous networks. Because the size of the sampled data set is typically much smaller than the original data size, sampling has the additional benefit of significantly reducing the computational load while maintaining the desired accuracy. Through extensive tests on both synthetic and empirical networks, we conclude that our deep learning framework is capable of faithfully detecting phase transitions and accurately pinning down the critical transition point for epidemic spreading dynamics on complex networks.

Supervised method is a simple, computationally efficient and powerful solution for identifying a critical point. It does need precise labels to support; otherwise, its performance might deteriorate. Nevertheless, it is still useful because a supervised method can classify one-line input data, i.e., a snapshot of the nodal dynamical states vector in one time step, into two phases, which is impossible with the confusion scheme. To overcome the drawbacks of a supervised method, we can use a truncated data set or we can simply let a confusion scheme method generate the good labels first, and the classification problem can be effectively solved.

Although the susceptibility measure is simple, precise, and universally applicable in identifying the critical point, a large number of dynamical configurations are required to calculate the first- and second-order moments of ρ, especially near the critical point. Provided that there are no enough simulations, the location of the peak of the susceptibility curve will be uncertain, and thus the critical point cannot be identified accurately. As shown in Fig. 9, the susceptibility curve becomes volatile when the data are not sufficient, while the confusion scheme gives relatively stable results regardless of the small size of the data set. In general, the confusion scheme, due to its effective learning ability, outperforms the susceptibility measure in almost every aspect. The confusion scheme requires multiple cycles of supervised learning process, with every output value in the accuracy curve being independent of each other. This multiple-round deep learning method performs better than the single-round statistical result, especially when faced with a small data set. In experiments, there are imperfections such as missing or non-existent labels. As a result, an algorithm of identification of the deterministic nature may not be effective. In this case, the machine learning approach can correct the errors in an automated fashion. Our

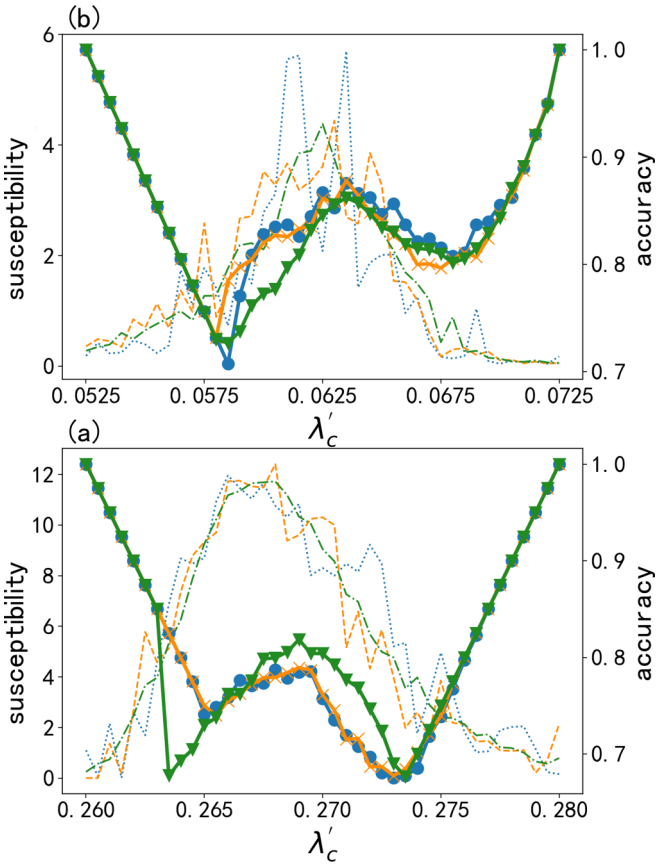

FIG. 9. Performance analysis with small data set. (a) Barabasi-Albert model with $N = 1000$ and $m = m_0 = 3$. (b) U.S. power grid network with 4941 nodes. In each panel, dotted, dashed, and dash-dot curves, respectively, represent the susceptibility values of data set with 10, 50, and 100 realizations for each λ, and blue circles, orange crosses, and green triangles, respectively, represent output values of data set with 10, 50, and 100 configurations for each λ by using confusion scheme.

work may have opened an avenue to exploit machine learning to solve challenging problems arising in the field of epidemic dynamics in complex networks.

Comparing with the traditional methods for identifying phase transitions in complex networks (e.g., numerical approaches based on susceptibility and other measures), our methods can not only identify the correct transition point but also classify the given input into different phases. Many open questions remain. For example, can the effect of asymmetrical training data set on supervised learning be mathematically analyzed? Can certain optimization methods be developed to increase the accuracy and reduce the computations in the execution of the confusion scheme? Can our framework of combining supervised and unsupervised learning be scaled to very large complex networks? Can a theory be developed to guide the sampling procedure for heterogeneous networks? Is it possible to develop machine learning methods to deal with phase transitions in time varying complex networks? We hope our work will stimulate further efforts in exploiting machine learning for detecting, decoding, predicting, and even controlling a variety of dynamical processes on complex networks.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grants No. 11975099, No. 11575041, and No. 61802321; the Natural Science Foundation of Shanghai under Grant No. 18ZR1412200; and the Science and Technology Commission of Shanghai Municipality under Grant No. 18dz2271000. Y.-C.L. acknowledges support from the Vannevar Bush Faculty Fellowship program sponsored by the Basic Research Office of the Assistant Secretary of Defense for Research and Engineering and funded by the Office of Naval Research through Grant No. N00014-16-1-2828.

APPENDIX

Here we list a number of basic notions, concepts, and methods in deep learning and complex networks, as well as a description of the real-world networks used in our study.

1. Feed-forward neural network

FFNN, also known as multilayer perceptron, is a basic and powerful deep learning model. The neural network is structurally dense because each node of the preceding layer is connected to all nodes in the next layer. The initial layer transmits information to the next layer through forward propagation, until the output layer is reached. The weights and biases of the whole network are updated backwards based on descending the cost function along the gradient to find its global optimum (back propagation). Forward and backward propagation is executed iteratively until the network finds a global minimum of the cost function. At this point, the network has successfully learned the information hidden in the training data set [47,48].

FFNN minimizes the cost function, which is used to describe the mismatch between the output and the preset label. Let w_{jk}^l be the weight on the edge from the kth neuron in layer $l-1$ to the jth neuron in layer l and b_j^l be the bias on the jth neuron in layer l. The input value of the jth neuron in layer l is $z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$ and its corresponding output is $a_j^l = \sigma(z_j^l)$ where the activation function σ is nonlinear and differentiable. There are many choices for the cost function, such as the quadratic cost function $C = \frac{1}{2n}\sum_x [y(x) - a^L(x)]^2$, where x, y, a^L, and L represent the input data, ground-truth labels, the output data, and the maximum number of layers of the FFNN, respectively. After an optimization target (i.e., cost function) is built, we use the gradient descend method to minimize it until it reaches the global minimum. Let δ_j^l be the error generated by the jth neuron in layer l, which is the deviation between the actual and predicted value. The error in the last layer can be obtained from the following equations:

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{A1}$$

or

$$\delta^L = \frac{\partial C}{\partial a^L} \odot \frac{\partial a^L}{\partial z^L} = \nabla_a C \odot \sigma'(z^L), \tag{A2}$$

where $\odot$ represents the Hadamard product, an element-wise matrix product. After the error in the last layer has been

calculated, errors in other layers can be obtained recursively as

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l}$$
$$= \sum_k \delta_k^{l+1} \frac{\partial \left(w_{kj}^{l+1} a_j^l + b_k^{l+1}\right)}{\partial a_j^l} \sigma'\left(z_j^l\right)$$
$$= \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'\left(z_j^l\right) \tag{A3}$$

or

$$\delta^l = [(w^{l+1})^T \delta^{l+1}] \odot \sigma'(z^l). \tag{A4}$$

Hence, the gradient of weight w and bias b can be resolved and the gradient descend method can be used to minimize the overall cost function as

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l \frac{\partial \left(w_{jk}^l a^{l-1} k + b_j^l\right)}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{A5}$$

and

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial \left(w_{jk}^l a_k^{l-1} + b_j^l\right)}{\partial b_j^l} = \delta_j^l. \tag{A6}$$

2. L2 regularization

When training a deep learning model, it is essential to prevent the model from overfitting the training set. An overfitted model has no practical usage because of lack of generalizability. There are different ways to deal with the problem such as data augmentation and regularization. For L2 regularization, a regularization term is added to the cost function: $C = C_0 + [\lambda/(2n)] \sum_\omega \omega^2$, where C_0 is the original cost function, $[\lambda/(2n)] \sum_\omega \omega^2$ is the regularization term, and ω's are the weights. We can keep the weights small during the training process to prevent overfitting.

3. Learning rate

This is an important hyperparameter that controls the speed at which the weights of the neural network are adjusted based on the loss gradient. It is a generic parameter in most optimization algorithms such as SGD and ADAM. The learning rate directly affects how fast the neural network can converge to a global minimum with the highest possible accuracy. In general, the greater the learning rate, the faster the neural network learns. If the learning rate is too small, then the network is likely to fall into some local optimum. However, if the rate is too large and exceeds some threshold value, then the loss of the cost function will oscillate and stop decreasing.

4. Batch size

If it is not possible to pass data through the neural network at once, it will be necessary to divide the data set into several batches. Batch size is a hyperparameter that will affect the training time, which is essential to finding an optimal value.

5. Uncorrelated configuration model

The uncorrelated configuration model (UCM) is a model for generating complex networks of arbitrary degree distributions [49]. In our work, the UCM is used to generate random uncorrelated scale-free networks with a prespecified degree distribution exponent. The algorithm consists of two steps. The first step is to assign to each vertex i in a set of N initially disconnected vertices a degree k_i, extracted from the probability distribution $P(k) \sim k^{-\gamma}$ and subject to the constraints $m \leqslant k_i \leqslant N^{1/2}$ and $\sum_i k_i$ even. The second step is to construct the network by randomly connecting the vertices with $\sum_i k_i/2$ edges, respecting the preassigned degrees and avoiding multiple and self-connections.

6. *k*-core decomposition

For a given complex network, k-core is the subgraph in which all nodes have at least k neighbors [44,45,50]. It characterizes the nodal importance to some extent. To find the k-core of a certain value of k, one removes from the network all nodes of degree less than k. Some of the remaining nodes may have a degree less than k after the removal, in which case one keeps removing nodes until no node in the core has degree less than k. The result, if it exists, is the k-core subgraph.

7. Real-world networks

a. CAIDA [51]. This is the undirected network of autonomous systems of the Internet from the CAIDA project, collected in 2007. Nodes are autonomous systems and edges represent communication.

b. Brightkite [52]. This undirected network contains user-user friendship relations from Brightkite, a former location-based social network were users share their locations. A node represents a user and an edge indicates that a friendship exists between a pair of users.

*c. Astro−*Ph [51]. This is the collaboration network of authors of scientific papers from the arXiv's Astrophysics (astro-ph) section. An edge between two authors represents a joint publication.

d. Pretty Good Privacy [53]. This is the interaction network of users of the Pretty Good Privacy algorithm. The network has only one giant connected component.

e. Route Views [51]. This is an undirected network of the autonomous system of the Internet.

f. Facebook [54]. This data set consists of "circles" (or "friend lists") from Facebook. Facebook data were collected from survey participants using this Facebook app. The data set includes nodal features (profiles), circles, and ego networks.

g. Gnutella [55]. This is a sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 2002. There are altogether nine snapshots of the Gnutella network collected in August 2002. Nodes represent hosts in the Gnutella network and edges are connections between the hosts.

[1] M. I. Jordan and T. M. Mitchell, Science **349**, 255 (2015).
[2] L. Wang, Phys. Rev. B **94**, 195105 (2016).
[3] T. Ohtsuki and T. Ohtsuki, J. Phys. Soc. Jpn. **85**, 123706 (2016).

[4] F. Schindler, N. Regnault, and T. Neupert, Phys. Rev. B **95**, 245134 (2017).
[5] Y. Zhang, R. G. Melko, and E.-A. Kim, Phys. Rev. B **96**, 245119 (2017).
[6] E. P. Van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Nat. Phys. **13**, 435 (2017).
[7] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017).
[8] G. Carleo and M. Troyer, Science **355**, 602 (2017).
[9] Y. Zhang and E.-A. Kim, Phys. Rev. Lett. **118**, 216401 (2017).
[10] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 041101(R) (2017).
[11] D.-L. Deng, X. Li, and S. Das Sarma, Phys. Rev. X **7**, 021021 (2017).
[12] D.-L. Deng, X. Li, and S. Das Sarma, Phys. Rev. B **96**, 195145 (2017).
[13] J. Venderley, V. Khemani, and E.-A. Kim, Phys. Rev. Lett. **120**, 257204 (2018).
[14] J. Shlens, arXiv:1404.1100.
[15] A. C. Courville, I. Goodfellow, and Y. Bengio, *Deep Learning* (MIT Press, Cambridge, MA, 2015).
[16] M. E. J. Newman, *Networks: An Introduction* (Oxford University Press, Oxford, UK, 2010).
[17] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, Rev. Mod. Phys. **80**, 1275 (2008).
[18] B. Perozzi, R. Al-Rfou, and S. Skiena, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM Press, New York, 2014), pp. 701–710.
[19] A. Grover and J. Leskovec, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM Press, New York, 2016), pp. 855–864.
[20] T. N. Kipf and M. Welling, arXiv:1609.02907.
[21] D. Wang, P. Cui, and W. Zhu, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM Press, New York, 2016), pp. 1225–1234.
[22] W. L. Hamilton, R. Ying, and J. Leskovec, arXiv:1709.05584.
[23] D. Liben-Nowell and J. Kleinberg, J. Am. Soc. Inf. Sci. Tech. **58**, 1019 (2007).
[24] C. H. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, in *Proceedings of the IEEE International Conference on Data Mining 2001 (ICDM'01)* (IEEE, Los Alamitos, CA, 2001), pp. 107–114.
[25] S. Bhagat, G. Cormode, and S. Muthukrishnan, in *Social Network Data Analytics* (Springer, Berlin, 2011), pp. 115–148.
[26] R. Pastor-Satorras and A. Vespignani, Phys. Rev. Lett. **86**, 3200 (2001).
[27] R. Pastor-Satorras and A. Vespignani, Phys. Rev. E **63**, 066117 (2001).
[28] M. E. J. Newman, Phys. Rev. E **66**, 016128 (2002).
[29] A. Vespignani, Science **325**, 425 (2009).
[30] A. Vespignani, Nat. Phys. **8**, 32 (2012).
[31] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, Rev. Mod. Phys. **87**, 925 (2015).
[32] M. De Domenico, C. Granell, M. A. Porter, and A. Arenas, Nat. Phys. **12**, 901 (2016).
[33] W. Wang, M. Tang, H. E. Stanley, and L. A. Braunstein, Rep. Prog. Phys. **80**, 036603 (2017).
[34] S. C. Ferreira, C. Castellano, and R. Pastor-Satorras, Phys. Rev. E **86**, 041125 (2012).
[35] P. Shu, W. Wang, M. Tang, and Y. Do, Chaos **25**, 063104 (2015).
[36] M. Boguná, C. Castellano, and R. Pastor-Satorras, Phys. Rev. Lett. **111**, 068701 (2013).
[37] R. S. Sander, G. S. Costa, and S. C. Ferreira, Phys. Rev. E **94**, 042308 (2016).
[38] L. Bottou, in *Neural Networks: Tricks of the Trade* (Springer, Berlin, 2012), pp. 421–436.
[39] D. P. Kingma and J. Ba, in *Proceeding of the third International Conference for Learning Representations, San Diego, 2015* (ICLR, 2015).
[40] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, in *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)* (ACM SIG, Savannah, GA, 2016), Vol. 16, pp. 265–283.
[41] V. Nair and G. E. Hinton, in *Proceedings of the 27th International Conference on Machine Learning (ICML'10)* (Omnipress, Haifa, Israel, 2010), pp. 807–814.
[42] X. Glorot, A. Bordes, and Y. Bengio, in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* (JMLR W&CP, Ft. Lauderdale, FL, 2011), pp. 315–323.
[43] C. Castellano and R. Pastor-Satorras, Sci. Rep. **2**, 371 (2012).
[44] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, Nat. Phys. **6**, 888 (2010).
[45] Y. Liu, M. Tang, T. Zhou, and Y. Do, Sci. Rep. **5**, 9602 (2015).
[46] V. Batagelj and M. Zaveršnik, Adv. Data Anal. Class. **5**, 129 (2011).
[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Nature **323**, 533 (1986).
[48] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Redwood City, CA, 1991).
[49] M. Catanzaro, M. Boguná, and R. Pastor-Satorras, Phys. Rev. E **71**, 027103 (2005).
[50] S. B. Seidman, Soc. Netw. **5**, 269 (1983).
[51] J. Leskovec, J. Kleinberg, and C. Faloutsos, ACM Trans. Knowl. Discov. Data **1**, 2 (2007).
[52] E. Cho, S. A. Myers, and J. Leskovec, in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM Press, New York, 2011), pp. 1082–1090.
[53] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, Phys. Rev. E **70**, 056122 (2004).
[54] J. Leskovec and J. J. Mcauley, in *Advances in Neural Information Processing Systems* (Neural Information Processing Systems Foundation, Lake Tahoe, NV, 2012), pp. 539–547.
[55] M. Ripeanu, I. Foster, and A. Iamnitchi, arXiv:cs/0209028.