# Feature learning and network structure from noisy node activity data

Junyao Kuang [1,*] Caterina Scoglio [1] and Kristin Michel[2]

[1]*Department of Electrical and Computer Engineering, Kansas State University, Manhattan, Kansas 66506, USA*
[2]*Division of Biology, Kansas State University, Manhattan, Kansas 66506, USA*

In the studies of network structures, much attention has been devoted to developing approaches to reconstruct networks and predict missing links when edge-related information is given. However, such approaches are not applicable when we are only given noisy node activity data with missing values. This work presents an unsupervised learning framework to learn node vectors and construct networks from such node activity data. First, we design a scheme to generate random node sequences from node context sets, which are generated from node activity data. Then, a three-layer neural network is adopted training the node sequences to obtain node vectors, which allow us to construct networks and capture nodes with synergistic roles. Furthermore, we present an entropy-based approach to select the most meaningful neighbors for each node in the resulting network. Finally, the effectiveness of the method is validated through both synthetic and real data.

## I. INTRODUCTION

A network is a system-level view of pairwise interactions between nodes, genes, or elements in a complex system [1–14]. The first step in analyzing a networked system is to construct the network from data obtained with different technologies. In most cases, network structures can be determined through direct measurements, meaning that pairwise relationships between nodes can be observed directly. For instance, the edges in friendship networks can be probed through various ways, including using questionnaires, checking Facebook or Twitter friendship, and investigating face-to-face interactions [15–19]. As another example, edges in web graphs can be directly determined by checking if hyperlinks exist between web pages. However, there are cases where the relationships between nodes cannot be observed directly [20]. Instead, we may only have node activity data that reflect the properties of nodes from various aspects. In these cases, we need to estimate the underlying network structure from nodal data. Such a problem exists in many areas, including the construction of financial, biological, and climate networks [21–31]. In these areas, measurements of pairwise relationships are not always feasible [6,20,32]. Instead, we can conduct various experiments to measure node activities under different conditions [33].

This work develops a model to learn node representations from noisy and heterogeneous data and proposes an entropy-based method to extract network structures. Specifically, we investigate the problems of feature learning and network construction for gene co-expression data. Different high throughput technologies, including microarray and RNA-sequencing, allow simultaneously evaluating thousands of gene expression data. Usually, the data can be organized into a matrix that consists of rows representing $N$ genes (nodes) and columns representing $M$ experimental conditions. To construct a network from such expression data, we need to consider three problems. (1) The expression data, measured through different experimental technologies, are distributed in various ranges. For example, the raw expression values obtained from different versions of RNA-sequencing in different labs are dispersed from zero to tens of thousands and do not follow any specific distribution. (2) Missing values are frequently present in the dataset. Some experiments may only test a subset of genes for specific purposes, or some experimental data for some genes (nodes) are not available. (3) The levels of noise are not constant. For instance, the environments, such as humidity, temperature, and light intensity, could potentially influence the accuracy of the devices and the measured node activity data. The method for network construction is not allowed to be affected by missing values and noisy data.

There are diverse approaches aiming at constructing networks from nodal data. A significant volume of works uses the correlation coefficient to measure the degree to which a pair of nodes is related, and edges are selected by thresholding the correlation coefficients [34–36]. However, the drawbacks of the correlation methods are that: (1) the expression data are required to follow a (quasi-) normal distribution, (2) the correlation coefficients are significantly affected by outliers, and (3) the number of measured conditions and missing values substantially affect the results [34,35]. Mutual information (MI) and its variants are also used to construct gene co-expression networks. The MI models do not require the data to follow the normal distribution. Still, the MI models are even more complex, since we are expected to find the joint probability distribution for every pair of genes [37]. We need to solve the problems mentioned above before applying either of the two methods. To solve problems (1) and (3), some researchers have proposed using rescaling and normalization methodologies to obtain quasinormal distributed data from the raw node activity data [38]. According to Ref. [39], the

---
*kuang@ksu.edu

number of experimental conditions significantly influences the correlation coefficients under the null hypothesis that two nodes are not correlated. Theoretical analysis shows that correlations based on ten conditions tend to be higher than those computed with 50 conditions. Missing values lead to node pairs with a different number of paired elements, meaning that the node pairs with fewer paired elements are more likely to have high correlation coefficients. Therefore, some works use imputation or interpolation to solve problem (2) [40,41]. The complex data processing procedures pose a severe challenge for the principle of parsimony when we further study the resulting network structure [42].

Edge selection is another issue we need to consider when constructing networks from node activity data. Both correlation and MI methods return coefficients between $-1$ and $1$. Many researchers construct unweighted networks by applying a threshold to select edges of the network corresponding to node pairs with the highest coefficients. However, choosing a threshold is always tricky since a high threshold could generate singleton nodes, while a low threshold generates networks with many weakly connected node pairs [38,43]. Though the problem can be solved by fixing the minimum number of neighbors of each node, the choice of the threshold influences the node degree distribution, meaning that nodes' roles in the resulting network are related to the choice of thresholds. As an alternative, we propose an entropy-based network construction method, which has better performance in maintaining nodes' roles (e.g., hubs and leaf nodes) and avoiding isolating nodes.

This paper proposes a neural network-based method to extract node representations, and presents an entropy-based approach to construct networks from noisy node activity data. Inspired by the application of neural networks in natural language processing (NLP) [44–49], we propose generating node sequences from node activity data to simulate sentences in documents. The neural network model can embed node sequences into vectors of identical dimensions, which allow us to study node features and construct networks. The main contributions of the paper are as follows: First, we design a simple and direct data processing scheme to generate random node sequences from $M$ conditions. In our approach, the raw data are not required to follow any specific distribution. Thus, rescaling and normalization are obsolete. In addition, the $M$ conditions are processed separately, meaning that negative impacts from missing data and outliers can be minimized. Second, the node sequences are trained with a three-layer neural network model, which builds on the hypothesis that nodes with similar properties tend to have similar neighbors [49]. As a result, similar nodes have similar values in the trained node vectors. Third, we propose an entropy-based method to extract the corresponding network where selected edges can recover node roles [50,51]. Finally, we demonstrate the validity of the proposed approach experimentally using synthetic and real data.

## II. APPROACH

In this section, we define the context set, node sequence generation, and the entropy-based method for network construction.

In human language, words in similar contexts tend to have similar meanings [44]. That is, words with similar meanings usually show in similar neighborhoods. We can use NLP models to learn node representations if we have node sequences in which nodes with similar measurements are in similar contexts. The measurements of nodes in different conditions represent different properties, similar to words in various topics that may have different meanings. Building on these observations, we design a scalable node sequence generation strategy to process the $M$ conditions separately.

### A. Generate context sets from node activity data

Suppose the $N$ nodes are measured in $M$ conditions. Given a node $v_i$ ($i \leqslant N$), we assume its value in the $\omega$th condition is $v_i(\omega)$. We define the context set of node $v_i$ in the $\omega$th condition as

$$C_\omega(v_i) = \left\{ v_j : |v_j(\omega) - v_i(\omega)| \leqslant \delta_i^\omega \right\}, \qquad (1)$$

where the tolerance $\delta_i^\omega$ can be a parameter such that $\delta_i^\omega = \beta_\omega v_i(\omega)$. By employing the parameter $\beta_\omega$, we can tune the size of the context set per the error levels of different technologies. In this work, we skip the generation of context set $C_\omega(v_i)$ when a missing value is present in the $\omega$th condition for node $v_i$, and we do not predict the missing values from other conditions.

Formally, the context set of node $v_i$ is composed of nodes with measurements falling in the range $[v_i(\omega) - \delta_i^\omega, \ v_i(\omega) + \delta_i^\omega]$. Therefore, the number of elements of the intersection set $C_\omega(v_i) \cap C_\omega(v_k)$ is related to the measurements of the two nodes $v_i$ and $v_k$. For example, assume the measurements of the three nodes $v_x$, $v_y$, and $v_z$ in the $\omega$th condition are, respectively, $v_x(\omega) = 1000$, $v_y(\omega) = 990$, and $v_z(\omega) = 950$. It is clear that $v_y(\omega)$ is closer to $v_x(\omega)$ than $v_z(\omega)$, i.e., $|v_x(\omega) - v_y(\omega)| < |v_x(\omega) - v_z(\omega)|$. Therefore, we have the following inequality:

$$|C_\omega(v_x) \cap C_\omega(v_y)| \geqslant |C_\omega(v_x) \cap C_\omega(v_z)|, \qquad (2)$$

where $|\cdot|$ denotes the cardinality of the intersection set. The context set $C_\omega(v_y)$ recapitulates more elements of $C_\omega(v_x)$ than the context set $C_\omega(v_z)$. For any node $v_j \in C_\omega(v_x)$, we have the probability

$$P[v_j \in C_\omega(v_y)] \geqslant P[v_j \in C_\omega(v_z)]. \qquad (3)$$

Furthermore, we assume $G_\omega(v_i)$ is a set consisting of nodes whose context set contains node $v_i$, such that

$$G_\omega(v_i) = \{v_j : v_i \in C_\omega(v_j)\}. \qquad (4)$$

Based on the same example above, we can say that there are more context sets containing simultaneously $v_x$ and $v_y$ than containing simultaneously $v_x$ and $v_z$. Therefore, we have

$$|G_\omega(v_x) \cap G_\omega(v_y)| \geqslant |G_\omega(v_x) \cap G_\omega(v_z)|, \qquad (5)$$

meaning that the nodes with closer values are more likely to be present in the same context sets. Similarly, for any node $v_j \in G_\omega(v_x)$, we have the probability

$$P[v_j \in G_\omega(v_y)] \geqslant P[v_j \in G_\omega(v_z)]. \qquad (6)$$

In the generation of node sequences, we always sample the subsequent node from the context set of the current node. For

example, given a node sequence $l$, suppose the $i$th node is $v_x$, i.e., $l_i = v_x$. Then, we have a node sequence

$$\{\cdots, \; l_{i-1} \in G_\omega(v_x), \; l_i = v_x, \; l_{i+1} \in C_\omega(v_x), \; \cdots\}. \quad (7)$$

Based on Eqs. (3) and (6), $l_{i+1}$ tends to be in $C_\omega(v_y)$ with higher probability than $C_\omega(v_z)$, and $l_{i-1}$ is more likely to be in $G_\omega(v_y)$ than in $G_\omega(v_z)$. That is, the context nodes of $v_x$ tend to be the context nodes of $v_y$ rather than $v_z$, since $v_y(\omega)$ is closer to $v_x(\omega)$ than $v_z(\omega)$. Therefore, in the generated node sequences, we can say that nodes with closer values tend to appear in similar contexts.

### B. Generate random node sequences

The simplest way to generate node sequences from context sets would be to randomly sample the next node from the context set of the current node, which is exactly the first order Markov chain [52]. Assume the $i$th node of a node sequence is $l_i$, the next node $l_{i+1} \in C_\omega(l_i)$ is chosen with probability

$$p(l_{i+1} \mid l_i) = \frac{1}{|C_\omega(l_i)|}. \quad (8)$$

Under this assumption, the nodes in the context set $C_\omega(l_i)$ have an equal probability of being chosen as the subsequent node.

Alternatively, we can generate biased random node sequences. Suppose we have just traversed node $l_{i-1}$, and now we reside at node $l_i$. The probability of sampling the next node $l_{i+1}$ is biased by the previous node $l_{i-1}$. Therefore, we introduce a parameter $\rho$, and the unnormalized probability of the next node is

$$p(l_{i+1} \mid l_i, l_{i-1}) = \begin{cases} 1 & \text{if if } l_{i+1} \in C_\omega(l_{i-1}) \cup \{l_{i-1}\}, \\ \rho & \text{else,} \end{cases} \quad (9)$$

where $l_{i+1} \in C_\omega(l_i)$. The sampling strategy is similar to a second-order Markov chain [52], in which the probability of adding the next node is not only influenced by the current node but also the previous node. A low value of $\rho$ boosts the rate of sampling an element from $C_\omega(l_{i-1})$. On the contrary, a high value of $\rho$ controls the probability of exploring a node far from $l_{i-1}$. Higher $\rho$ allows sampling a node in $C_\omega(l_i)$ but not in $C_\omega(l_{i-1})$. If $\rho = 1$, then Eq. (9) is equivalent to Eq. (8).

In the $\omega$th condition, we generate $K$ random node sequences starting from each node. Repeating the process for all the $M$ conditions, we obtain a corpus $T$ containing $KNM - KZ$ node sequences, where $Z$ represents the number of missing values.

The goal of generating random node sequences is to feed the corpus $T$ to a three-layer neural networks to obtain node vectors [47–49,53,54]. Please refer to Appendix A for more information about the neural network model.

### C. Construct network from trained node vectors

After training the neural network model, we obtain $N$ vectors for the $N$ nodes. With the node vectors, we can predict relationships between the nodes, visualize the global structures of the nodes, and construct a corresponding network.

A conventional way to select the edges is by global thresholding the cosine similarities to filter out weak links and obtain a backbone of the underlying network. Globally

thresholding edges (GTE) is widely used in determining gene co-expression networks [36]. However, the drawback of the GTE is that some nodes could be isolated from the network if the threshold is high. Though we can force isolated nodes connected to some other nodes, the degree distribution of the constructed network is still affected by the selection of threshold, meaning that the roles of nodes in the network are sensible to the choice of threshold. To avoid these issues, we propose a Rényi entropy-based method (REM) to extract a network from the trained node vectors [50,55–59].

Once we have the node vectors, we can compute the cosine similarity to quantify the connection strength for each pair of nodes. Here, we define $S^0(v_i)$ as the initial neighbor set of node $v_i$. $S^0(v_i)$ is composed of nodes that are positively similar to $v_i$, i.e., $S^0(v_i) = \{v_j : s(v_i, v_j) > 0\}$, where $s(v_i, v_j)$ is the cosine similarity between $v_i$ and $v_j$. The network constructed from $S^0(v_i)$, $\forall i < N$ is not helpful in real applications because most node pairs are weakly connected.

Inspired by the application of entropy in ecology, we regard the nodes in the set $S^0(v_i)$ are the states of the system $v_i$. Then, we associate each state with a probability, which is computed from the similarity values, such that

$$\bar{s}_i^0(v_j) = \frac{s(v_i, v_j)}{\sum_{v_j \in S^0(v_i)} s(v_i, v_j)}. \quad (10)$$

In information theory, entropy depicts the diversity and randomness of a system [51]. The Rényi entropy for node $v_i$ with order $\alpha$ is

$$H_\alpha^1(v_i) = \frac{1}{1-\alpha} \ln \sum_{v_j \in S^0(v_i)} \left(\bar{s}_i^0(j)\right)^\alpha, \quad (11)$$

where $\alpha > 0$. Note that the Rényi entropy converges to the Shannon entropy in the case $\alpha \to 1$, i.e., $H_\alpha^1(v_i) = \sum_{v_j \in S^0(v_i)} \bar{s}_i^0(v_j) \log \bar{s}_i^0(v_j)$. For any $\alpha$, the entropy $H_\alpha^1(v_i)$ varies from zero to $\ln |S^0(v_i)|$. In the case of a certain event, i.e., $\exists \; v_j \in S^0(v_i)$, where $\bar{s}_i^0(v_j) = 1$ and $H_\alpha(v_i) = 0$. Conversely, the entropy $H_\alpha^1(v_i) = \ln |S^0(v_i)|$ when $\bar{s}_i^0(v_j)$ follows a uniform distribution. The diversity index $D_\alpha^1(v_i)$ is

$$D_\alpha^1(v_i) = \exp(H_\alpha^1(v_i)) = \left[\sum_{v_j \in S^0(v_i)} \left(\bar{s}_i^0(v_j)\right)^\alpha\right]^{1/(1-\alpha)}, \quad (12)$$

which is also known as the Hill numbers [55]. It is unsurprising that $\bar{s}_i^0(v_j)$ is not uniformly distributed, and $H_\alpha^1(v_i) \in [0, \; \ln |S^0(v_i)|]$. In ecology, the diversity index quantifies the abundance of species in a community. The diversity index approaches the total number of species when the species are equally abundant and approaches one if there is a dominant species. In Eq. (12), the order $\alpha$ influences the sensitivity of the diversity index. Increasing $\alpha$ strengthens the weights of the most abundant species. That is, higher $\alpha$ allows us to select the more abundant species, while lower $\alpha$ will detect more species. Therefore, we can use $\alpha$ to control the number of neighbors of node $v_i$.

We pick $\lfloor D_\alpha^1(v_i) \rceil$ nodes that have highest similarities from $S^0(v_i)$ as the effective number of neighbors of node $v_i$. Then, the selected neighbors compose a new neighbor set $S^1(v_i)$. The nodes in $S^1(v_i)$ are more strongly connected to $v_i$ than the

nodes in $S^0(v_i)$. The network constructed from $S^1(v_i)$, $\forall i \leqslant N$ is denser than that constructed from $S^0(v_i)$, $\forall i \leqslant N$. We can repeatedly run Eqs. (10), (11), and (12) to obtain a network with desired edge density. Assume $\lfloor D_\alpha^k(v_i) \rceil$ is the diversity index of $k$th iteration. Then, we have

$$S^k(v_i) = \left\{ v_j \in S^{k-1}(v_i) : \left| \left\{ v_z \in S^{k-1}(v_i) : s(v_i, v_j) \right. \right. \right.$$
$$\left. \left. \left. < s(v_i, v_z) \right\} \right| < \lfloor D_\alpha^k(v_i) \rceil \right\}, \quad (13)$$

where $k \geqslant 1$. In each iteration, $\lfloor D_\alpha^k(v_i) \rceil$ nodes with highest similarity values are selected as the neighbors of $v_i$. Intuitively, the REM can filter out weak links for $v_i$, and the remaining nodes $S^k(v_i)$ are the most *meaningful* neighbors of $v_i$.

In real networks, leaf nodes are those connected to a small number of others, while hubs have many neighbors. Considering the property of entropy [56,59], the size of the resulting neighbor set $S^k(v_i)$ is relatively small if the similarity value distribution of $S^0(v_i)$ is right-skewed. On the contrary, the size of $S^k(v_i)$ is much larger if the similarity value distribution of $S^0(v_i)$ is left-skewed [50,57]. That is, the role of node $v_i$ in the resulting network is related to the similarity value distribution.

## III. RESULTS

The method we have presented falls in the category of unsupervised learning. In this section, we use both synthetic and real data to evaluate the performance of the proposed approach in recovering global and local structures in terms of feature learning and network reconstruction.

### A. Feature learning

#### 1. Synthetic data

In this part, we used two case studies with $N_1 = 5000$ and $N_2 = 5500$ nodes to evaluate the performance of the proposed approach in recovering a global structure. The nodes in the two case studies are measured in six conditions ($M_1$, $M_2$, $\cdots$, $M_6$) and distributed in five communities ($G_1$, $G_2$, $\cdots$, $G_5$). The first case study has five communities of equal size, i.e., each group has 1,000 nodes. The five communities in the second case study have, respectively, $G_1 = 1000$, $G_2 = 1500$, $G_3 = 500$, $G_4 = 750$, and $G_5 = 1750$ nodes (the sizes of the communities are chosen randomly). Note that the sixth condition is a perturbation. In each condition, nodes in the same community are assigned random values from one of the intervals: $A = [1, 100]$, $B = [101, 200]$, $C = [201, 300]$, $D = [301, 400]$, $E = [401, 500]$, and $R = [1, 500]$. In this work, we created four datasets for each case study per the tables in Appendix B. In Table VI (*Data*.1), $G_1$ and $G_2$ are adjacent but not overlapped. In Tables VII (*Data*.2), VIII (*Data*.3), and IX (*Data*.4), nodes from $G_1$ and $G_2$ are assigned with values from two, three, and four same intervals, respectively, as shown in bold fonts. The relative distance between $G_1$ and $G_2$ is expected to decrease with respect to the increase of the number of overlapped intervals.

#### 2. Experimental results

Based on the approach introduced in Sec. II, we generated context sets with a tolerance of $\delta_i^\omega = 0.1 v_i(\omega)$ ($\beta_\omega = 0.1$).
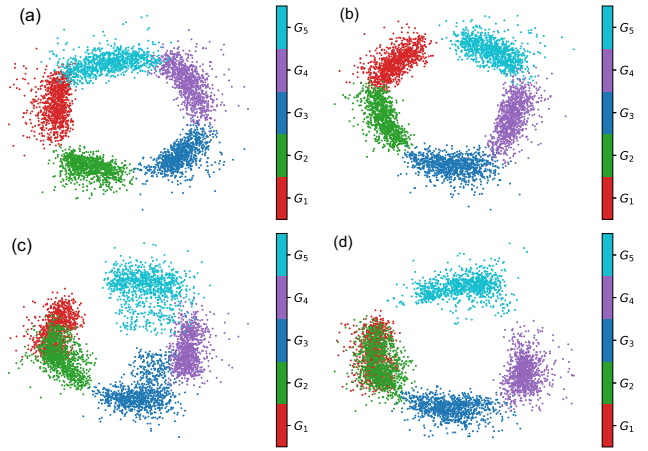


FIG. 1. The node vectors trained from the first case study are visualized via PCA. The five communities have an equal number of nodes. Panels (a–d) are, respectively, the training results of *Data*.1 to *Data*.4.

In the experiments, we generated $K = 10$ random node sequences of length $l = 80$ starting from each node in each of the six conditions. Consequently, the corpus $T_1$ and $T_2$ consist of 300 000 and 330 000 node sequences, respectively. In the neural network, we set the node vector dimension to $d = 128$.

To evaluate the training results qualitatively, we mapped the trained node vectors to a 2D plane via the principle component analysis (PCA) [60,61]. In Figs. 1(a) and 2(a), the nodes from the same communities are mapped to the same areas, meaning that the proposed method can recover the global structure of the dataset. Note that nodes in $G_1$ and $G_2$ are assigned to values from two, three, and four overlapped subintervals from *Data*.2 to *Data*.4 (see the details in Appendix B). That is, the distance between $G_1$ and $G_2$ is assumed to be decreasing for *Data*.2, *Data*.3, and *Data*.4. In Figs. 1(b) and 2(b), we observed the relative distance between $G_1$ and $G_2$ was closer than that in Figs. 1(a) and 2(a). Similarly, the rela-
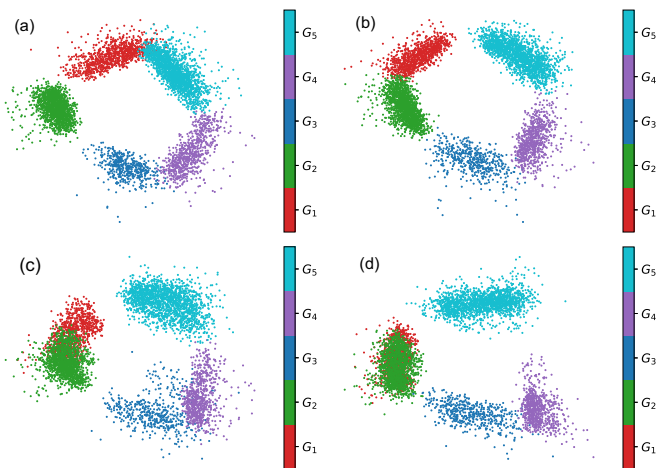


FIG. 2. The node vectors trained from the second case study are visualized via the PCA. The five communities have 1000, 1500, 500, 750, and 1250 nodes. Panels (a–d) are, respectively, the training results of *Data*.1 to *Data*.4.

TABLE I. The relative distance between $G_1$ and $G_2$ and classification accuracy.

| Data | First case study | | Second case study | |
|---|---|---|---|---|
| | Distance | Accuracy(%) | Distance | Accuracy(%) |
| 1 | 2.18 | 99.98 | 2.14 | 100 |
| 2 | 1.82 | 99.96 | 1.86 | 99.98 |
| 3 | 1.52 | 99.96 | 1.49 | 99.90 |
| 4 | 1.18 | 79.84 | 1.16 | 77.92 |

TABLE II. The influence of missing values on the distance between $G_1$ and $G_2$ and classification accuracy.

| Data | 10% missing values | | 20% missing values | |
|---|---|---|---|---|
| | Distance | Accuracy(%) | Distance | Accuracy(%) |
| 1 | 2.19 | 100 | 2.16 | 99.92 |
| 2 | 1.77 | 99.92 | 1.76 | 99.72 |
| 3 | 1.49 | 99.64 | 1.47 | 98.76 |
| 4 | 1.15 | 79.68 | 1.14 | 79.96 |

tive distance between $G_1$ and $G_2$ was even closer in Figs. 1(c) and 2(c), and the two communities were almost merged in Figs. 1(d) and 2(d). The results for the two case studies (eight datasets in total) demonstrated that the node vectors can reflect the relative distances of the node communities, which are affected by the number of overlapped subintervals.

To quantitatively show the results, we computed the distance between $G_1$ and $G_2$. To this end, we calculated the cosine distance (1-cosine similarity) between node pairs. The distance between $G_1$ and $G_2$ was computed as the summation of all possible node pairs between the two communities. For example, the cosine distance between $G_1$ and $G_2$ is

$$Dis(G_1, G_2) = \sum_{v_i \in G_1, v_j \in G_2} 1 - s(v_i, v_j). \quad (14)$$

Then, we calculate the relative distance between $G_1$ and $G_2$ as

$$RelaDis(G_1, G_2) = \frac{Dis(G_1, G_2) * Dis(G_1, G_2)}{Dis(G_1, G_1) * Dis(G_2, G_2)}. \quad (15)$$

Additionally, we perform the simple $K$-means clustering method [62] to classify the trained node vectors into five communities. The classification results are compared to the ground-truth communities.

The relative distances between $G_1$ and $G_2$ and classification results are shown in Table I. We observe that the cosine distance between $G_1$ and $G_2$ is decreasing for $Data.1$ to $Data.4$, in accord with the visualizations in Figs. 1 and 2. Specifically, the distance is close to one for $Data.4$, which suggests that the two communities almost merged. The classification results also agree with the visualization. The classification accuracy is above 99% for $Data.1$, $Data.2$, and $Data.3$, and the classification accuracy has dropped significantly in $Data.4$ since the two communities are almost overlapped, and the nodes from the two communities are falsely classified. From a global view, the node vectors can recover the mesoscopic structure of the nodes. In the following experiments, we will only use the first case study to conduct further analysis.

To study the influence of missing values, we generate incomplete datasets by randomly removing 10% and 20% of values from each condition. We use the same parameters to train the neural network, and the results are shown in Table II. It can be observed that the relative distances between $G_1$ and $G_2$ and the classification accuracies are not significantly affected by the missing values. In Fig. 3, the visualization shows that the global structure of the nodes can still be recovered even when 20% data have been removed randomly. Therefore, the results suggest the proposed method is robust to missing values.

The training results are robust to the choice of training parameters. In the generation of node sequences, we assigned different values to $\beta_\omega$ to control the size of the context set $\delta_\omega^i$. The node sequences are trained using the neural network model with $\rho = 1$. Similarly, we computed the relative distances between $G_1$ and $G_2$ and the classification accuracy. Table III shows that the relative distances are at the same levels for the same datasets, and the classification accuracies are not significantly affected by $\beta_\omega$, meaning that the global structure is still maintained. Thus, the choice of $\beta_\omega$ has limited influence on the embedded node vectors.

To compare the proposed approach with the widely used correlation approach [36,39], we generated four networks from the synthetic data and trained the networks with semi-supervised learning algorithms to obtain node vectors. First, the values of each condition are normalized with the $z$ score,

$$\overline{v}_i(\omega) = \frac{v_i(\omega) - \mu_\omega}{\sigma_\omega}, \quad (16)$$

where $\mu_\omega$ is the mean of all the values in the $\omega$th condition, $\sigma_\omega$ is the standard deviation, and $\overline{v}_i(\omega)$ is the normalized expression value.

The Pearson correlation coefficient (PCC) of any two nodes is

$$r_{(v_x, v_y)} = \frac{\sum_{i=1}^{M}(\overline{v}_x(\omega) - \overline{v}_x)(\overline{v}_y(\omega) - \overline{v}_y)}{\sqrt{\sum_{i=1}^{M}(\overline{v}_x(\omega) - \overline{v}_x)^2}\sqrt{\sum_{i=1}^{M}(\overline{v}_y(\omega) - \overline{v}_y)^2}}, \quad (17)$$
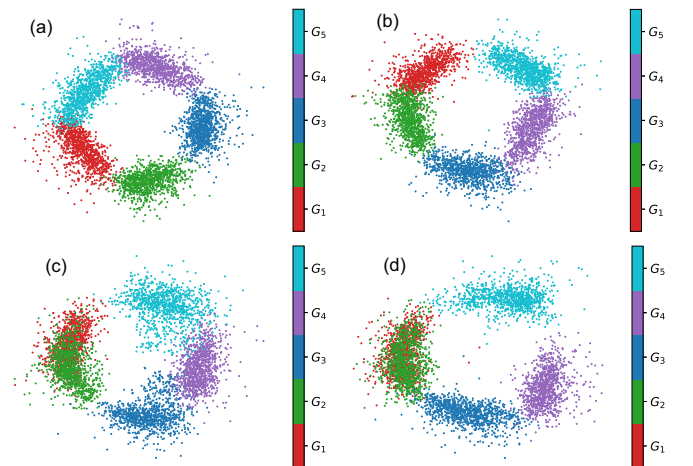


FIG. 3. Visualization of node vectors trained from the first example with 20% missing values. Panels (a–d) are, respectively, the training results of $Data.1$ to $Data.4$.

TABLE III. The relative distance between $G_1$ and $G_2$ and classification accuracy w.r.t. the variation of $\beta_\omega$.

| | Distance | | | | Accuracy(%) | | | |
|---|---|---|---|---|---|---|---|---|
| Data | 0.05 | 0.1 | 0.15 | 0.2 | 0.05 | 0.1 | 0.15 | 0.2 |
| 1 | 2.08 | 2.18 | 2.22 | 2.16 | 100 | 99.98 | 99.86 | 99.86 |
| 2 | 1.78 | 1.82 | 1.82 | 1.76 | 99.98 | 99.96 | 99.86 | 99.88 |
| 3 | 1.49 | 1.52 | 1.48 | 1.47 | 99.92 | 99.96 | 99.06 | 99.84 |
| 4 | 1.15 | 1.18 | 1.16 | 1.14 | 80.72 | 79.84 | 80.86 | 81.34 |

where $r_{(v_x,v_y)}$ is the PCC between node $v_x$ and $v_y$, and $\overline{v}_x$ is the mean of node $v_x$ across the $M$ conditions. The PCC measures how much the two genes are related [36,38]. In this experiment, we did not consider the missing value problem, which could substantially influence the correlation coefficients according to the results in Ref. [39]. The edges are selected by thresholding correlation coefficients, such that PCC $\geqslant 0.95$ [38,63]. All four networks have edge densities above 5%, as shown in Table IV.

To study the properties of nodes, different methodologies are used to determine node vectors from network structure [44–46,64]. Here, we used the node2vec method introduced in Ref. [44] to obtain node vectors from the constructed networks since the approach has shown outstanding performance in reconstructing networks. Similarly, we computed the relative distances between $G_1$ and $G_2$ from the trained node vectors, and the results are shown in Table IV. We observed that the relative distances between the two communities for the first three networks are much higher than in our method (Table I). In the synthetic datasets, $G_1$ and $G_2$ are assumed to partially overlap. However, this characteristic is not recovered from trained node vectors per the visualization of Fig. 4. In the PCC method, errors could be introduced in data normalization, network construction, and feature learning, which consequently influence the accuracy of trained node vectors. As a comparison, our proposed approach trains the node vectors directly from the raw data.

More experimental results on the choice of $\rho$ can be found in Appendix C.

### 3. Real data

We used two real *Anopheles gambiae* gene expression datasets [20,41] to show that the learned node vectors can capture the local structure of the nodes. The first dataset consists of 10,433 *Anopheles gambiae* genes measured in time series after desiccation stress (five conditions) [65]. The five measurements (conditions) of each gene are almost at the

TABLE IV. The relative distance between $G_1$ and $G_2$ and the classification accuracy of the PCC networks.

| Data | Edge density | Distance | Accuracy(%) |
|---|---|---|---|
| 1 | 6.30% | 6.13 | 100 |
| 2 | 5.81% | 3.64 | 81.30 |
| 3 | 5.21% | 2.66 | 80.82 |
| 4 | 5.73% | 1.17 | 64.26 |
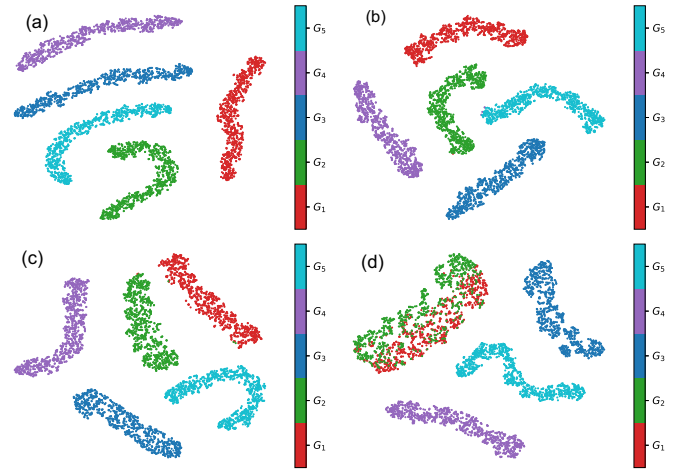


FIG. 4. The visualization of node vectors trained from the Pearson correlation network. Panels (a–d) are, respectively, *Data*.1 to *Data*.4.

same level, and the distributions of the coefficient of variation (CV) and means of the 10 433 genes are shown in Figs. 5(a) and 5(c). The second dataset measures the gene expression values after mating [66], consisting of four measurements (also in time series). The distributions of the CV and means are shown in Figs. 5(b) and 5(d).

In Figs. 5(a) and 5(b), we observe that the CVs of most genes are at a low level. Therefore, we can set the tolerance $\delta_i^\omega$ as the average CV of all the nodes, such that

$$\overline{\text{CV}} = \frac{1}{N} \sum_i \frac{\sigma_i}{m_i}, \tag{18}$$

where $m_i$ is the mean value of gene $v_i$, $\sigma_i$ is the standard deviation, and $\frac{\sigma_i}{m_i}$ is the CV of gene $v_i$. The $\overline{\text{CV}}$s of the two data sets are, respectively, 0.086 and 0.12. Therefore, we set $\beta_\omega = \overline{\text{CV}}$.
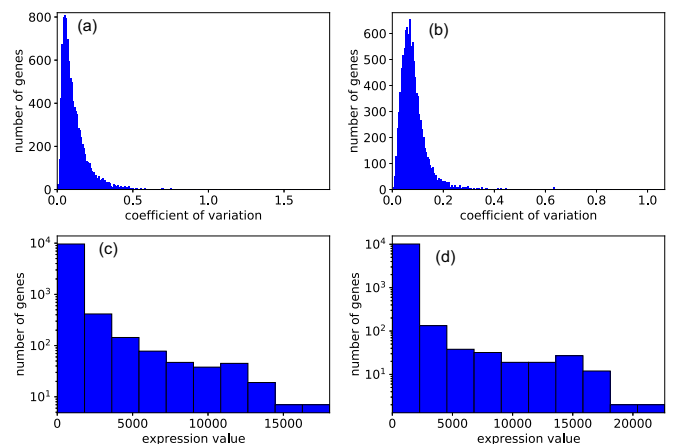


FIG. 5. The properties of the two real datasets. Panels (a) (first dataset) and (b) (second dataset) are the distributions of the CV of the expression values. Panels (c) (first dataset) and (d) (second dataset) are the distributions of means of the expression values.
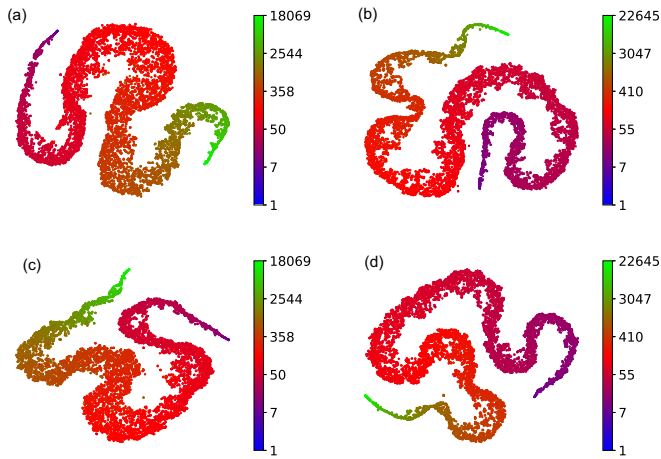
FIG. 6. Visualization of the gene vectors via t-SNE, the genes are colored by the expression values. Panels (a) (first dataset) and (b) are the visualizations of node vectors trained from raw data. Panels (c) (first dataset) and (d) (second dataset) are the visualizations of node vectors trained from the incomplete datasets with 20% values randomly removed from each condition.

### 4. *Experimental results*

The trained node vectors are visualized via the t-distributed stochastic neighbor embedding (t-SNE) method [67] in Fig. 6. The t-SNE constructs probability distribution over pairs of vectors and does not retain the distances of node pairs, but their probabilities. Therefore, the t-SNE approach has better performance in preserving local structure. In Fig. 6, we can observe that the genes with similar expression values are mapped closer, even when 20% of values have been removed from each condition.

As a comparison, we construct a PCC network for the first real data. The raw expression values are rescaled with log2 [20,38] and normalized per Eq. (16) [the distribution of the raw data is heterogeneous as shown in Fig. 5(c)]. Then, a PCC network is constructed by thresholding the edges with a threshold PCC $\geqslant 0.95$ (the network is not sparse). The resulting network consists of 756 330 edges. Similarly, the node
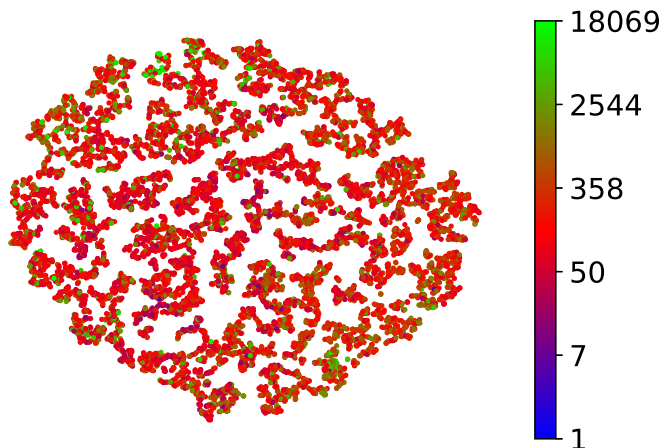


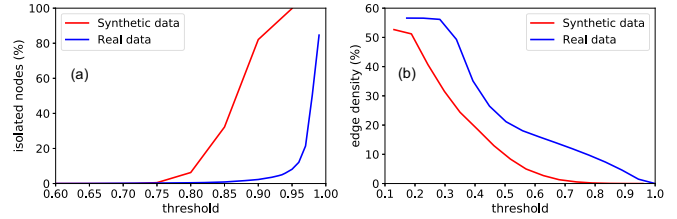FIG. 7. The visualization of node vectors trained from the PCC network.



FIG. 8. Experimental results when different thresholds are applied. (a) shows the percentages of isolated nodes w.r.t. the thresholds. (b) shows the edge densities of networks when different thresholds are used.

vectors are obtained by training the node2vec model. In Fig. 7, we observed that nodes are distributed randomly in the 2D plane, suggesting that nodes with close values are not mapped to the same area. For example, the expression values of the two genes AGAP004677 and AGAP012093 are, respectively, [2764, 2869, 3276, 3690, 3671] and [129, 149, 184, 221, 265], and it is apparent that the expression values of the two genes are at different levels. However, the PCC between the two genes is 0.983, suggesting the two nodes are highly related. The reason is that the PCC method does not depend on the scale of expression values but detects the linear dependence of two genes. In contrast, our approach assumes that similar nodes have more shared elements in their context sets.

### B. Results of network extraction

Thresholding similarity value is the most straightforward and widely used approach in network construction. However, some nodes could be isolated from the network since these nodes may have relatively low similarities to all other nodes. In Fig. 8, we applied different thresholds to the cosine similarities computed from the node vectors trained with the synthetic data [Fig. 1(a)] and the real data [Fig. 5(a)]. We observed that the percentage of isolated nodes increases rapidly when the thresholds are greater than 0.8 (synthetic data) and 0.95 (real data), respectively. In this paper, we define such threshold as the critical value. If we use a threshold smaller than the critical value, then most nodes are connected to at least one other node. On the contrary, if the threshold is larger than the critical value, then we possibly obtain a network with a large percentage of singleton nodes.

We can force isolated nodes to connect with highly similar nodes in real applications. However, the neighbors selected through a single threshold are not affected by the distribution of similarity values. It is not rare that hubs are connected to
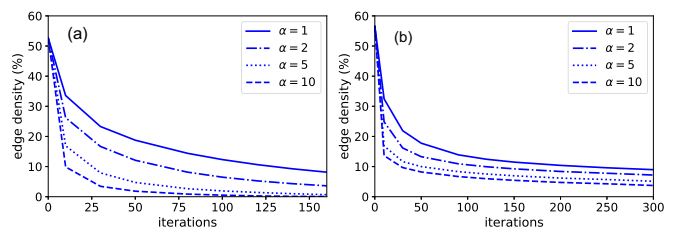


FIG. 9. Edge density analysis of the REM. (a) Synthetic data. (b) Real data.
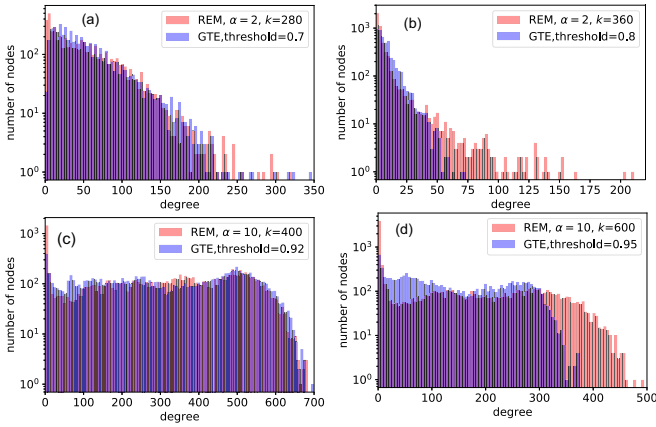
FIG. 10. The degree distributions of networks generated from the GTE and REM. Panels (a) and (b) are the degree distributions of the networks constructed from synthetic data. Panels (c) and (d) are the degree distributions of the networks constructed from real data. Panels (a) and (c) show the results of densely connected networks, while panels (b) and (d) are the results of sparsely connected networks.

many other nodes but with relatively low similarity values, while leaf nodes may connect to a small number of nodes with high similarities. That is, the distribution of similarity values is not considered in the selection of edges.

The proposed REM will maintain every node connected to at least one other node since the "threshold" of each node is determined via the distribution of similarity values per Eq. (12). In the experiments, we applied the REM to the two datasets used in Fig. 8, and the results are shown in Fig. 9. We observed that edge density decreases drastically in the first several iterations, and then the edge density decreases gradually. The reason is that the weakly connected edges are removed immediately from the network in the first several cycles. In contrast, the remaining edges have relatively high similarities, which are removed at a slower speed. In addition, the parameter $\alpha$ allows us to control the removal speed and edge density. Higher $\alpha$ removes weak links more efficiently, which aligns with our analysis in Sec. II C. In real applications, we can fix $\alpha$ and update Eq. (10) to Eq. (10) iteratively until we obtain a network with desired edge density.

Furthermore, we generated four GTE-based networks with different thresholds for the datasets used in Fig. 8. The properties of the networks are shown in Table V. Specifically, The
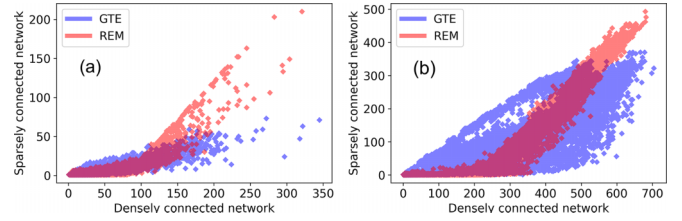


FIG. 11. The comparison of node degrees between densely and sparsely connected networks.

GTE networks are respectively generated with thresholds less and equal to the critical thresholds. In addition, we generated four REM networks, which have similar edge densities to their GTE counterparts. In Table V, we found that both GTE and REM return networks with similar average degrees when the edge densities are the same. However, the GTE networks always have a higher average clustering coefficient, suggesting that nodes in the GTE networks are more likely to cluster together. In Fig. 10, we compared the degree distributions of the eight networks. We observed that the degree distributions of the GTE and REM networks almost overlap when the thresholds are less than the critical values [panels (a) and (c)]. When the thresholds are at the critical values [panels (b) and (d)], some nodes in the REM networks still have high degrees, which are similar to the *hubs* in many real networks. Besides, we observed that all four REM networks have many low-degree nodes, which account for the lower average clustering coefficients in Table V.

Finally, we compared how edge density affects the roles of nodes. In Fig. 11, each point represents a node. The horizontal coordinate represents the node's degree in the densely connected network, while the vertical coordinate represents the node's degree in the sparse network. We observed that the node degree of the GTE networks is remarkably affected by the threshold selection. The highest node degrees have dropped from 345 to 73 for the synthetic data and from 704 to 370 for the real data. In the REM network, the highest node degrees have dropped from 321 to 210 for the synthetic data and 682 to 493 for the real data. In Fig. 11, the REM approach is more likely to remove edges from low-degree nodes. Edges from high-degree nodes are removed proportionally, which means the nodes' roles are maintained and not significantly influenced by edge densities. However, the nodes' degree in the resulting GTE networks is strongly related to the choice

TABLE V. The properties of networks constructed with the GTE and REM.

| Property | < CT[a] (Syn.[b]) | | = CT (Syn.) | | < CT (Real[c]) | | = CT (Real) | |
|---|---|---|---|---|---|---|---|---|
| | GTE | REM | GTE | REM | GTE | REM | GTE | REM |
| Threshold | 0.7 | — | 0.8 | — | 0.92 | — | 0.95 | — |
| Isolated nodes | 0 | 0 | 287 | 0 | 372 | 0 | 849 | 0 |
| Edge density | 1.03% | 0.972% | 0.164% | 0.160% | 2.88% | 2.81% | 1.18% | 1.18% |
| Ave. degree | 51.5 | 48.6 | 8.2 | 8 | 300.8 | 293.2 | 123.2 | 123.4 |
| Ave. clustering | 0.46 | 0.41 | 0.38 | 0.32 | 0.66 | 0.55 | 0.59 | 0.37 |

[a]CT denotes Critical threshold.
[b]Syn. denotes the synthetic data used in Figs. 8 and 9.)
[c]Real denotes the real data used in Figs. 8 and 9.

of edge density. In Fig. 11(b), we can see that the points of the GTE networks are over-dispersed in the diagram.

More experimental results on real data are discussed in Appendix D.

## IV. CONCLUSION AND FUTURE WORKS

This paper presents a neural network-based approach for learning node vectors from noisy node activity data. The primary advantage of the proposed method is that data are not required to follow any specific distribution since we generate *context sets* from raw data for each condition. The proposed approach is not constrained by missing values that ubiquitously exist in experimental results. Inspired by the application of neural networks in natural language processing, we generate a corpus of node sequences to simulate sentences in documents. The corpus is trained by a neural network model, which produces node vectors and allows comparing and identifying nodes with synergistic roles. The experimental results show that the proposed approach is robust to the choice of parameters and missing values. In addition, we offer an alternative method to select edges for the underlying network. The REM method is based on the Rényi entropy and selects edges according to the distribution of similarity values. The proposed approach constructs networks without isolating nodes and can recover the roles of nodes.

In this work, we designed two experiments to test the proposed method. With both synthetic and real data, we showed that the proposed method could unveil the global and local structure of the nodal data even when 20% values are randomly removed from the datasets. Furthermore, we tested the proposed entropy-based network extraction method. We can obtain a network with desired edge density without isolated nodes by controlling the parameter $\alpha$ and the number of iterations.

The experiments in this paper show promising results in detecting global and local structures from noisy nodal data. We expect the proposed data processing methodology to be used in different areas, including biology and finance, especially where node activity data are measured with different techniques and missing values are present.

The code used for this work is available via a GitHub repository [68].

## ACKNOWLEDGMENTS

## APPENDIX A: THE SKIP-GRAM MODEL

The goal of generating random node sequences is to feed the corpus $T$ to neural networks to train node vectors. In this work, we adopt the simple three-layer skip-gram model as shown in Fig. 12. This neural network framework has three layers; input, hidden, and output layer [47–49,53,54]. In this work, the goal is to find the $d$-dimensional vector for each of the $N$ nodes.
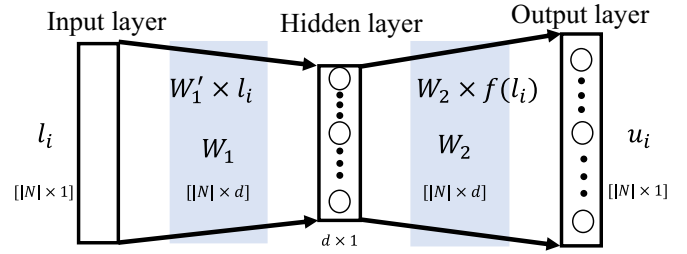


FIG. 12. The three-layer neural network model. Each input node is associated with an $N$-dimensional one-hot vector [47], which is mapped to the node vector $f(v_i)$ (the hidden layer) of dimension $d$ by matrix $W_1$. The hidden layer is mapped to the output vector by matrix $W_2$. The elements of $W_1$ and $W_2$ are initialized with random values, which are expected to be optimized by backpropagation [48].

In our assumption, nodes with similar values tend to appear in a similar context. Given a neighborhood $H$ consisting of $2c$ nodes, we denote $P(v_x \mid H)$ as the conditional probability of node $v_x$ is neighboring to the $2c$ nodes in $H$. Based on Bayes' theorem, we have

$$P(v_x \mid H) = \frac{P(H \mid v_x)P(v_x)}{P(H)}, \quad (A1)$$

where $P(H)$ and $P(v_x)$ are, respectively, the probability of $H$ and $v_x$, and $P(H)$ and $P(v_x)$ can be regarded as constants. Then, we have

$$P(v_x \mid H) \propto P(H \mid v_x). \quad (A2)$$

Now, we take one of the node sequences from the corpus. Let $l_i$ denote the $i$th node of the sequence, and $H = \{l_{i-c}, \ldots, l_{i-1}, l_{i+1}, \ldots, l_{i+c}\}$. That is, we have an outcome $H$ given $l_i$. Since the goal is to determine $f(l_i)$, we replace $v_x$ in Eq. (A2) with $f(l_i)$, and assume the $2c$ nodes are independent [47]. We have

$$P(f(l_i) \mid H) \propto P[H \mid f(l_i)] = \prod_{-c \leqslant j \leqslant c, j \neq 0} P[l_{i+j} \mid f(l_i)], \quad (A3)$$

where $P[l_{i+j} \mid f(l_i)]$ is the occurring probability of node $l_{i+j}$ given the vector $f(l_i)$. To determine $f(l_i)$, we have the optimization problem after taking the log form of Eq. (A3):

$$E = -\min_{f} \sum_{-c \leqslant j \leqslant c, j \neq 0} \log P[l_{i+j} \mid f(l_i)]. \quad (A4)$$

In the model, the node vector $f(l_i)$ is projected to an $N$-dimensional output vector $u_i$ as shown in Fig. 12. The $N$ dimensions of $u_i$ are associated to the $N$ nodes in the corpus. Then, we use the softmax function [48,49,69] to map the entries of $u_i$ into probabilities, which all together give a probability distribution. For example, the probability of the $r$th entry of $u_i$ given $f(l_i)$ is

$$P[v_r \mid f(l_i)] = \frac{\exp\left(u_i^r\right)}{\sum_{r' \in N} \exp\left(u_i^{r'}\right)}, \quad (A5)$$

where $u_i^r$ is $r$th entry of $u_i$ and $P[v_r \mid f(l_i)]$ is the probability of node $v_r$ to be the context of $l_i$. According to Eq. (A5), nodes in $H$ have higher probabilities to be the context of

TABLE VI. The synthetic dataset 1.

| Group | Conditions | | | | | |
| | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ |
|---|---|---|---|---|---|---|
| $G_1$ | A | B | C | D | E | R |
| $G_2$ | B | C | D | E | A | R |
| $G_3$ | C | D | E | A | B | R |
| $G_4$ | D | E | A | B | C | R |
| $G_5$ | E | A | B | C | D | R |

TABLE VIII. The synthetic dataset 3.

| Group | Conditions | | | | | |
| | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ |
|---|---|---|---|---|---|---|
| $G_1$ | **A** | **B** | **C** | D | E | R |
| $G_2$ | **A** | **B** | **C** | E | D | R |
| $G_3$ | B | C | D | A | C | R |
| $G_4$ | C | D | E | B | A | R |
| $G_5$ | D | E | B | C | B | R |

node $l_i$. Combining Eqs. (A4) and (A5), we have the loss function [70–72]

$$E = -\min_f \sum_{-c \leqslant j \leqslant c, j \neq 0} u_{i+j} + \log \left[ \sum_{r' \in N} \exp\left(u_i^{r'}\right) \right], \quad \text{(A6)}$$

which is applied to every node in the sequence. Equation (A6) is optimized by using the stochastic gradient descent approach [44,48,49], which backpropagates [69] errors to update the elements of the matrices $W_1$ and $W_2$ in Fig. 2.

The method we have introduced falls in the category of unsupervised learning, in which we learn node vectors from nodal data. The node vectors can be used to extract networks or detect nodes with similar properties.

## APPENDIX B: SYNTHETIC DATASETS

The four synthetic datasets for the two case studies are generated according to Tables VI, VII, VIII, and IX.

## APPENDIX C: PARAMETER CHOICE

We study the influence of $\rho$ on the relative distance between $G_1$ and $G_2$, and the results are shown in Tables X and XI. We observe that the distance between the two communities slightly increases when we employ a low value of $\rho$ since a small $\rho$ encourages adding nodes that also exist in the context set of the previous node. As a result, far away nodes will become closer, reflected in the reduced distance between the two communities. However, the results are not significantly influenced by $\rho$ since the relative distances of two communities are maintained at the same level for the same data. Therefore, we recommend using $\rho = 1$ in most cases.

## APPENDIX D: STUDY THE REM APPROACH WITH AUC METRICS ON REAL DATA

In this part, we use two real datasets with both network structure and node activity data to study the proposed approach. It is often hard to quantitatively determine the relationships between the network structure and node activity data because they describe the properties of nodes from different aspects. In the experiments, we learn node vectors from the node activity data, compute similarity and construct networks. The constructed network is compared to the network structures, and we use the AUC to evaluate our REM approach.

### 1. The cora dataset

The cora dataset [73] contains a sparse citation network with 2708 nodes and 5278 edges (the edge density is 0.144%), where nodes represent publications and edges represent the citation relationships between the papers. Each node in the network is described by a 0/1-valued word vector, indicating the absence/presence of the corresponding word from a dictionary. The dictionary consists of 1433 unique words presented at least ten times in one of the 2708 publications.

### 2. The pubmed dataset

The pubmed dataset [74] contains a sparse citation network with 19717 nodes and 44324 edges (the edge density is 0.0228%), where nodes represent publications and edges represent the citation relationships between the papers. Each node in the network is described by a TF/IDF weighted word vector from a dictionary consisting of 500 words.

The two networks represent citation relationships between the publications (nodes), while the node activity data are extracted from the content of each publication. We implement the proposed approach on these two real

TABLE VII. The synthetic dataset 2.

| Group | Conditions | | | | | |
| | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ |
|---|---|---|---|---|---|---|
| $G_1$ | **A** | **B** | C | D | E | R |
| $G_2$ | **A** | **B** | D | E | D | R |
| $G_3$ | B | C | E | A | C | R |
| $G_4$ | C | D | A | B | B | R |
| $G_5$ | D | E | B | C | A | R |

TABLE IX. The synthetic dataset 4.

| Group | Conditions | | | | | |
| | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ |
|---|---|---|---|---|---|---|
| $G_1$ | **A** | **B** | **C** | **D** | E | R |
| $G_2$ | **A** | **B** | **C** | **D** | D | R |
| $G_3$ | B | C | D | E | C | R |
| $G_4$ | C | D | E | A | B | R |
| $G_5$ | D | E | B | C | A | R |

TABLE X. The relative distance between $G_1$ and $G_2$ w.r.t. $\rho$.

| Data | 1/10 | 1/5 | 1/3 | 3 | 5 | 10 |
|------|------|------|------|------|------|------|
| 1 | 2.23 | 2.22 | 2.19 | 2.18 | 2.17 | 2.07 |
| 2 | 1.84 | 1.83 | 1.82 | 1.80 | 1.78 | 1.67 |
| 3 | 1.53 | 1.52 | 1.52 | 1.48 | 1.42 | 1.32 |
| 4 | 1.22 | 1.21 | 1.20 | 1.17 | 1.16 | 1.13 |

TABLE XI. The prediction accuracy w.r.t. $\rho$.

| Data | 1/10 | 1/5 | 1/3 | 3 | 5 | 10 |
|------|------|------|------|------|------|------|
| 1 | 98.24 | 98.24 | 98.72 | 98.80 | 98.90 | 98.82 |
| 2 | 98.26 | 97.22 | 98.68 | 98.00 | 97.30 | 98.12 |
| 3 | 97.52 | 97.48 | 97.72 | 98.22 | 97.58 | 97.32 |
| 4 | 78.14 | 78.58 | 78.62 | 79.72 | 79.32 | 79.78 |

datasets to generate node vectors (128 dimensions). Then, we calculate the similarity for every node pair, and the performance of the approach is evaluated by comparing it to the true citation networks. The AUCs of the two datasets are, respectively, 0.81 and 0.73. Though the true relationship between network structure and node activity data is unknown, the results reveal that the node activity data are related to the network structure. Therefore, one of the advantages of our approach is that it allows us to compare two different types of data.

[1] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, Rev. Mod. Phys. **87**, 925 (2015).

[2] D. M. Gysi, Í. Do Valle, M. Zitnik, A. Ameli, X. Gan, O. Varol, S. D. Ghiassian, J. J. Patten, R. A. Davey, J. Loscalzo, and A. L. Barabási, Proc. Natl. Acad. Sci. USA **118**, 19 (2021).

[3] C. De Bacco, E. A. Power, D. B. Larremore, and C. Moore, Phys. Rev. E **95**, 042317 (2017).

[4] J. Kuang and C. Scoglio, Phys. Rev. E **104**, 024301 (2021).

[5] J. Young, G. T. Cantwell, and M. E. J. Newman, J. Complex Networks **8**, cnaa046 (2021).

[6] M. E. J. Newman, Phys. Rev. E **98**, 062321 (2018).

[7] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. E **84**, 066106 (2011).

[8] R. Guimerá and M. Sales-Pardo, Proc. Natl. Acad. Sci. USA **106**, 22073 (2009).

[9] T. P. Peixoto, Phys. Rev. E **97**, 012306 (2018).

[10] T. P. Peixoto, Phys. Rev. X **8**, 041011 (2018).

[11] B. Prasse and P. Van Mieghem, arXiv:1807.08630

[12] L. Song, P. Langfelder, and S. Horvath, BMC Bioinf. **14**, 5 (2013).

[13] B. Karrer and M. E. J. Newman, Phys. Rev. E **83**, 016107 (2011)

[14] T. P. Peixoto, Phys. Rev. Lett. **123**, 128301 (2019).

[15] M. Timme, Phys. Rev. Lett. **98**, 224101 (2007).

[16] S. G. Shandilya and M. Timme, New J. Phys. **13**, 013004 (2011).

[17] P. Van Mieghem and Q. Liu, Phys. Rev. E **100**, 022317 (2019).

[18] S. Boccaletti, G. Bianconi, R. Criado, CI. del Genio, J. Gómez-Gardeñes, and M. Romance, Phys. Rep. **544**, 1 (2014).

[19] M. E. J. Newman, Nat. Phys. **14**, 542 (2018).

[20] R. M. MacCallum, S. N. Redmond, and G. K. Christophides, BMC Genomics **12**, 620 (2011).

[21] S. Horvath and J. Dong, PLoS Comput. Biol. **4**, e1000117 (2008).

[22] M. E. Lynall, D. S. Bassett, R. Kerwin, P. J. McKenna, M. Kitzbichler, U. Muller, and E. Bullmore, J. Neurosci. **30**, 9477 (2010).

[23] S. Raimondo and M. De Domenico, Phys. Rev. E **103**, 022311 (2021).

[24] G. Sugihara, R. May, H. Ye, C. H. Hsieh, E. Deyle, M. Fogarty, and S. Munch, Science **338**, 496 (2012).

[25] E. Bullmore and O. Sporns, Nat. Rev. Neurosci. **10**, 186 (2009).

[26] B. Benigni, A. Ghavasieh, A. Corso, V. d'Andrea, and M. De Domenico, Network Neurosci. **5**, 831 (2021).

[27] J. Schiefer, A. Niederbühl, V. Pernice, C. Lennartz, J. Hennig, P. LeVan, and S. Rotter, PLoS Comput. Biol. **14**, e1006056 (2018).

[28] J. Jeong, J. C. Gore, and B. S. Peterson, Clin. Neurophysiol. **112**, 827 (2001).

[29] A. Namaki, A. H. Shirazi, R. Raei, R. and G. R. Jafari, Physica A **390**, 3835 (2011).

[30] K. Yamasaki, A. Gozolchiani, and S. Havlin, Phys. Rev. Lett. **100**, 228501 (2008).

[31] J. F. Donges, Y. Zou, N. Marwan, and J. Kurths, Europhys. Lett. **87**, 48007 (2009).

[32] J. G. Young, A. Kirkley, and M. E. J. Newman, Phys. Rev. E **105**, 014312 (2022).

[33] I. A. Kovács, D. L. Barabási, and A. L. Barabási, Proc. Natl. Acad. Sci. USA **117**, 33570 (2020).

[34] M. C. Oldham, S. Horvath, and D. H. Geschwind, Proc. Natl. Acad. Sci. USA **103**, 17973 (2006).

[35] B. Zhang and S. Horvath, Stat. Appl. Genet. Mol. Biol. **4**, 1 (2005).

[36] L. Song, P. Langfelder, and S. Horvath, BMC Bioinf. **13**, 168 (2012).

[37] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano, BMC Bioinf. **7**, S7 (2006).

[38] R. Ramirez, Y. C. Chiu, A. Hererra, M. Mostavi, J. Ramirez, Y. Chen, Y. Huang, and Y. F. Jin, Front. Phys. **8**, 203 (2020).

[39] A. Reverter and E. K. Chan, Bioinformatics **24**, 2491 (2008).

[40] S. García, J. Luengo, and F. Herrera, Intell. Syst. Ref. Library **72**, 59 (2015).

[41] J. Kuang, N. Buchon, K. Michel, and C. Scoglio, BMC Bioinf. **23**, 170 (2022).

[42] B. Dresp-Langley, O. K. Ekseth, J. Fesl, S. Gohshi, M. Kurz, and H.-W. Sehring, Appl. Sci. **9**, 3065 (2019).

[43] S. Van Dam, U. Võsa, A. van der Graaf, L. Franke, and J. P. de Magalhães, Brief. Bioinform. **19**, 575 (2018).

[44] A. Grover and J. Leskovec, In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)* (ACM, New York, NY, 2016), p. 855.

[45] B. Perozzi, R. Al-Rfou, and S. Skiena, In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)* (ACM, New York, NY, 2014), p. 701.

[46] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)* (ACM, New York, NY, 2015), p. 1067.

[47] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, arXiv:1802.05365

[48] R. Xin, arXiv:1411.2738

[49] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, arXiv:1310.4546.

[50] M. J. Lee, E. Lee, B. Lee, H. Jeong, D. S. Lee, and S. H. Lee, Phys. Rev. Res. **3**, 043136 (2021).

[51] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora, Nat. Commun. **6**, 6864 (2015).

[52] A. Shamshad, M. A. Bawadi, W. M. A. Wan Hussin, T. A. Majid, and S. A. M. Sanusi, Energy **30**, 693 (2005).

[53] O. Barkan and N. Koenigstein, In *Proceedings of the IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)* (IEEE, Piscataway, NJ, 2016).

[54] M. Grbovic and H. Cheng, In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)* (ACM, New York, NY, 2018).

[55] M. O. Hill, Ecology **54**, 427 (1973).

[56] K. Życzkowski, Open Syst. Inf. Dyn. **10**, 297 (2003).

[57] R. J. Hill, J. Econom. **130**, 25 (2006).

[58] A. Chao, N. J. Gotelli, T. C. Hsieh, E. L. Sander, K. H. Ma, R. K. Colwell, and A. M. Ellison, Ecol. Monogr. **84**, 45 (2014).

[59] L. Jost, Oikos **113**, 363 (2006).

[60] S. Wold, K. Esbensen, and P. Geladi, Chemom. Intell. Lab. Syst. **2**, 37 (1987).

[61] P. Filzmoser, K. Hron, and C. Reimann, J. Int. Environmet. Soc. **20**, 621 (2009).

[62] A. Likas, N. Vlassis, and J. J. Verbeekb, Pattern Recogn. **36**, 451 (2003).

[63] K. Baltakys, J. Kanniainen, and F. Emmert-Streib, Sci. Rep. **8**, 8198 (2018).

[64] B. Ball, B. Karrer, and M. E. J. Newman, Phys. Rev. E **84**, 036103 (2011).

[65] M. H. Wang, O. Marinotti, A. Vardo-Zalik, R. Boparai, and G. Yan, PLoS One **6**, e26011 (2011).

[66] D. W. Rogers, M. M. Whitten, J. Thailayil, J. Soichot, E. A. Levashina, and F. Catteruccia, Proc. Natl. Acad. Sci. USA **105**, 19390 (2008).

[67] L. Van Der Maaten, In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics* (MLResearch Press, 2009), pp. 384–391.

[68] GitHub repository, https://github.com/BigBroKuang/embed-data-to-vector.

[69] Y. LeCun, Y. Bengio, and G. Hinton, Nature (London) **521**, 436 (2015).

[70] D. Wang, AI Mag. **22**, 10 (2001).

[71] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, Ann. Data Sci. **9**, 187 (2022).

[72] J. Ruan, A. K. Dean, and W. Zhang, BMC Syst. Biol. **4**, 8 (2010).

[73] The cora dataset, https://relational.fit.cvut.cz/dataset/CORA.

[74] The pubmed dataset, https://relational.fit.cvut.cz/dataset/PubMed_Diabetes.