Network Structure and Feature Learning from Rich but Noisy Data

Junyao Kuang and Caterina Scoglio

Department of Electrical and Computer Engineering, Kansas State University

(Dated: November 5, 2021)

In the study of network structures, much attention has been devoted to network reconstruction, which relies on partial edge-related information or dynamical processes on the network. However, there are cases where we are only given incomplete nodal data, and the nodal data are measured with different methodologies. In this work, we present an unsupervised learning framework to construct networks from noisy and heterogeneous nodal data. First, we introduce the creating nodes’ context sets, which are used to generate random node sequences. Then, a three-layer neural network is adopted to train the node sequences to infer node vectors, enabling us to capture nodes with synergistic roles within the network. Further, the effectiveness of the method is validated through both synthetic data and real data. Finally, we compare the differences between the global thresholding method and the entropy-based method in edge selection. In summary, this work presents a neural network method for node vector learning from heterogeneous nodal data and an entropy-based method for edge selection.

A network is a system-level view of pairwise interactions between nodes which represent parts of a complex system [1–5]. In the construction of networks, one needs to determine the types of interactions (edges), which can be identified directly in many cases. For instance, in the construction of social networks, one can identify edges by simply asking people about their friends or the frequency they meet their friends [6]. Similarly, edges in web graphs can be directly determined by checking if hyperlinks exist between web pages. However, there are cases that the edges cannot be detected directly, but nodal information can be measured from different aspects [7]. Such problems are ubiquitous in constructing gene regulatory networks, metabolic networks, and protein interaction networks [8]. The expression data of different sets of genes or proteins are acquired from various experiments conducted through different techniques, such as microarray and RNA-sequencing [9,7]. We describe such problems as constructing networks from heterogeneous nodal data [9].

The existing network construction methods solve the problem with diverse approaches. One is to reconstruct networks by applying known network-related data, which is based on stochastic block models [10–16]. Another common approach reconstructs network with maximum a posteriori (MAP) estimation which incorporates similar layers in multilayer networks [17]. These network reconstruction methods are effective, especially when missing links are uniformly distributed among nodes. Further, some researchers used the Bayesian model to reconstruct social networks based on dynamical processes [18–21], or incorporated dynamics on the network such as epidemic spreading [22,23]. However, A common but neglected problem is estimating the hidden interactions between the nodes with only nodal data. Such problems are ubiquitous in constructing biological networks, in which edges are mostly obtained by thresholding correlations between nodes [24,25,27]. Such methods require some pre-processing of the raw data, such as log transformation and z-score normalization [7]. In addition, the missing value problem and heterogeneous data ranges are also daunting challenges for researchers.

This work presents an approach for network construction and node feature learning from rich but noisy data. Inspired by the application of neural networks in natural language processing [29,31], we convert the heterogeneous nodal data into tractable node sequences to simulate sentences in documents. First, we generate context sets for each node, akin to the contexts of words in documents. Second, we introduce a random node sequence generation framework that can circumvent the missing value problem. Third, we employ the simple skip-gram model to embed nodes into vectors [34]. Finally, we propose an entropy-based method to select edges that can ensure nodes with at least one edge [35,30].

Context set.—Suppose the nodes in the set $N$ are measured in $\Omega$ conditions. Thus, for any node $v$, the goal is to find $f : v^{\Omega} \rightarrow \mathbb{R}^d$, where $d$ is the dimension of the node vector. Note that there could be missing values in any conditions for any node.

Given a node $v$, we assume its measurement in condition $\omega$ is $v_\omega$. It is not surprising that there are many other nodes with measurements close to $v$, and here, we say these nodes are node $v$’s contexts. Thus, these nodes compose the set

$$C_\omega(v) = \{v_j : |v_j(\omega) - v_\omega| \leq \delta_\omega\},$$

(1)

where we call the context set of node $v$ in condition $\omega$. $\delta_\omega$ is the tolerance, it can be a variable such as $\delta_\omega = 0.1v_\omega(\omega)$. As an example, Fig. 1 shows the process to generate the context set. Likewise, we can obtain $|N|$ context sets for all the nodes in the $\omega$th condition. Based on our definition of context set in Eq. 1, we can say that the closer the measurements for the nodes $v$ and $v_j$, the more elements in the intersection $C_\omega(v) \cap C_\omega(v_j)$, and vice versa. Similarly, nodes with close measurements are more likely to
present in the same context set simultaneously. In a massive corpus, two words generally have similar properties if they present before and after the same set of words frequently. Analogously, if we generate node sequences in which the subsequent nodes are always sampled from the context set of the current node, then two nodes would have close measurements if they appear before and after the same set of nodes repeatedly.

For a node sequence $v$, $v$ is the measurement context set of the current node, and we generate a sequence is the measurement chosen with probability $v$. The depth searching rate $q$ controls the rate of adding a new node far from $v$, high $q$ enables exploring more nodes, which is akin to the Depth-first Search (DFS). Nodes in $C_w(v)$ but not in $C_w(v-1)$ have higher probability to be explored as the next node in $\omega$. If $q = p = 1$, Eq. 3 is equivalent to Eq. 2. In this strategy, we can balance the explorations of BFS and DFS with the parameters $p$ and $q$.

**Skip-gram model.**—The goal of generating random node sequences is to feed the corpus $T$ to neural networks to obtain node vectors. In this work, we adopt the simple three-layer skip-gram model, which is shown in Fig. 2. This neural network framework has three layers; input, hidden, and output layer.

In a node sequence, suppose we are interested in $2c$ ($c$ is an integer) nodes in the vicinity of $v$, and we expect the $2c$ nodes to be more likely to present in the context of $v$. The problem is then equivalent to the maximum likelihood estimation, i.e., the likelihood $P(\{|v_{i-c}, ..., v_{i-1}, v_{i+1}, ..., v_{i+c}\}) | v_i)$ is expected to be maximized. Assuming the $2c$ nodes are independent, we have

$$ P(\{|v_{i-c}, ..., v_{i-1}, v_{i+1}, ..., v_{i+c}\}) | v_i) = \prod_{-c \leq j < c} P(v_{i+j} | f(v_i)).$$

We have the optimization problem after taking the log form:

$$ E = - \min_f \sum_{-c \leq j \leq c} \log P(v_{i+j} | f(v_i)).$$

In Eq. 5 we need to specify the expression $P(v_{i+j} | f(v_i))$ to update the parameters in the neural network. In the skip-gram model, the node vector $f(v_i)$ is projected to an output vector $u_i$ of $|N|$ dimensions as shown in Fig. 2. In the output vector, each dimension corresponds to a node in the corpus. Apply the softmax function, and we can compress the entries of the output vector to the range $[0, 1]$ and forcing the sum equal to one. We then have the conditional probability

$$ P(v_r | f(v_i)) = \frac{\exp(u_i^r)}{\sum_{r' \in N} \exp(u_i^{r'})}. $$

**FIG. 1.** Steps to generate random node sequences. Note that " × " are missing values that are neglected in context sets generation.

**Random node sequences.**—With context sets generated above, we can generate node sequences to simulate sentences in documents. Assume the length of the node sequence is $l$, for the $i$th node in the sequence, the simplest way to append a new node after $v_i$ is by randomly choosing a node from the context set of $v_i$. The node is chosen with probability

$$ p(v_{i+1} | v_i) = \frac{1}{C_\omega(v_i)}. \quad (2) $$

In the generation of node sequences, each node is taken as the starting node, and we generate $K$ node sequences for each starting node. Thus, we have $T = \sum_{\omega \in T} K(|N| - \epsilon_\omega)$ node sequences, where $\epsilon_\omega$ is the number of missing values in context $\omega$.

Instead of randomly sampling the subsequent node from the context set, we introduce two parameters to bias the probability. Suppose we have just traversed node $v_{i-1}$, and now we reside at node $v_i$. The next node will be chosen from $C_\omega(v_i)$, and we assume it is influenced by the preceding node $v_{i-1}$. We give the unnormalized probability

$$ p(v_{i+1} | v_i) = \begin{cases} 1 & \text{if } v_{i+1} = v_{i-1} \\ p & \text{if } v_{i+1} \in C_\omega(v_{i-1}) \cap C_\omega(v_i) \\ q & \text{if } v_{i+1} \notin C_\omega(v_{i-1}) \cup \{v_{i-1}\}, \quad (3) \end{cases} $$

where we define $p$ as the breadth searching rate and $q$ as the depth searching rate.

The breadth searching rate $p$ controls the probability of traversing nodes in the intersection of $C_\omega(v_{i-1}) \cap C_\omega(v_i)$, a high value boosts the rate of traversing local nodes. Similar to the Breadth-first Search (BFS), nodes in the intersection $C_\omega(v_{i-1}) \cap C_\omega(v_i)$ are more likely to be appended after $v_i$. The depth searching rate $q$ controls the rate of adding a new node far from $v_{i-1}$, high $q$ enables exploring more nodes, which is akin to the Depth-first Search (DFS). Nodes in $C_\omega(v_i)$ but not in $C_\omega(v_{i-1})$ have higher probability to be explored as the next node in $\omega$. If $q = p = 1$, Eq. 3 is equivalent to Eq. 2. In this strategy, we can balance the explorations of BFS and DFS with the parameters $p$ and $q$.

**FIG. 2.** The three-layer neural network model.
where \( v_r \) is any node in the corpus, \( u_r^f \) is the corresponding entry in the output vector \( u_r \). Therefore, Combining Eq. 5 and Eq. 6 we have the loss function

\[
E = -\min_f \sum_{-c \leq j < c, j \neq 0} u_{i+j} + \log(\sum_{r' \in N} \exp(u_{r'})). \tag{7}
\]

Note that \( u_j \) is derived from \( f(v_j) \), and Eq. 7 is applied to all nodes in the corpus. Eq. 7 is optimized by using stochastic gradient descent to backpropgate errors to update parameters in the weighted matrices in Fig. 2.

The method we have introduced falls in the category of unsupervised learning, in which we learn node vectors from non-network data, and thus there is no ground truth network for comparison. Here, we evaluate the method from two aspects. First, we implement the method to synthetic data with community structure to show that nodes within the same communities can be clustered closer. Second, we use heterogeneously distributed real data to show that the nodes with close values are embedded with similar vectors.

**Synthetic data.—** Assume we are given \(|N| = 5,000\) nodes which are divided into five communities, i.e., \( G_1 = [1, 1000], ..., G_5 = [4001, 5000] \). The nodes are measured in six conditions within the range \( R = [1, 500] \), and \( R \) is divided into five adjacent sub-intervals, i.e., \( A = [1, 100], B = [101, 200], ..., E = [401, 500] \). In the first five tests (measurements), nodes in each community are assigned with random values from one specific sub-interval, as shown in Table I. The sixth test is supposed to be a perturbation, in which nodes are assigned with random values from the range \( R \).

**TABLE I. Synthetic network properties**

| Tests | \( T_1 \) | \( T_2 \) | \( T_3 \) | \( T_4 \) | \( T_5 \) | \( T_6 \) |
|-------|----------|----------|----------|----------|----------|----------|
| \( G_1 \) | 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 |

The generation of node sequences is illustrated in Fig. 3(a). In the experiment, we generate ten sequences for each starting node in each test. Thus, the corpus \( T \) has 60\(|N| \) node sequences. The sequences are trained with the above skip-gram model with a hidden layer of 128 dimensions.

In Fig. 3(a), the trained node vectors are projected into a 2D plane via Principal Component Analysis (PCA), which can capture the global structure of the communities [39, 40]. We observe that the nodes from the same communities are clustered together, and communities assigned with adjacent sub-intervals are closer. For example, \( G_2 \) is adjacent to \( G_1 \), since the first four testing ranges of \( G_2 \) are adjacent to that of \( G_1 \).

Further, we compare how the number of overlapped sub-intervals affects the relative locations of the communities. In the first two tests, nodes in communities \( G_1 \) and \( G_2 \) are assigned random values from the same sub-intervals (Fig. 3(b)), and we observe the distance between communities \( G_1 \) and \( G_2 \) is closer than that in Fig. 3(a). Likewise, communities \( G_1 \) and \( G_2 \) are then assigned with random values from three and four same sub-intervals. We see that locations of \( G_1 \) and \( G_2 \) are even closer (Fig. 3(c)) and almost merged (Fig. 3(d)). The results of the four experiments demonstrate that the node vectors can reflect the relative distances of the node communities, which are affected by the number of overlapped sub-intervals.

**Real data.—** Two real *Anopheles gambiae* gene expression datasets are adopted to show that the method can embed genes (nodes) with similar values into similar vectors. The gene expression levels are recorded in time series after desiccation stress (five measurements) and mating activities (four measurements) [7, 41, 42], so each gene has almost constant expression levels across the conditions within the same dataset. We prefer using the t-sne method to visualize the relative distances of the trained node vectors since the distributions of gene expressions are heterogeneous (Fig. 4(a) and (c)) [43].

Given the truth that t-sne has better performance in capturing local information while PCA is suitable for projecting global structure [40]. From Fig. 4(b) and (d), we observe that the genes with similar expression levels are mapped closer, demonstrating our method’s effectiveness in recovering similar nodes.

**Edge selection.—** Assume the cosine similarity between nodes \( v_i \) and \( v_j \) is \( w_{ij} \). Basically, we can consider nodes \( v_i \) and \( v_j \) are connected by an edge as long as \( w_{ij} > 0 \). As a result, we will obtain a densely connected network (DCN) containing many weak and invalid links. 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 DCN. Global thresholding edges (GTE) is widely used in determining gene coexpression
networks [20]. As an alternative, we propose a Rényi entropy-based method (REM) to extract the backbone of such DCNs [35] [44–46], which is applied to individual nodes.

We assume the nodes with positive cosine similarities to \(v_i\) are in the set \(w_p(v_i) = \{v_j : w_{ij} > 0\}\). The normalized weight of \(v_j\) in terms of \(v_i\) is

\[
\overline{w}_i(j) = \frac{w_{ij}}{\sum_{v_j \in w_p(v_i)} w_{ij}}.
\]

(8)

We correspond the normalized weight \(\overline{w}_i(j)\) as the probability of \(v_j\) connected to \(v_i\) among the \(|w_p(v_i)|\) nodes. In information theory, entropy depicts the diversity and randomness of a system [36]. The Rényi entropy for node \(v_i\) with order \(\alpha\) is

\[
H_\alpha(v_i) = \frac{1}{1-\alpha} \ln \sum_{v_j \in w_p(v_i)} \overline{w}_i(j)^\alpha,
\]

(9)

where \(\alpha > 0\). Note that the Rényi entropy converges to the Shannon entropy in the case \(\alpha \to 1\), i.e., \(H_1(v_i) = \sum_{v_j \in w_p(v_i)} \overline{w}_i(j) \ln \overline{w}_i(j)\). For any \(\alpha\), the entropy \(H_\alpha(v_i)\) varies from zero to \(\ln |w_p(v_i)|\). In the case of a certain event, i.e., \(\overline{w}_i(j) = 1\), \(H_\alpha(v_i) = 0\). Conversely, the entropy \(H_\alpha(v_i) = \ln |w_p(v_i)|\) if \(\overline{w}_i(j)\) follows uniform distribution. The diversity index \(D_\alpha(v_i)\) is calculated as

\[
D_\alpha(v_i) = \exp(H_\alpha(v_i)) = \left( \sum_{v_j \in w_p(v_i)} \overline{w}_i(j)^\alpha \right)^{1/(1-\alpha)}.
\]

(10)

which is also known as the Hill numbers [44]. It is unsurprising that \(\overline{w}_i(j)\) is heterogeneous distributed in practice, and \(H_\alpha(v_i) = \ln D_\alpha(v_i) \in [0, \ln |w_p(v_i)|]\). 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 it approaches one if there are one dominant species. In Eq. 10 the order \(\alpha\) influences the computation sensitivity of the diversity index. Increasing \(\alpha\) strengthens the weights of the most abundant species and consequently reduces the diversity index.

We can pick top \(|D_\alpha(v_i)|\) nodes from \(w_p(v_i)\) as the effective number of neighbors of \(v_i\). In practical applications, there could be a substantial number of nodes in \(w_p(v_i)\), top \(|D_\alpha(i)|\) nodes are still a large number. Here, we propose filtering the weakly connected neighbors through multiple iterations. Assume the diversity index of \(k\)th iteration is \(|D_\alpha^{k}(v_i)|\), then, we have

\[
w^k_p(v_i) = \{v_j \in w^{k-1}_p(v_i) : \{|v_z \in w^{k-1}_p(v_i) : w_{ij} < w_{iz}\} < |D_\alpha(v_i)|\},
\]

where \(k \geq 1\) and \(w^0_p(v_i) = w_p(v_i)\). The procedure to update \(D_\alpha^{k}(i)\) is by alternatively update Eqs. 8 [4] 9 and 10. In each iteration, the edges with lowest \(w_{ij}\) are knocked down. Intuitively, the REM can filter out weak links for \(v_i\), and \(v_i\) is connected to at least one other nodes \((H_\alpha(v_i) \geq 0)\).

The degree distributions of eight resulting networks by the two methods are compared in Fig. 6. In the left panels, all four networks have abundant edges, and we observe that the GTE networks have similar degree distribution as the REM networks. After we increase the thresholds to the critical points, the degree distributions...
FIG. 6. The degree distributions of networks after different thresholds and iterations are applied. The GTE networks have zero (a) and 287 (b) isolated nodes for the synthetic data, and 522 (c) and 849 (d) isolated nodes are detected for the real data. The four GTE networks have edge densities of 1%, 0.16%, 2%, and 1.2%, respectively. The corresponding REM networks have similar edge densities as the GTE networks.

of the REM networks are based on numerical data, we believe the approach can be extended to other data types as long as context sets can be generated.

Acknowledgements.—This research is supported by the National Institutes of Health under Grant No. R01AI140760.
[38] M. Grbovic and H. Cheng, In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’18), (2018).

[39] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D.L. Lee., In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’18), 839 (2018).

[40] S. Karamizadeh, S.M. Abdullah, A.A. Manaf, M. Zamani, and A. Hooman, Journal of Signal and Information Processing, 4, 173 (2013). 2013. An overview of principal component analysis. , (3B), p.173.

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

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

[43] L. Van Der Maaten, In Artificial Intelligence and Statistics, 384 (2009).

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

[45] K. Zyczkowski, Syst. Inf. Dyn 10, 297 (2003).

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