Big Data
Volume 8, Number 4, 2020

DOI: 10.1089/big.2019.0169

ORIGINAL ARTICLE

HONEM: Learning Embedding for Higher Order Networks

Mandana Saebi,[1] Giovanni Luca Ciampaglia,[2] Lance M. Kaplan,[3] and Nitesh V. Chawla[1,*]

Abstract

Representation learning on networks offers a powerful alternative to the oft painstaking process of manual feature engineering, and, as a result, has enjoyed considerable success in recent years. However, all the existing representation learning methods are based on the first-order network, that is, the network that only captures the pairwise interactions between the nodes. As a result, these methods may fail to incorporate non-Markovian higher order dependencies in the network. Thus, the embeddings that are generated may not accurately represent the underlying phenomena in a network, resulting in inferior performance in different inductive or transductive learning tasks. To address this challenge, this study presents higher order network embedding (HONEM), a higher order network (HON) embedding method that captures the non-Markovian higher order dependencies in a network. HONEM is specifically designed for the HON structure and outperforms other state-of-the-art methods in node classification, network reconstruction, link prediction, and visualization for networks that contain non-Markovian higher order dependencies.

Keywords: higher order network; network embedding; network representation learning

Introduction

Networks are ubiquitous, representing interactions between components of a complex system. Applying machine-learning algorithms on such networks has typically required a painstaking feature engineering process to develop feature vectors for downstream inductive or transductive learning tasks. For example, a typical link prediction task may require the computation of several network statistics or characteristics such as centrality, degree, and common neighbors.[1] And then a node classification task may require a different feature subset or selection, requiring yet another feature engineering task. This adds significant computational complexity especially with increasing graph sizes. This challenge inspired representation learning algorithms for networks that led to generalized feature representations as low-dimensional embeddings, learned in an unsupervised manner, and thus being flexible enough for different downstream network mining tasks.[2–4]

However, the research on network representation learning has largely focused on first-order networks (FONs), that is, the networks where edges represent the pairwise interactions between the nodes—assuming the naive Markovian property for node interactions (Fig. 1b).[2,5–9] In recognition of the possible higher order interactions between the nodes beyond first order, recent research has led to methods that try to capture the higher order proximity in the network structure. These methods often define a "higher order proximity measure" based on the multihop node connectivity pathways. Such higher order methods perform better in common network mining tasks such as link prediction, network reconstruction, and community detection.[2] However, these methods infer the higher order proximities from the FON structure, which in itself is limiting in capturing the variable and higher order dependencies in a complex system.[10]

Recent research in the network science domain has pointed out the challenges with the FON view and the limitations it poses in network analysis (e.g., community detection,[11–15] node ranking,[16] dynamic processes,[17] risk assessment,[18] and anomaly detection[19]), and proposed higher order network (HON) representation methods that have been shown to be more accurate in capturing the trends in the underlying raw data of a complex system.[10–12,17,20,21] Unlike the conventional FON in which

[1]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, Indiana, USA.
[2]Department of Computer Science and Engineering, University of South Florida, Tampa, Florida, USA.
[3]U.S. Army Research Laboratory, Adelphi, Maryland, USA.
A pre-print version of this article is available at: https://arxiv.org/abs/1908.05387

*Address correspondence to: Nitesh V. Chawla, Department of Computer Science and Engineering, University of Notre Dame, 384 Fitzpatrick Hall, Notre Dame, IN 46556, USA, E-mail: nchawla@nd.edu

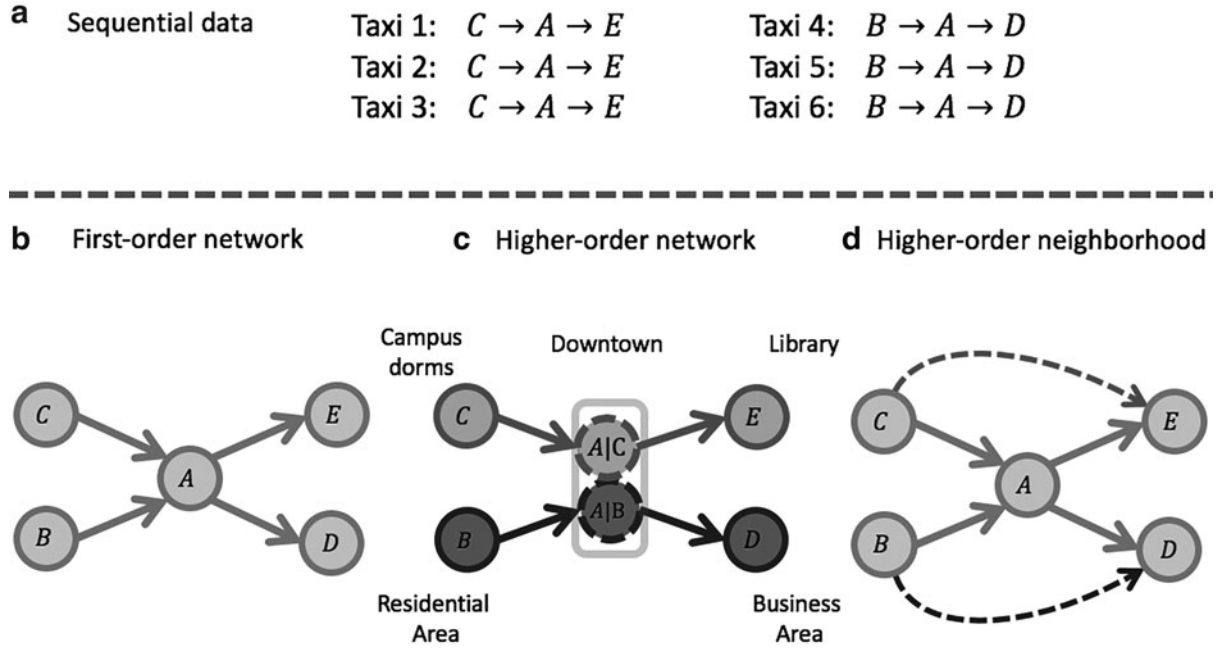

FIG. 1. A toy example showing how higher order neighborhood can be inferred from HON. Given the sequential data provided in **(a)**, we can construct both FON **(b)** and HON. From FON, it is not clear that only node C and E have a second-order dependency through node $A|C$. Similarly, only node B and D have a second-order dependency through node $A|B$. There is no second-order dependency between B and E, or C and D. **(c)** The neighborhood information is inferred from HON **(d)**. FON, first-order network; HON, higher order network.

every node represents a single state, a node in this HON structure represents the current *and* previous states (illustrated in Fig. 1c), thus capturing valuable higher order dependencies in the raw data.[10,11,20,21]

This study advances a representation learning algorithm for HON—higher order network embedding (HONEM)—and shows that representation learning algorithms developed for FON, including ones that capture higher order proximities, are limited in their performance on HON. HONEM is scalable and generalizable to a variety of tasks such as node classification, network reconstruction, link prediction, and visualization.

To that end, this study addresses the following key questions:

1. How to develop a network embedding method that captures the dependencies encoded in the HON structure?
2. Does a network embedding developed specifically for HON offer an advantage in common network mining tasks compared with embedding methods based on FON?

Contributions

The main idea of *HONEM* is to generate a low-dimensional embedding of the networks such that the higher order dependencies (represented in HON) are accurately preserved. HONEM takes HON directly as input and is thus able to capture higher order dependencies present in the raw data that are encoded in HON.

Consider the following example. We are provided human trajectory/traffic data of the area around a university campus. Suppose from the trajectory data, we observe that students who live on-campus are more likely to visit the central library after visiting the downtown area, while people living in a certain residential area are more likely to go to the business area of the city after passing through the downtown area (assuming none of the four locations overlap with each other). In Figure 1, each of the nodes represents the following: C: on-campus dorm, B: residential area, A: downtown area, E: library, D: business area. Suppose we model such dependencies as FON (Fig. 1b), and then try to infer second-order dependencies from FON structure

to derive the node embeddings (as typically done in existing methods[5–8]). In the FON structure, both the library and the business area are two-steps away from the campus dorms (or the residential area). Therefore, we may conclude that students living on-campus have an equal probability of visiting the library and the business area through downtown. As a result, *all the abovementioned methods based on FON will miss important higher order dependencies or infer higher order dependencies that do not exist in the original raw data.* By modeling these interactions as HON, instead (Fig. 1c), we observe that node C and E have a second-order dependency through node $A|C$. Similarly, nodes B and D have a second-order dependency through node $A|B$. There is not a second-order dependency between B and E, or C and D. The question then becomes: how to learn embeddings on HON such that these higher and variable order of dependencies are captured? Methods such as GraRep[22] or Node2Vec[5] assume a fixed k^{th}-order neighborhood to infer the higher order neighborhood. When k is set to 2, these methods assume a second-order relation for all pairs $(C, E), (C, D), (B, D), (B, E)$. However, based on the raw data, there is not a second-order relation between (C, D) or (B, E). HONEM can assign the correct order for each pair of nodes, which can vary depending on the higher order patterns in the raw data. Similarly, if higher order dependencies beyond second order exist in the raw data, methods based on FON cannot discover such patterns with $k \leq 2$.

To summarize, the key contributions of *HONEM* are as follows:

1. Data-agnostic: *HONEM* extracts the *actual* order of dependency from the non-Markovian interactions of the nodes in raw data by allowing for variable orders of dependencies rather than a fixed order for the entire network, as used in prior work.[5,6,8,22]
2. Scalable and parameter-free: *HONEM* does not require a sweep through the parameter space of window length. *HONEM* also does not require any hyperparameter tuning or sampling as is often the case with deep learning or random walk-based embedding methods.
3. Generalizable: *HONEM* embeddings are directly applicable to a variety of network analytic tasks such as network reconstruction, link prediction, node classification, and visualization.

Higher Order Network Embedding

In summary, the *HONEM* algorithm comprises the following steps:

1. Extraction of the higher order dependencies from the raw data.
2. Generation of a higher order neighborhood matrix, given the extracted dependencies.
3. Applying truncated singular value decomposition (SVD) on the higher order neighborhood matrix of the network to discover embeddings, which can then be used by machine-learning algorithms.

Preliminaries

Let us consider a set N of interacting entities and a set S of variable-length sequences of interactions between entities. Given the raw sequential data, the HON can be represented as $G_H = \{N_H, E_H\}$ with E_H edges and N_H nodes of various orders, in which a node can represent a sequence of interactions (path). For example, a higher order node $i|j$ represents the fact that node i is being visited given that node j was previously visited, while a higher order node $i|k, j$ represents the node i given previously visited nodes j and k. In this context, a first-order node p is shown by node $p|\cdot$, in which the notation "$|\cdot$" indicates that no previous step information is included in the data.

Using these higher order nodes and edges in G_H, our goal is to learn embeddings for nodes in the FON, $G_F = \{N_F, E_F\}$. Keep in mind that $N_H \geq N_F$, as several nodes in G_H will correspond to a node in G_F. For example, all HON nodes $A|B, A|C, D$, and $A|E$ represent node A in the FON. It is important to highlight this connection between HON nodes and their FON counterparts. Indeed, we are interested in evaluating our embeddings in a number of machine-learning tasks—such as node classification and link prediction—that are formulated in terms of FON nodes, for example, the class label information is available on A (and not $A|B, A|C, D, A|E$). Therefore, it is important to eventually obtain embeddings for FON nodes.

One approach to address the above challenge is to learn embeddings on higher order nodes $A|B, A|C, D, A|E$ using existing network embedding methods and then combine them to derive the embedding for node A. We experimented with this approach using a different method of combining HON embeddings (max, mean, weighted mean) and realized that it does not scale to large networks, as the number of higher order nodes can be much higher than that of first-order nodes. We

therefore refrain from constructing the HON directly, and modify the "rule extraction" step in the HON algorithm to generate the higher order dependencies and the higher order neighborhood matrix.

Extracting high-order dependencies

The first step of the *HONEM* framework is to extract higher order dependencies from the raw sequential data. To accomplish this task, we modify the rule extraction step in the HON construction algorithm.[10] Please note that the HON algorithm simply results in the network representation of the data but does not generate any embeddings or feature vectors. HON is simply a network representation of the raw data. HONEM, on the contrary, learns embeddings from the said HON to automate the process of feature vector generation. We briefly explain the rule extraction in the HON algorithm below:

Rule extraction (HON). In the FON, all the nodes are assumed to be connected through pairwise interactions.

To discover the higher order dependencies in the sequential data, given a pathway of order k: $\mathcal{S} = [S_{t-k}, S_{t-(k-1)}, \ldots, S_t]$, we follow the steps below:

1. Step 1: Count all the observed paths of length $= 1, 2, \ldots, k$ (where k is the *MaxOrder*) in the sequential data.
2. Step 2: Calculate probability distributions d for next step in each path, given the current and previous steps.
3. Step 3: Extent the current path by checking whether including a previous step $S_{t-(k+1)}$ and extending $\mathcal{S}$ to $\mathcal{S}_{new} = [S_{t-(k+1)}, S_{t-k}, S_{t-(k-1)}, \ldots, S_t]$ (of order $k_{new} = k+1$) will significantly change the normalized count of movements (or the probability distribution, d_{ext}). To detect a significant change, the Kullback–Leibler divergence[23] of S and S_{new}, defined as $d_{KL}(d_{ext}||d)$, is compared with a dynamic threshold, $\delta = \frac{k_{new}}{\log_2(1+\text{Support}_{\mathcal{S}_{new}})}$. If d_{KL} is larger than δ, order k_{new} is assumed as the new order of dependency, and $\mathcal{S}$ will be extended to $\mathcal{S}_{new}$.

This procedure is repeated recursively until a predefined parameter, *MaxOrder*, is reached. However, the new parameter-free version of the algorithm (which is used in the study) does not require setting a predefined *MaxOrder*, and extracts the *MaxOrder* automatically for each sequence. The parameter $\text{Support}_{\mathcal{S}_{new}}$ refers to the number of times the path $\mathcal{S}_{new}$ appears in the raw trajectories. The threshold δ assures that higher orders are only considered if they have sufficient support, which is set with the parameter *MinSupport*. Patterns less frequent than *MinSupport* are discarded. For an example of this procedure, refer to Supplementary Data S1 in the Extracting Higher Order Dependencies section.

The above method only accepts dependencies that are significant and that have occurred a sufficient enough number of times. This is required to ensure that any random pattern in the data will not appear as a spurious dependency rule. Furthermore, this method admits dependencies of variable order for different paths. Using this approach, we extract all possible higher order dependencies from the sequential data. These dependencies are then used to construct the HON. For example, the edge $i|\cdot \rightarrow j|q\cdot$ in the HON corresponds to the rule $i \rightarrow q \rightarrow j$—in other words, i and j are connected through a second-order path.

Modified rule extraction for *HONEM*. In the *HONEM* framework, we modify the standard HON rule extraction approach by preserving all lower orders when including any higher order dependency. This is motivated by a limitation of the previously proposed HON algorithm.[10] In the original HON rule extraction algorithm, after extracting all dependencies, the HON is constructed with the assumption that if higher orders are discovered, all the lower orders (except the first-order) are ignored. *However, discovering a higher order path between two nodes does not imply that the nodes cannot be connected through shorter pathways.* For example, if q and j are connected through the third-order path $q \rightarrow i \rightarrow k \rightarrow j$, and a second-order path $q \rightarrow i \rightarrow j$, they have a second-order dependency as well as a third-order dependency.

Note that in *HONEM*, we extract the higher order dependencies from the sequential data and not from the FON topology, as is done by other methods in the literature.[5,6,8,24] Therefore, our notion of "higher order dependencies" refers to such dependencies that are extracted from sequential data over time. Although these methods are able to improve performance by preserving higher order distances between nodes given the topology of the FON, they are unable to capture dependencies over time. This is important because not all the connections through higher order pathways will occur if they do not exist in the raw sequential data in the first place.

Higher order neighborhood matrix

In the second step of our framework, we design a mechanism for encoding these higher order dependencies into a neighborhood matrix. In this context, we refer to higher order dependencies as *higher order distances*. We define a v^{th}-order neighborhood matrix as D^v, in which the $D^v(i,j)$ element represents the v^{th}-order distance between nodes i and j. Intuitively, D^1 is the first-order adjacency matrix. We derive the neighborhood matrices of various orders until the maximum existing order in the network, k, is reached. The maximum order k is determined by finding the nodes of highest order in the network. For each node pair, the $D^v(i,j)$ distance is obtained by the edge weights of HON (or the corresponding higher order dependencies). For example, in Figure 1 $D^2(C,E)=e_1$ and $D^2(B,D)=e_2$.

It is possible, however, that two given nodes are connected through multiple higher order distances (i.e., multiple paths). In this case, the average probabilities of all paths (or the average edge weights in HON) are considered as the higher order distance. For example, suppose node j can be reached from node i via either path 1: $i \rightarrow q \rightarrow j$ (with probability p_1) or path 2: $i \rightarrow p \rightarrow j$ (with probability p_2). The higher order distance $D^2(i,j)$ between node i and node j is equal to the average edge weight of $q|i \xrightarrow{p_1} j| \cdot \cdot$ and $p|i \xrightarrow{p_2} j|\cdot$, corresponding to path 1 and path 2, respectively. Both of these connections have a second-order dependency. Note that node i (or j) may have different dependency orders, but only second-order ones are included in D^2. Once distances D^v for all desired orders are obtained, we derive the higher order neighborhood matrix S as follows:

$$S = \frac{1}{N}\sum_{k=0}^{L} e^{-k} D^{k+1} \tag{1}$$

For $k=1$, S equals the conventional first-order adjacency matrix. The exponentially decaying weights are chosen to prefer lower order distances over higher orders ones, since higher order paths are generally less frequent in the sequential data.[10] We experimented with increasing and constant weights, and found decaying weights to work best with our method. We leave out the exploration of other potential weighting mechanisms to future work.

It is worth mentioning that the higher order neighborhood matrix provides a richer and more accurate representation of node interactions in FON and thus can be viewed as a means of connecting HON and FON representation. In many network analysis and machine-learning applications—such as node classification and link prediction—working with the HON representation is inconvenient, and requires some form of transformation. *HONEM* provides a more convenient and generalizable interpretation of HON, while preserving the benefits of the more accurate HON representation.

Higher order embeddings

In the third step, the higher order embeddings are obtained by preserving the higher order neighborhood in vector space. A popular method to accomplish this is to obtain embedding vector $\mathbf{U}$ using matrix factorization, in which the objective is to minimize the loss function:

$$min \parallel \mathbf{S} - \mathbf{U}^* \cdot \mathbf{V}^{*T} \parallel_F \tag{2}$$

The widely adopted method for solving the above equation is SVD. Formally, we can factorize a given matrix $\mathbf{S}$ as below:

$$\mathbf{S} = \mathbf{U}^* \delta \mathbf{V}^{*T} \tag{3}$$

where $\mathbf{U}^*, \mathbf{V}^* \in \mathbb{R}^{N \times N}$ are the orthogonal matrices containing content and context embedding vectors. δ is a diagonal matrix containing the singular values in decreasing order.

However, this solution is not scalable to sparse large networks. Therefore, we use truncated SVD[25] to approximate the matrix $\mathbf{S}$ by $\mathbf{S}_d$ ($\mathbf{S} \approx \mathbf{S}_d$) as below:

$$\mathbf{S}_d = \mathbf{U}_d^* \delta_d \mathbf{V}_d^{*T} \tag{4}$$

where $\mathbf{U}_d^*, \mathbf{V}_d^* \in \mathbb{R}^{N \times d}$ contain the first d columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. δ_d contains the top-d singular values. The embedding vectors can then be obtained by means of the following equations: $\mathbf{U}^* = \mathbf{U}_d^* \sqrt{\delta_d}$, $\mathbf{V}^* = \sqrt{\delta_d}\mathbf{V}_d^{*T}$. Without loss of generality, we use $\mathbf{U}^*$ as the embedding matrix.

Experiments

We used three different real-world data sets representing transportation and information networks, and assess the performance on the following tasks: (1) network reconstruction; (2) link prediction; (3) node classification; and (4) visualization. We compared *HONEM* to a number of baselines representing the popular deep learning and matrix factorization-based methods. We provide details on the data and benchmarks first, before presenting the performance results on the aforementioned tasks. We also provide a complexity analysis of *HONEM* in the next section.

Table 1. Basic properties of each data set

	Rome	*Bari*	*Shipping*	*Wiki*
FON nodes	477	522	3058	4043
FON edges	5614	5916	52,366	38,580
FON avg. in-degree	11.76	11.33	17.12	9.54
HON nodes	19,403	13,893	59,779	67,907
HON edges	119,566	88,594	311,691	255,672
HON avg in-degree	6.16	6.37	5.214	3.76
γ	21.29	14.97	5.95	6.62

The gap between the number of first-order and higher order nodes and edges in each data set indicates the density of higher order dependencies in each data.

FON, first-order network; HON, higher order network.

Data sets

The *HONEM* framework can be applied to any sequential data set or HON describing interacting entities to extract latent higher order dependencies among them. To validate our method, we use four different data sets for which raw sequential data are available and there is a higher or variable order of dependency among the nodes. Table 1 summarizes the basic FON and HON properties for each of the data sets. To emphasize the versatility of *HONEM*, these data sets are drawn from three different domains: vehicular traffic flows from two Italian cities (Rome and Bari), Web browsing patterns on Wikipedia, and global freight shipping. Specifically, the four data sets are as follows:

- Traffic data of Rome: This is a car-sharing data provided by Telecom Italia big data challenge 2015,* which contains the trajectories of 616, 356 unique vehicles over 30 days. We divided the city into a grid containing 477 first-order nodes with 5614 edges. Each taxi location is mapped to a node in the grid, and the edges are derived from the number of taxis traveling between the nodes. This data set contains higher order dependencies of 10th order or less. With the inclusion of higher order patterns, the number of nodes and edges increases by 39.67% and 20.29%, respectively. This data set also contains locations of accident claims, which are used for node labeling.
- Traffic data of Bari: This is another car-sharing data (provided by Telecom Italia big data challenge 2015) containing trajectories of 962, 100 taxis over 30 days. We divided the city into a grid containing 522 first-order nodes with 5916 edges (obtained using the same approach as the Rome traffic data). This data set contains higher order dependencies of 12th order or less. With the inclusion of higher order patterns, the number of nodes and edges increases by 25.61% and 13.97%, respectively. This data set also contains locations of accident claims, which are used for node labeling.
- Global shipping data: Provided by Lloyd's Maritime Intelligence Unit; this contains a total of 9, 482, 285 voyages over a span of 15 years (1997–2012). Applying the rule extraction step to this network yields higher order dependencies of up to the 14th order. The number of nodes and edges increase by 18.54% and 4.95%, respectively, after including the higher order patterns in HON.
- Wikipedia game: Available from West and Leskovec[26]; this contains human navigation paths on Wikipedia. In this game, users start at a Wikipedia entry and are asked to reach a target entry by following only hyperlinks to other Wikipedia entries. The data include a total of 4043 articles with 51, 318 incomplete and 24, 875 complete paths. We discarded incomplete paths of length 3 or shorter. This data set contains higher order dependencies of 10th order or less. The inclusion of higher order patterns results in an increase in the number of nodes and edges by 15.79% and 5.62%, respectively.

We define the ratio $\gamma = \frac{\#\text{HONedges}}{\#\text{FONedges}}$ as a measure of the density of higher order dependencies, resulting in a larger gap between FON and HON. The two traffic data sets show the highest gap between FON and HON in terms of the number of nodes and edges. Specifically, the gap is the highest in the traffic data of Rome.

Baselines

We compare our method with the following state-of-the-art embedding algorithms, which only work on FON representation of the raw data.

- DeepWalk[24]: This algorithm uses uniform random walks to generate the node similarity and learns embeddings by preserving the higher order proximity of nodes. It is equivalent to Node2Vec with $p = 1$ and $q = 1$.
- Node2Vec[5]: This method is a generalized version of DeepWalk, allowing biased random walks. We

*https://bit.ly/2UGcEoN

used 0.5, 1, and 2 for p and q values and report the best performing results.

- LINE[6]: This algorithm derives the embeddings by preserving the first- and second-order proximities (and a combination of the two). We ran the experiments for both the second-order and combined proximity, but did not notice a major improvement with the combined version. Thus, we report results only for the embeddings derived from second-order proximity.
- Graph factorization (GF)[27]: This method generates the embeddings by factorizing the adjacency matrix of the network. *HONEM* will reduce to GF if it only uses the first-order adjacency matrix.
- LAP[28]: This method generates the embeddings by performing eigen-decomposition of the Laplacian matrix of the network. In this framework, if two nodes are connected with a large weight, their embeddings are expected to be close to each other.
- GraRep[22]: This is a powerful higher order embedding method that preserves the k-order proximity of the nodes. It uses SVD to factorize the higher order neighborhood of the nodes obtained by the random walk transition probabilities. We use $k=5$ as this value yields the highest performance for this baseline.

Among the above baselines, Node2Vec, DeepWalk, LINE, and GraRep learn embeddings using higher order proximities. GraRep in particular goes beyond second-order proximity and is the closest method to ours, but it extracts the higher order proximity from the FON structure and only accepts a fixed order of dependency. We also used locally linear embedding (LLE) as a baseline in our early experiments. However, LLE failed to converge on several dimensions in link prediction and network reconstruction experiments. Therefore, we did not include it in the final results.

Network reconstruction

Network embedding can be interpreted as a compression of the graph.[2,8] An accurate compression should be able to reconstruct the original graph from the embeddings. To accomplish this, we use the embeddings to predict the original links of the network. This task is closely related to the link prediction task, where the goal is to predict the future links using the existing links of the graph. However, in the reconstruction task, we use the existing links as ground truth. Please note that this is different from the link prediction task, which is trying to predict the probability of link formation in the future.

Network reconstruction is an important evaluation task for representation learning algorithms, as it provides an insight into the quality of the embeddings generated by the method. We measure the reconstruction precision for the top k evaluated edge pairs using $Precision@k = \frac{1}{k}\sum_{i=1}^{k} \delta_i$, where $\delta_i = 1$ when the i_{th} reconstructed edge is correctly recovered, and $\delta_i = 0$ otherwise.

Figure 2b shows the network reconstruction results with varying k. We notice that although the performance of other baselines is data dependent, *HONEM* performs significantly better on all data sets. Results on both the traffic data sets display similar trends, and methods such as LINE, which perform relatively well on these data sets but fail on the larger data sets (shipping and Wikipedia). *HONEM* not only performs better than GF, which preserves the first-order proximity, but also outperforms Node2Vec, DeepWalk, LINE, and GraRep, which preserve the higher order proximity based on FON. GraRep is the second-best performing baseline in all data sets except the shipping data, but still does poorly compared with HONEM. With the increase in k, all of the actual edges are recovered but the number of possible pairs of edges becomes too large, and thus, almost all methods converge to a small value. However, there is still a large gap between *HONEM* and other baselines even at the largest k on all data sets.

Link prediction

We posit that embeddings derived from HON perform better for link prediction as those embeddings are more accurately capturing the higher and variable order dependencies in a complex system, which are missed by the FON representation that contemporary embedding methods work on. Methods based on FON do not capture the non-Markovian higher order interactions of the nodes, which creates a potential for link formation. For example, suppose there is a directed edge in HON from $B|\cdot$ to $A|\cdot$, denoted by $B|\cdot \rightarrow A|\cdot$ (corresponding to the path $B \rightarrow A$) and another directed edge from a second-order node $C|D$ to B, denoted by $C|D \rightarrow B|\cdot$ (corresponding to the path $D \rightarrow C \rightarrow B$). In this structure, node A can be reached within three steps from node D. In FON, however, we only have $D \rightarrow C$, $C \rightarrow B$, and $B \rightarrow A$. Therefore, FON might miss the potential interesting edge between D and A, or D and

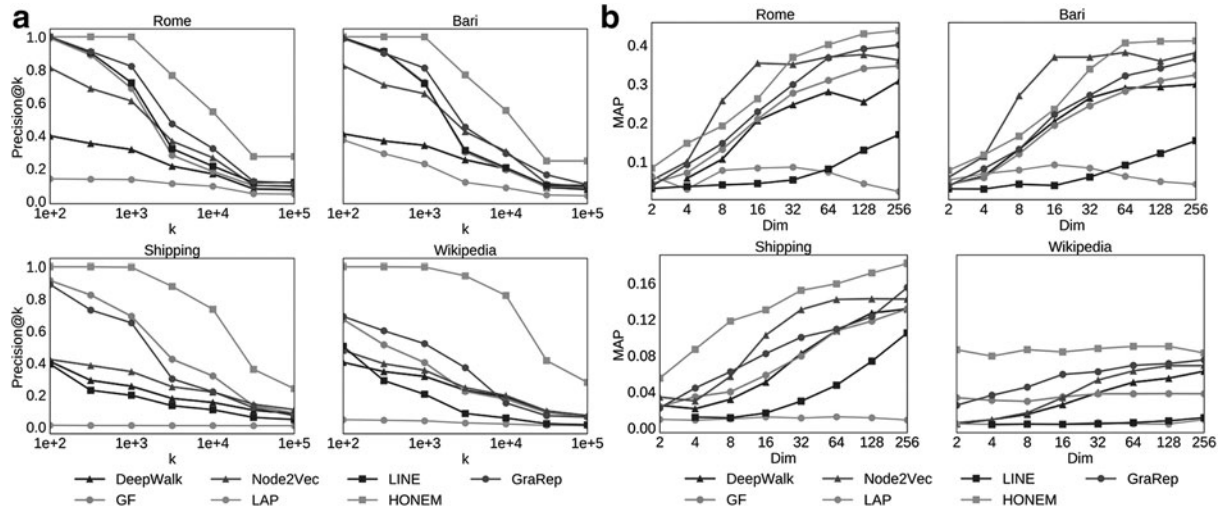

FIG. 2. **(a)** Reconstruction results. The *x*-axis represents the number of evaluated edge pairs. *HONEM* performs better than other baselines with a large margin. **(b)** Link prediction results. The *x*-axis indicates the embedding dimension. *HONEM* provides the best performance on all data sets in dimension 64 or more. In the traffic data set, even though Node2Vec provides better MAP scores. *HONEM* provides the best precision for the top-*k* predictions (refer to Table 2). GF, graph factorization; HONEM, higher order network embedding.

B. To validate our argument, we remove 20% of the edges from the current network, and derive node embeddings on the remaining network using *HONEM*. We then predict the missing edges by calculating the pairwise distance between embedding vectors and select the top highest values as potential edges.

We use mean average precision (MAP) as the link prediction evaluation metric. MAP is the average precision over all the nodes, and is defined as: $\text{MAP} = \frac{\sum_i AP(i)}{N_F}$, Where $AP(i) = \frac{\sum_k Precision@k(i) \times \delta_{ik}}{\sum_k \delta_{ik}}$ in which $Precision@k(i) = \frac{1}{k}\sum_{j=1}^{k} \delta_{ij}$. k is the number of evaluated edges, $\delta_{ij} = 1$ when the j_{th} reconstructed edge for node i exists in the original network, and $\delta_{ij} = 0$ otherwise. We evaluated link prediction using the $Precision@k$ measure on dim = 128 as well (refer to the Supplementary Data S1 for details). However, since we are interested to analyze the effect of dimension, we provide MAP as a precision measure for all nodes. The results are displayed in Figure 2a. We notice that the MAP score is generally lower in larger data sets, namely Shipping and Wiki (due to sparsity). In the traffic data sets (Bari and Rome), the *HONEM* shows a monotonically increasing performance with increasing the embedding dimension, while the performance of other methods either saturates after a certain dimension or deteriorates.

Effect of dimensionality. Overall, *HONEM* provides superior performance in dimensions of 64 or larger. We notice that while Node2Vec provides a better MAP score on the traffic data sets in lower dimensions (smaller than 64 in Bari and smaller than 32 in Rome), it fails to improve over higher dimensions. We further investigated our results by visualizing the node precision, $AP(i)$, over various dimensions on the Rome city map. The results are shown in Figure 3. We realize that nodes with the highest precision (darker color) are located in the high-traffic city zones (green lines show the major highways of the city). Based on our analysis, nodes located in the high-traffic zones are 80.56% more likely to have a dependency of second order or more. As a result, we observe that in lower dimensions, *HONEM* consistently exhibits high precision for these higher order nodes. As the dimension increase, the precision of the lower order nodes also increases. On the contrary, node precision obtained by Node2Vec is not related to the node location. In dim = 32 and dim = 64, *HONEM* provides an overall better coverage and better precision than Node2Vec. A comparison of the top-*k* ($k = 1024$) prediction between Node2Vec and *HONEM* is provided in Table 2. Even though Node2Vec provides better MAP scores in lower dimensions, *HONEM* provides better precision for the top-*k*

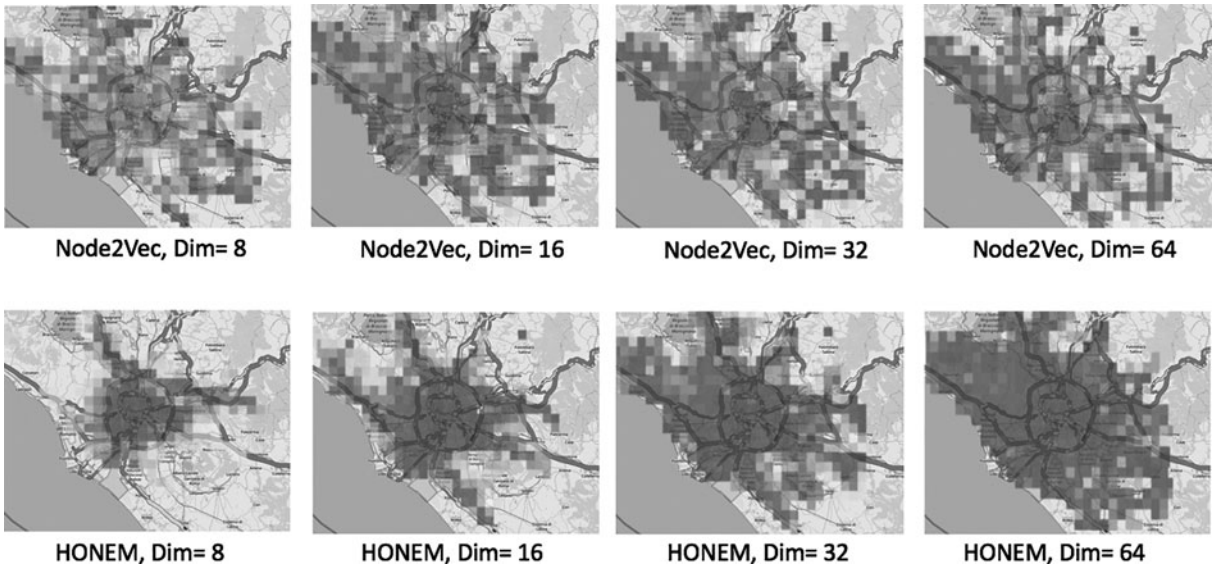

FIG. 3. Variation of node precision with embedding dimension for Rome. The highlighted green lines indicate the major traffic routes of the city. The node color intensity indicates the link prediction precision for node *i* [*AP*(*i*)]. In lower dimension, higher order nodes have the best precision using *HONEM*. The precision of other nodes increases in higher dimensions, eventually outperforming Node2Vec in Dim = 32 and Dim = 64. Node2Vec does not differentiate between higher order and first-order nodes in lower dimensions.

predictions. Looking back at data characteristics, we notice that this phenomenon only happens for the traffic data set, where γ is significantly larger than the other two data sets. Therefore, in data sets with significant higher order dependencies resulting in a large gap between HON and FON, our method provides the best precision for the potentially most important nodes (i.e., those of higher order).

Node classification

We hypothesize that higher order dependencies can reveal important node structural roles. In this section, we validate this hypothesis using experiments on real-world data sets. Our goal is to find out whether *HONEM* can improve the node classification accuracy by encoding the knowledge of higher order dependencies.

Table 2. Comparison of Precision@k ($k = 1024$) for link prediction using Node2Vec and higher order network embedding over various dimensions

	4	*8*	*16*	*32*	*64*	*128*	*256*
Node2Vec	0.079	0.152	0.165	0.184	0.195	0.1536	0.141
HONEM	0.316	0.364	0.409	0.528	0.529	0.543	0.591

Even though Node2Vec provides better MAP score in lower dimensions, it fails to accurately predict the top-*k* links.

HONEM, higher order network embedding; MAP, mean average precision.

We answer the above question by comparing state-of-the-art node embedding methods based on FON and our proposed embedding method, *HONEM*, capturing higher order dependencies. We evaluate our method on four different data sets and compare the performance with state-of-the-art embedding methods based on FON. In the traffic data, nodes are labeled given the likelihood of having accidents (i.e., "Low" or "High"). In Wikipedia, the nodes are labeled based on whether or not they are reachable within less than 5 clicks in the network. In the shipping data, nodes are labeled given the volume of the shipping traffic (i.e., "Low" or "High"). We use 70% of the data for training and 30% for testing. Our experiments show that compared with five state-of-the-art embedding methods, *HONEM* yields significantly more accurate results across all data sets regardless of the type of classifier used.

We evaluated the node classification performance using area under receiver operating characteristic curve across four different classifiers: Logistic Regression, Random

Forest, Decision Tree, and AdaBoost. The results are shown in Figure 4. We observe that *HONEM* performs consistently better than other embedding methods. Specifically, we analyzed the *HONEM* advantage in each data set. We noticed that in the traffic data sets, nodes with more higher order dependencies are more likely to have an accident (Pearson correlation: 0.7535, $p < 0.005$). In the Wikipedia data, reachable nodes are more likely to have higher order dependencies (Pearson correlation: 0.6845, $p < 0.001$). In the shipping data, nodes with higher shipping traffic contain more higher order dependencies (Pearson correlation: 0.8612, $p < 0.005$). Such higher order signals do not emerge in methods based on FON (regardless of the method complexity). Furthermore, we notice that *HONEM* is fairly robust to the type of classifier. However, Decision Tree performs poorly regardless of the embedding method, as it picks a subset of features that do not fully capture the node representation in the network. In line with expectations, ensemble methods perform better overall, even though Logistic Regression offers competitive performance on the Wikipedia data set.

Visualization

To provide a more intuitive interpretation for the improvement offered by *HONEM*, we compare visualizations of the produced embeddings against those of the baseline methods. As a case example, we visualize the subgraphs corresponding to two different topics from the Wikipedia data set. This is shown in Figure 5. Topics were selected from standard Wikipedia categories. Here, we show results for mathematics and geography, as they arguably represent two topics that are comparable in terms of generality but are also distinct enough to allow for meaningful interpretation. We use t-SNE[29] to map 128-dimensional embeddings to the 2-dimensional coordinates. Figure 5 shows two separate clusters for the embeddings derived from *HONEM*. However, it is possible to notice that a number of mathematics entries are interspersed with geography entries.

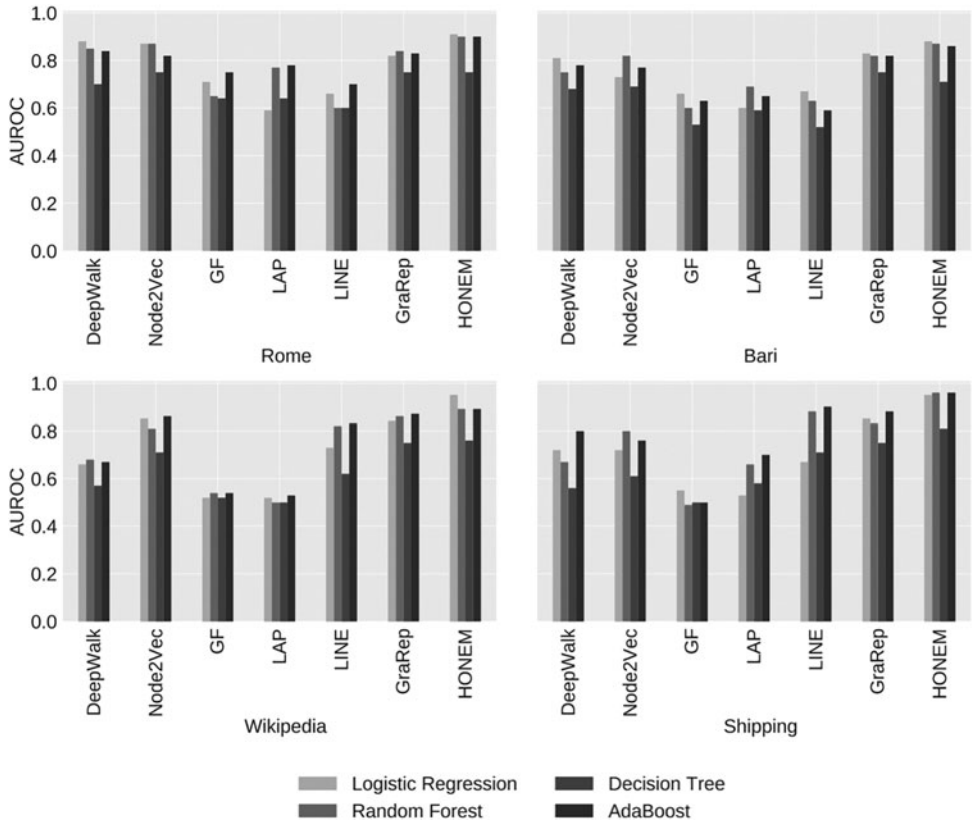

FIG. 4. Node classification results. *HONEM* performs better across all data sets and is fairly robust to the type of the classifier.

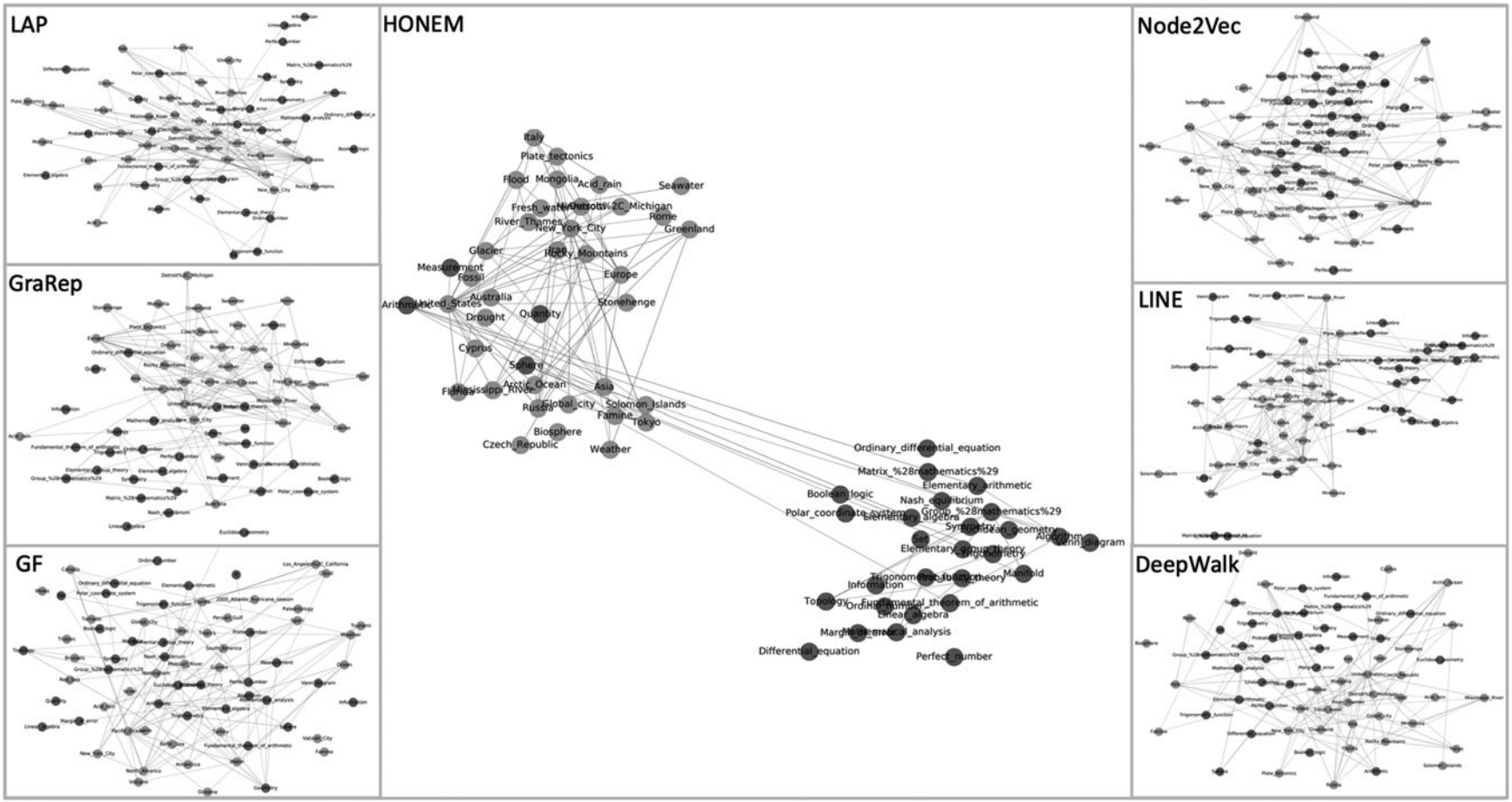

FIG. 5. Visualization of mathematics and geography topics in the Wikipedia data set.

These are the nodes of encyclopedic entries such as *Sphere, Quantity, Arithmetic, Measurement*, which, although primarily categorized under mathematics, are also related to many other topics—including geography.

Figure 5 shows also the visualization results for the baselines. We observe that for many methods, the clusters are not as neatly distinguishable as those produced by *HONEM*. Specifically, DeepWalk, Node2Vec, and GraRep display separate clusters, but there are many misclassified nodes within each cluster. With GF and LINE, it is even more difficult to identify proper clustering among the articles. This indicates that higher order patterns are important to distinguish clusters and capture node concepts within the network.

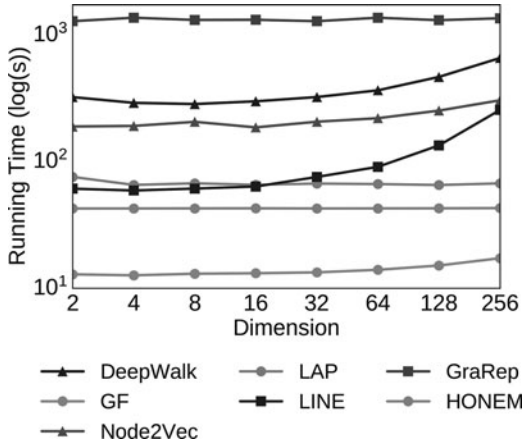

FIG. 6. Comparison of the running time on the global shipping data. *HONEM* provides the best running time after GF. Both methods are robust to embedding dimension.

Analysis of Running Time

The running time of *HONEM* consists of the time required for extracting the higher order dependencies and the time required for factorizing the higher order local neighborhood matrix. In practice, this is dominated by the time complexity of extracting higher order dependencies. To analyze this complexity, suppose the size of raw sequential data is L, and N is the number of unique entities in the raw data. Then, the time complexity of the algorithm is $\Theta(N(2R_1 + 3R_2 + \dots))$, where R_k is the *actual* number of higher order dependencies for order k: all observations will be traversed at least once. Testing whether adding a previous step significantly changes the probability distribution of the next step (using Kullback–Leibler divergence) takes up to $\Theta(N)$ time.[19]

We compare the running time of *HONEM* with the state-of-the-art baselines on the shipping data. We tested the running time on other data sets and found the shipping data to be the most challenging, both in terms of the number of nodes and edges, and network density. All the experiments were run on the same machine (Intel(R) Xeon(R) CPU E7-4850 v4 @ 2.10GHz). The results are shown in Figure 6. The running time of *HONEM* is robust to the embedding dimension. We notice that GF is the only method having better running time than *HONEM*. This is understandable since GF directly factorizes the first-order adjacency matrix of the network, while *HONEM* requires extra time for extracting the higher order neighborhood. However, the difference in running time of *HONEM* and GF translates to significantly better performance in link prediction, network reconstruction, and node classification. Moreover, higher order dependencies only need to be extracted once for each data set (regardless of the embedding dimension). However, for fair comparison, we added this time for experiments over all dimensions.

Related Work

Higher order networks

Networks have become a common way of representing rich interactions among the components of a complex system. As a result, it is critical for the network model to accurately capture the inherent phenomena in the underlying system. This has motivated a new line of research on HON models that are capable of capturing complex interactions beyond the pairwise node relations. Motif-based higher order models,[12,30,31] multilayer higher order models,[32,33] and non-Markovian higher order models[10,11,17] are examples of efforts for more accurate network models. In particular, non-Markovian models have been shown to be more accurate in community detection,[11–15] node ranking,[16] dynamic processes,[17] risk assessment,[18] and anomaly detection[19] system.[10–12,17,20,21] In this work, we use the non-Markovian network model proposed by Xu et al.[10] due to its accuracy and efficiency in modeling higher order dependencies.

Network representation learning

Recent advances in graph mining have motivated the need to automate feature engineering from networks.

This problem finds its roots in traditional dimensionality reduction techniques.[34–36] For example, LLE[34] represents each node as a linear combination of its immediate neighbors, and LE[28] uses the spectral properties of the Laplacian matrix of the network to derive node embeddings.

More recently, methods based on random walks, matrix factorization, and deep learning have been proposed as well, although applicable to FONs. DeepWalk[24] learned node embeddings by combining random walks with the skip-gram model.[37] Node2Vec[5] extended this approach further, proposing to use biased random walks to capture any homophily and structural equivalence present in the network. A random walk-based method for knowledge graph embedding is proposed in Yu et al.[38] Role2Vec[39] further leverages attributed random walks to capture the behavioral roles of the nodes. In contrast, factorization methods derive embeddings by factorizing a matrix that represents the connections between nodes. GF[27] explicitly factorizes the adjacency matrix of the FON. LINE[6] attempts to preserve both first-order and second-order proximities by defining explicit functions. GraRep[22] and HOPE[8] go beyond second order, and factorize a similarity matrix containing higher order proximities. Walklets[40] approximates the higher order proximity matrix by skipping over some nodes in the network. Qiu et al.[41] show that LINE, Node2Vec, DeepWalk, and predictive text embedding[42] are implicitly factorizing a higher order proximity matrix of the network. Rossi et al.[4] propose another taxonomy by categorizing the existing embedding methods into role-based[9,39,43,44] and community-based methods.[5,6,22,24,45] A new crop of methods have been proposed recently that allow for dependencies of arbitrary order.[7,22] However, this order needs to be set by the user beforehand. Therefore, these methods are unable to extract the order of the system from raw sequential data and fail to identify the higher order dependencies of the network without trial and error. HONE[9] uses motifs as higher order structures, however, these motifs do not capture higher order dependencies stemming from non-Markovian interactions in the raw data. In addition, several deep learning-based methods have also been proposed. SDNE[46] uses autoencoders to preserve first-order and second-order proximities. DNGR[47] combines autoencoders with random surfing to capture higher order proximities beyond second order. However, both methods present high computational complexity. Models based on convolutional neural networks were proposed to address the complexity issue.[45,48–50]

Finally, dynamic approaches have been recently proposed to capture the evolution of the network with embeddings.[51–57] These methods still feature a computationally demanding task of dynamic network modeling. Furthermore, these methods are developed based on the FON structure and require specification of a time window, making them data dependent.

To the best of our knowledge, there is a gap in the literature when it comes to representation learning approaches that capture the higher order dependencies based on the raw data. *HONEM* fills an important and critical gap in the literature by addressing the challenges of learning embeddings from the higher order dependencies in a network, thereby providing a more accurate and effective embedding.

Note that although the raw trajectory data are collected over a period of time, we view this as a single snapshot to build both FON and HON. All the corresponding higher order dependencies in this snapshot are encoded as rules into the HON structure, which *HONEM* uses as higher order neighborhood information. Other static methods based on FON use the same sequential data but only consider pairwise relations to build the FON. Therefore, *HONEM* is not a dynamic network representation learning approach. We leave the exploration of the dynamic scenario for future work.

Conclusion

In this study, we developed *HONEM*, a representation learning algorithm that captures the higher and variable order dependencies in the HONs. HONEM works directly on HON and is able to discover embeddings that preserve the higher order dependencies based on non-Markovian interactions of the nodes. We show that the contemporary representation learning algorithms fail to capture higher order dependencies, resulting in missing important information and thus inaccuracies when dealing with HON. *HONEM*, on the contrary, extracts the significant higher order proximities from the data to construct the higher order neighborhood matrix of the network. The node embeddings are obtained by applying truncated SVD on the higher order neighborhood matrix. We demonstrate that compared with five state-of-the-art methods, *HONEM* performs better in node classification, link prediction, network reconstruction, and visualization tasks. We show that HONEM is computationally efficient and scalable to large data sets.

There are several directions for future improvements. In particular, different weighting mechanisms for modeling the effect of distance matrix for various orders can be explored. The *HONEM* framework creates a new path for the exploration of HONs. In the context of network embedding, various decomposition methods—other than truncated SVD—can be applied to learn the node embeddings from the proposed higher order neighborhood matrix. We also plan to implement the dynamic version of *HONEM* that can update the embeddings based on snapshots of HON over time.

Author Disclosure Statement

No competing financial interests exist.

Funding Information

This study is based on research supported by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (PI: N.V.C.). G.L.C. acknowledges support from the Indiana University Network Science Institute.

Supplementary Material

Supplementary Data S1
Supplementary Table S1
Supplementary Figure S1
Supplementary Figure S2

References

1. Lichtenwalter RN, Lussier JT, Chawla NV. New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010. pp. 243–252.
2. Goyal P, Ferrara E. Graph embedding techniques, applications, and performance: A survey. Knowl Based Syst 2018;151:78–94.
3. Cui P, Wang W, Pei J, Zhu W. A survey on network embedding. IEEE Trans Knowl Data Eng 2018;31:833–852.
4. Rossi RA, Jin D, Kim S, et al. From community to role-based graph embeddings. arXiv preprint arXiv:1908.08572, 2019.
5. Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016. pp. 855–864.
6. Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015. pp. 1067–1077.
7. Zhang Z, Cui P, Wang X, et al. Arbitrary-order proximity preserved network embedding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2018. pp. 2778–2786.
8. Ou M, Cui P, Pei J, et al. Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016. pp. 1105–1114.
9. Rossi RA, Ahmed NK, Koh E. Higher-order network representation learning. In: Companion of the Web Conference 2018 on the Web Conference 2018. International World Wide Web Conferences Steering Committee, 2018. pp. 3–4.
10. Xu J, Wickramarathne TL, Chawla NV. Representing higher-order dependencies in networks. Sci Adv 2016;2:e1600028.
11. Rosvall M, Esquivel AV, Lancichinetti A, et al. Memory in network flows and its effects on spreading dynamics and community detection. Nat Commun 2014;5:1–13.
12. Benson AR, Gleich DF, Leskovec J. Higher-order organization of complex networks. Science 2016;353:163–166.
13. Benson AR, Gleich DF, Leskovec J. Tensor spectral clustering for partitioning higher-order network structures. In: Proceedings of the 2015 SIAM International Conference on Data Mining. SIAM, 2015. pp. 118–126.
14. Zhou D, Zhang S, Yildirim MY, et al. A local algorithm for structure-preserving graph cut. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017. pp. 655–664.
15. Zhou D, He J, Davulcu H, Maciejewski R. Motif-preserving dynamic local graph cut. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018. pp. 1156–1161.
16. Scholtes I, Wider N, Garas A. Higher-order aggregate networks in the analysis of temporal networks: Path structures and centralities. Eur Phys J B 2016;89:61.
17. Scholtes I, Wider N, Pfitzner R, et al. Causality-driven slow-down and speed-up of diffusion in non-markovian temporal networks. Nat Commun 2014;5:5024.
18. Saebi M, Xu J, Grey EK, et al. Higher-order patterns of aquatic species spread through the global shipping network. BioRxiv, p. 704684, 2019.
19. Xu J, Saebi M, Ribeiro B, et al. Detecting anomalies in sequential data with higher-order networks. arXiv preprint arXiv:1712.09658, 2017.
20. Scholtes I. When is a network a network? Multi-order graphical model selection in pathways and temporal networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017. pp. 1037–1046.
21. Lambiotte R, Rosvall M, Scholtes I. Understanding complex systems: From networks to optimal higher-order models. arXiv preprint arXiv: 1806.05977, 2018.
22. Cao S, Lu W, Xu Q. Grarep: Learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM, 2015. pp. 891–900.
23. Kullback S, Leibler RA. On information and sufficiency. Ann Math Stat 1951;22:79–86.
24. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2014. pp. 701–710.
25. Eckart C, Young G. The approximation of one matrix by another of lower rank. Psychometrika 1936;1:211–218.
26. West R, Leskovec J. Human wayfinding in information networks. In: Proceedings of the 21st International Conference on World Wide Web. ACM, 2012. pp. 619–628.
27. Ahmed A, Shervashidze N, Narayanamurthy S. et al. Distributed large-scale natural graph factorization. In: Proceedings of the 22nd International Conference on World Wide Web. ACM, 2013. pp. 37–48.
28. Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 2003;15:1373–1396.
29. Maaten Lvd, Hinton G. Visualizing data using t-SNE. J Mach Learn Res 2008;9:2579–2605.
30. Arenas A, Fernandez A, Fortunato S, Gomez S. Motif-based communities in complex networks. J Phys A Math Theor 2008;41:224001.
31. Petri G, Scolamiero M, Donato I, Vaccarino F. Topological strata of weighted complex networks. PLoS One 2013;8:e66506.
32. Kivelä M, Arenas A, Barthelemy M, et al. Multilayer networks. J Complex Netw 2014;2:203–271.
33. De Domenico M, Granell C, Porter MA, Arenas A. The physics of spreading processes in multilayer networks. Nat Phys 2016;12:901.
34. Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science 2000;290:2323–2326.
35. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometr Intell Lab Syst 1987;2:37–52.
36. Kruskal JB, Wish M. Multidimensional scaling. In: Sage University Paper Series on Quantitative Applications in the Social Sciences, No. 07-011. Newbury Park: Sage Publications, 1978.

37. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. 2013. pp. 3111–3119.
38. Yu H, Kulkarni V, Wang W. Mohone: Modeling higher order network effects in knowledgegraphs via network infused embeddings. arXiv preprint arXiv:1811.00198, 2018.
39. Ahmed NK, Rossi RA, Lee JB. et al. role2vec: Role-based network embeddings. In: Proceedings of the DLG KDD, 2019.
40. Perozzi B, Kulkarni V, Skiena S. Walklets: Multiscale graph embeddings for interpretable network classification. arXiv preprint arXiv:1605.02115, 2016.
41. Qiu J, Dong Y, Ma H, et al. Network embedding as matrix factorization: Unifying deepwalk, line, PTE, and node2vec. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. ACM, 2018. pp. 459–467.
42. Tang J, Qu M, Mei Q. PTE: Predictive text embedding through large-scale heterogeneous text networks. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015. pp. 1165–1174.
43. Sankar A, Zhang X, Chang KCC. Motif-based convolutional neural network on graphs. arXiv preprint arXiv:1711.05697, 2017.
44. Ribeiro LF, Saverese PH, Figueiredo DR. struc2vec: Learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017. pp. 385–394.
45. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, 2017. pp. 1024–1034.
46. Wang D, Cui P, Zhu W. Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016. pp. 1225–1234.
47. Cao S, Lu W, Xu Q. Deep neural networks for learning graph representations. In: AAAI, 2016. pp. 1145–1152.
48. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.
49. Henaff M, Bruna J, LeCun Y. Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163, 2015.
50. Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493, 2015.
51. Ma J, Cui P, Zhu W. Depthlgp: Learning embeddings of out-of-sample nodes in dynamic networks. In: The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 2018.
52. Yin H, Benson AR, Leskovec J, Gleich DF. Local higher-order graph clustering. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017. pp. 555–564.
53. Zuo Y, Liu G, Lin H, et al. Embedding temporal network via neighborhood formation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2018. pp. 2857–2866.
54. Zhou L-K, Yang Y, Ren X, et al. Dynamic network embedding by modeling triadic closure process. In: The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 2018.
55. Sankar A, Wu Y, Gou L, et al. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020. pp. 519–527.
56. Kumar S, Zhang X, Leskovec J. Learning dynamic embeddings from temporal interaction networks. Learning 2018;17:29.
57. Nguyen GH, Lee JB, Rossi RA, et al. Dynamic network embeddings: From random walks to temporal random walks. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018. pp. 1085–1092.

Cite this article as: Saebi M, Ciampaglia GL, Kaplan LM, Chawla NV (2020) HONEM: learning embedding for higher order networks. *Big Data* 8:4, 255–269, DOI: 10.1089/big.2019.0169.

Abbreviations Used

FON = first-order network
GF = graph factorization
HON = higher order network
HONEM = higher order network embedding
LLE = locally linear embedding
SVD = singular value decomposition