The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Structural Deep Embedding for Hyper-Networks

Ke Tu,[1] **Peng Cui,**[1] **Xiao Wang,**[1] **Fei Wang,**[2] **Wenwu Zhu**[1]
[1] Department of Computer Science and Technology, Tsinghua University
[2] Department of Healthcare Policy and Research, Cornell University
tuke1993@gmail.com, cuip@tsinghua.edu.cn, wangxiao007@mail.tsinghua.edu.cn
feiwang03@gmail.com,wwzhu@tsinghua.edu.cn

Abstract

Network embedding has recently attracted lots of attentions in data mining. Existing network embedding methods mainly focus on networks with pairwise relationships. In real world, however, the relationships among data points could go beyond pairwise, i.e., three or more objects are involved in each relationship represented by a hyperedge, thus forming hyper-networks. These hyper-networks pose great challenges to existing network embedding methods when the hyperedges are indecomposable, that is to say, any subset of nodes in a hyperedge cannot form another hyperedge. These indecomposable hyperedges are especially common in heterogeneous networks. In this paper, we propose a novel *Deep Hyper-Network Embedding (DHNE)* model to embed hyper-networks with indecomposable hyperedges. More specifically, we theoretically prove that any linear similarity metric in embedding space commonly used in existing methods cannot maintain the indecomposibility property in hyper-networks, and thus propose a new deep model to realize a non-linear tuplewise similarity function while preserving both local and global proximities in the formed embedding space. We conduct extensive experiments on four different types of hyper-networks, including a GPS network, an online social network, a drug network and a semantic network. The empirical results demonstrate that our method can significantly and consistently outperform the state-of-the-art algorithms.

Introduction

Nowadays, networks are widely used to represent the rich relationships of data objects in various domains, forming social networks, biology networks, brain networks, etc. Many methods are proposed for network analysis, among which network embedding methods (Tang et al. 2015; Wang, Cui, and Zhu 2016; Deng et al. 2016; Ou et al. 2015) arouse more and more interests in recent years. Most of the existing network embedding methods are designed for conventional pairwise networks, where each edge links only a pair of nodes. However, in real world applications, the relationships among data objects are much more complicated and they typically go beyond pairwise. For example, John purchasing a shirt with cotton material forms a high-order relationship ⟨John, shirt, cotton⟩. The network capturing those high-order node relationships is usually referred to as a hyper-network.

A typical way to analyze hyper-network is to expand them into conventional pairwise networks and then apply the analytical algorithms developed on pairwise networks. Clique expansion (Sun, Ji, and Ye 2008) (Figure 1 (c)) and star expansion (Agarwal, Branson, and Belongie 2006) (Figure 1 (d)) are two representative techniques to achieve such a goal. In clique expansion, each hyperedge is expanded as a clique. In star expansion, a hypergraph is transformed into a bipartite graph where each hyperedge is represented by an instance node which links to the original nodes it contains. These methods assume that the hyperedges are *decomposable* either explicitly or implicitly. That is to say, if we treat a hyperedge as a set of nodes, then any subset of nodes in this hyperedge can form another hyperedge. In a homogeneous hyper-network, this assumption is reasonable, as the formation of hyperedges are, in most cases, caused by the latent similarity among the involved objects such as common labels. However, when learning the heterogeneous hyper-network embedding, we need to address the following new requirements.

1. **Indecomposablity**: The hyperedges in heterogeneous hyper-networks are usually indecomposable. In this case, a set of nodes in a hyperedge has a strong relationship, while the nodes in its subset does not necessarily have a strong relationship. For example, in the recommendation system with ⟨user, movie, tag⟩ relationships, the ⟨user, tag⟩ relationships are not typically strong. This means that we cannot simply decompose hyperedges using those traditional expansion methods.
2. **Structure Preserving**: The local structures are preserved by the observed relationships in network embedding. However, due to the sparsity of networks, many existing relationships are not observed. It is not sufficient for preserving hyper-network structure using only local structures. And global structures, e.g. the neighborhood structure, are required to address the sparsity problem. How to capture and preserve both local and global structures simultaneously in a hyper-network is still an unsolved problem.

To address **Indecomposablity** issue, we design an indecomposable tuplewise similarity function. The function is

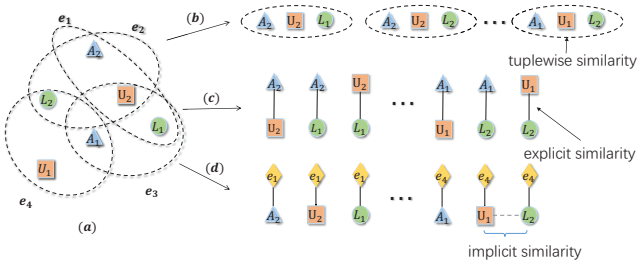

Figure 1: (a) An example of a hyper-network. (b) Our method. (c) The clique expansion. (d) The star expansion. Our method models the hyperedge as a whole and the tuplewise similarity is preserved. In clique expansion, each hyperedge is expanded into a clique. Each pair of nodes has explicit similarity. As for the star expansion, each node in one hyperedge links to a new node which stands for the origin hyperedge. Each pair of nodes in the origin hyperedge has implicit similarity for the reason that they link to the same node.

directly defined over all the nodes in a hyperedge, ensuring that the subsets of a hyperedge are not incorporated in network embedding. We theoretically prove that the indecomposable tuplewise similarity function can not be a linear function. Therefore, we realize the tuplewise similarity function with a deep neural network and add a non-linear activation function to make it highly non-linear. To address **Structure Preserving** issue, we design a deep autoencoder to learn node representations by reconstructing neighborhood structures, ensuring that the nodes with similar neighborhood structures will have similar embeddings. The tuplewise similarity function and deep autoencoder are jointly optimized to simultaneously address the two issues.

It is worthwhile to highlight the following contributions of this paper:

- We investigate the problem of *indecomposable hyper-network embedding*, where indecomposibility of hyperedges is a common property in hyper-networks but largely ignored in literature. We propose a novel deep model, named Deep Hyper-Network Embedding (DHNE), to learn embeddings for the nodes in heterogeneous hyper-networks, which can simultaneously address indecomposable hyperedges while preserving rich structural information. The complexity of this method is linear to the number of node and it can be used in large scale networks
- We theoretically prove that any linear similarity metric in embedding space cannot maintain the indecomposibility property in hyper-networks, and thus propose a novel deep model to simultaneously maintain the indecomposibility as well as the local and global structural information in hyper-networks.
- We conduct experiments on four real-world information networks. The results demonstrate the effectiveness and efficiency of the proposed model.

The remainder of this paper is organized as follows. In the next section, we review the related work. Section 3 gives the preliminaries and formally defines our problem. In Section 4, we introduce the proposed model in details. Experimental results are presented in section 5. Finally, we conclude in Section 6.

Related work

Our work is related to network embedding which aims to learn low-dimension representations for networks. Earlier works, such as Local Linear Embedding (LLE) (Roweis and Saul 2000), Laplacian eigenmaps (Belkin and Niyogi 2001) and IsoMap (Tenenbaum, De Silva, and Langford 2000), are based on matrix factorization. They express a network as a matrix where the entries represent relationships and calculate the leading eigenvectors as network representations. Eigendecomposition is a very expensive operation so these methods cannot efficiently scale to large real world networks. Recently, DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) learns latent representations of nodes in a network by modeling a stream of short random walks. LINE (Tang et al. 2015) optimizes an objective function which aims to preserve both the first-order and second-order proximities of networks. HOPE (Ou et al. 2016) extends the work to utilize higher-order information and M-NMF (Wang et al. 2017) incorporates the community structure into network embedding. Furthermore, due to the powerful representation ability of deep learning (Niepert, Ahmed, and Kutzkov 2016), several network embedding methods based on deep learning (Chang et al. 2015; Wang, Cui, and Zhu 2016) have been proposed. (Wang, Cui, and Zhu 2016) proposes a deep model with a semi-supervised architecture, which simultaneously optimizes the first-order and second-order proximity. (Chang et al. 2015) employs deep model to transfer different objects in heterogeneous networks to unified vector representations.

However, all of the above methods assume pairwise relationships among objects in real world networks. In view of the aforementioned facts, a series of methods (Zhou, Huang, and Schölkopf 2006; Liu et al. 2013; Wu, Han, and Zhuang 2010) is proposed by generalizing spectral clustering techniques (Ng et al. 2001) to hypergraphs. Nevertheless, these methods focus on homogeneous hypergraph. They construct hyperedge by latent similarity like common label and preserve hyperedge implicitly. Therefore, they cannot preserve the structure of indecomposable hyperedges. For heterogeneous hyper-network, the tensor decomposition (Kolda and Bader 2009; Rendle and Schmidt-Thieme 2010; Symeonidis 2016) may be directly applied to learn the embedding. Unfortunately, the time cost of tensor decomposition is usually very expensive so it cannot scale efficiently to large network. Besides, HyperEdge Based Embedding (HEBE) (Gui et al. 2016) is proposed to model the proximity among participating objects in each heterogeneous event as a hyperedge based on prediction. It does not take high-order network structure and high degree of sparsity into account which affects the predictive performance in our task.

Table 1: Notations.

Symbols	Meaning
T	number of node types
$\mathbf{V} = \{\mathbf{V}_t\}_{t=1}^{T}$	node set
$\mathbf{E} = \{(v_1, v_2, ..., v_{n_i})\}$	hyperedge set
$\mathbf{A}$	adjacency matrix of hyper-network
$\mathbf{X}_i^j$	embedding of node i with type j
$\mathcal{S}(\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N)$	N-tuplewise similarity function
$\mathbf{W}_j^{(i)}$	the i-th layer weight matrix with type j
$\mathbf{b}_j^{(i)}$	the i-th layer biases with type j

Notations and Definitions

In this section, we define the problem of hyper-network embedding. The key notations used in this paper are shown in Table 1. First we give the definition of a hyper-network.

Definition 1 (Hyper-network)**.** *A **hyper-network** is defined as a hypergraph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with the set of nodes V belonging to T types $\mathbf{V} = \{\mathbf{V}_t\}_{t=1}^{T}$ and the set of edges $\mathbf{E}$ which may have more than two nodes $\mathbf{E} = \{\mathrm{E}_i = (\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_{n_i})\}(n_i \geq 2)$. If the number of nodes is 2 for each hyperedge, the hyper-network degenerates to a network. The type of edge E_i is defined as the combination of types of nodes belonging to the edge. If $T \geq 2$, the hyper-network is defined as a **heterogeneous hyper-network**.*

To obtain the embeddings in a hyper-network, the indecomposable tuplewise relationships need be preserved. We define the indecomposable structures as the first-order proximity of hyper-network:

Definition 2 (The First-order Proximity of Hyper-network)**.** ***The first-order proximity** of hyper-network measures the N-tuplewise similarity between nodes. For any N vertexes $\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_N$, if there exists a hyperedge among these N vertexes, the first-order proximity of these N vertexes is defined as 1, but this implies no first-order proximity for any subsets of these N vertexes.*

The first-order proximity implies the indecomposable similarity of several entities in real world. Meanwhile, real world networks are always incomplete and sparse. Only considering first-order proximity is not sufficient for learning node embeddings. Higher order proximity needs to be considered to fix this issue. We then introduce the second-order proximity of hyper-network to capture the global structure.

Definition 3 (The Second-order Proximity of Hyper-network)**.** ***The second-order Proximity** of hyper-network measures the proximity of two nodes with respect to their neighborhood structures. For any node $\mathrm{v}_i \in \mathrm{E}_i$, $\mathrm{E}_i/\mathrm{v}_i$ is defined as a **neighborhood** of v_i. If v_i's neighborhoods $\{\mathrm{E}_i/\mathrm{v}_i \ for \ any \ \mathrm{v}_i \in \mathrm{E}_i\}$ are similar to v_j's, then v_i's embedding $\mathbf{x}_i$ should be similar to v_j's embedding $\mathbf{x}_j$.*

For example, in Figure 1(a), A_1's neighborhoods set is $\{(L_2, U_1), (L_1, U_2)\}$. A_1 and A_2 have second-order similarity since they have common neighborhood, (L_1, U_2).

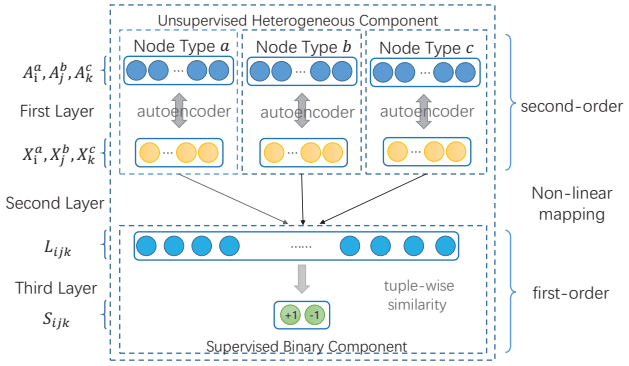

Figure 2: Framework of Deep Hyper-Network Embedding.

Deep Hyper-Network Embedding

In this section, we introduce the proposed Deep Hyper-Network Embedding (DHNE). The framework is shown in Figure 2.

Loss function

To preserve the first-order proximity of a hyper-network, an N-tuplewise similarity measure in embedding space is required. If there exists a hyperedge among N vertexes, the N-tuplewise similarity of these vertexes should be large, and small otherwise.

Property 1. *We mark $\mathbf{X}_i$ as the embedding of node v_i and $\mathcal{S}$ as N-tuplewise similarity function.*

- *if $(\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_N) \in \mathbf{E}$, $\mathcal{S}(\mathbf{X}_1, \mathbf{X}_2, .., \mathbf{X}_N)$ should be large (without loss of generality, large than a threshold l).*
- *if $(\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_N) \notin \mathbf{E}$, $\mathcal{S}(\mathbf{X}_1, \mathbf{X}_2, .., \mathbf{X}_N)$ should be small (without loss of generality, smaller than a threshold s).*

In our model, we propose a data-dependent N-tuplewise similarity function. In this paper, we mainly focus on hyperedges with uniform length $N = 3$, but it is easy to extend to $N > 3$.

Here we provide the theorem to demonstrate that a linear tuplewise similarity function cannot satisfy Property 1.

Theorem 1. *Linear function $\mathcal{S}(\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N) = \sum_i \mathbf{W}_i \mathbf{X}_i$ cannot satisfy Property 1.*

Proof. To prove it by contradiction, we assume that theorem 1 is false, i.e., the linear function $\mathcal{S}$ satisfies Property 1. We suggest the following counter example. Assume we have 3 types of nodes, and each type of node has 2 clusters (0 and 1). There is a hyperedge if and only if 3 nodes from different types have the same cluster id. We use $\mathbf{Y}_i^j$ to represent embeddings of nodes with type j in cluster i. By Property 1, we have

$$\mathbf{W}_1\mathbf{Y}_0^1 + \mathbf{W}_2\mathbf{Y}_0^2 + \mathbf{W}_3\mathbf{Y}_0^3 > l \quad (1)$$

$$\mathbf{W}_1\mathbf{Y}_1^1 + \mathbf{W}_2\mathbf{Y}_0^2 + \mathbf{W}_3\mathbf{Y}_0^3 < s \quad (2)$$

$$\mathbf{W}_1\mathbf{Y}_1^1 + \mathbf{W}_2\mathbf{Y}_1^2 + \mathbf{W}_3\mathbf{Y}_1^3 > l \quad (3)$$

$$\mathbf{W}_1\mathbf{Y}_1^0 + \mathbf{W}_2\mathbf{Y}_1^2 + \mathbf{W}_3\mathbf{Y}_1^3 < s. \quad (4)$$

By combining Equation (1)(2)(3)(4), we get $\mathbf{W}_1 * (\mathbf{Y}_0^1 - \mathbf{Y}_1^1) > l - s$ and $\mathbf{W}_1 * (\mathbf{Y}_1^1 - \mathbf{Y}_0^1) > l - s$, which is a contradiction.

This completes the proof. □

Above theorem shows that N-tuplewise similarity function $\mathcal{S}$ should be in a non-linear form. This motivates us to model it by a multilayer perceptron. The multilayer perceptron is composed of two parts, which are shown separately in the second layer and third layer of Figure 2. The second layer is a fully connected layer with non-linear activation functions. With the input of the embeddings $(\mathbf{X}_i^a, \mathbf{X}_j^b, \mathbf{X}_k^c)$ of 3 nodes (v_i, v_j, v_k), we concatenate them and map them non-linearly to a common latent space $\mathbf{L}$. Their joint representation in latent space is shown as follows:

$$\mathbf{L}_{ijk} = \sigma(\mathbf{W}_a^{(2)} * \mathbf{X}_i^a + \mathbf{W}_b^{(2)} * \mathbf{X}_j^b + \mathbf{W}_c^{(2)} * \mathbf{X}_k^c + \mathbf{b}^{(2)}), \quad (5)$$

where σ is the sigmoid function.

After obtaining the latent representation $\mathbf{L}_{ijk}$, we finally map it to a probability space in the third layer to get the similarity:

$$\mathbf{S}_{ijk} \equiv \mathcal{S}(\mathbf{X}_i^a, \mathbf{X}_j^b, \mathbf{X}_k^c) = \sigma(\mathbf{W}^{(3)} * \mathbf{L}_{ijk} + \mathbf{b}^{(3)}). \quad (6)$$

Combining the aforementioned two layers, we obtain a non-linear tuplewise similarity measure function $\mathcal{S}$. In order to make this similarity function satisfy Property 1, we present the objective function as follows:

$$\mathcal{L}_1 = -(\mathbf{R}_{ijk} \log \mathbf{S}_{ijk} + (1 - \mathbf{R}_{ijk}) \log(1 - \mathbf{S}_{ijk})), \quad (7)$$

where $\mathbf{R}_{ijk}$ is defined as 1 if there is a hyperedge between v_i, v_j and v_k and 0 otherwise. From the objective function, it is easy to check that if $\mathbf{R}_{ijk}$ equals to 1, the similarity $\mathbf{S}_{ijk}$ should be large, and otherwise the similarity should be small. In other words, the first-order proximity is preserved.

Next, we consider to preserve the second-order proximity. The first layer of Figure 2 is designed to preserve the second-order proximity. Second-order proximity measures neighborhood structure similarity. Here, we define the adjacency matrix of hyper-network to capture the neighborhood structure. First, we give some basic definitions of hypergraph. For a hypergraph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, a $|\mathbf{V}| * |\mathbf{E}|$ incidence matrix $\mathbf{H}$ with entries $\mathbf{h}(v, e) = 1$ if $v \in e$ and 0 otherwise, is defined to represent the hypergraph. For a vertex $v \in \mathbf{V}$, the degree of vertex is defined by $d(v) = \sum_{e \in \mathbf{E}} \mathbf{h}(v, e)$. Let $\mathbf{D}_v$ denote the diagonal matrix containing the vertex degree. Then the adjacency matrix $\mathbf{A}$ of hypergraph $\mathbf{G}$ can be defined as $\mathbf{A} = \mathbf{H}\mathbf{H}^T - \mathbf{D}_v$, where $\mathbf{H}^T$ is the transpose of $\mathbf{H}$. The entries of adjacency matrix $\mathbf{A}$ denote the concurrent times between two nodes, and the i-th row of adjacency matrix $\mathbf{A}$ shows the neighborhood structure of vertex v_i. We use an adjacency matrix $\mathbf{A}$ as our input feature and an autoencoder (LeCun, Bengio, and Hinton 2015) as the model to preserve the neighborhood structure. The autoencoder is composed by an encoder and a decoder. The encoder is a non-linear mapping from feature space $\mathbf{A}$ to latent representation space $\mathbf{X}$ and the decoder is a non-linear mapping from latent representation $\mathbf{X}$ space back to origin feature space $\hat{\mathbf{A}}$, which is shown as follows:

$$\mathbf{X}_i = \sigma(\mathbf{W}^{(1)} * \mathbf{A}_i + \mathbf{b}^{(1)}) \quad (8)$$

$$\hat{\mathbf{A}}_i = \sigma(\hat{\mathbf{W}}^{(1)} * \mathbf{X}_i + \hat{\mathbf{b}}^{(1)}). \quad (9)$$

The goal of autoencoder is to minimize the reconstruction error between the input and the output. The autoencoder's reconstruction process will make the nodes with similar neighborhoods have similar latent representations, and thus the second-order proximity is preserved. It is noteworthy that the input feature is the adjacency matrix of the hyper-network, and the adjacency matrix is often extremely sparse. To speed up our model, we only reconstruct non-zero element in the adjacency matrix. The reconstruction error is shown as follows:

$$||sign(\mathbf{A}_i) \odot (\mathbf{A}_i - \hat{\mathbf{A}}_i)||_F^2, \quad (10)$$

where $sign$ is the sign function.

Furthermore, in hyper-networks, the vertexes often have various types, forming heterogeneous hyper-networks. Considering the special characteristics of different types of nodes, it is required to learn unique latent spaces for different node types. In our model, each heterogeneous type of entities have their own autoencoder model as shown in Figure 2. Then for all types of nodes, the loss function is defined as:

$$\mathcal{L}_2 = \sum_t ||sign(\mathbf{A}_i^t) \odot (\mathbf{A}_i^t - \hat{\mathbf{A}}_i^t)||_F^2, \quad (11)$$

where t is the index for node types.

To preserve both first-order proximity and second-order proximity of heterogeneous hyper-networks, we jointly minimize the objective function by combining Equation 7 and Equation 11:

$$\mathcal{L} = \mathcal{L}_1 + \alpha \mathcal{L}_2. \quad (12)$$

Optimization

We use stochastic gradient descent (SGD) to optimize the model. The key step is to calculate the partial derivative of the parameters $\theta = \{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}, \hat{\mathbf{W}}^{(i)}, \hat{\mathbf{b}}^{(i)}\}_{i=1}^3$. These derivatives can be easily estimated by using back-propagation algorithm (LeCun, Bengio, and Hinton 2015). Notice that there is only positive relationship in most real world network, so this algorithm may converge to a trivial solution where all tuplewise relationships are similar. To address this problem, we sample multiple negative edges based on noisy distribution for each edge, as in (Mikolov et al. 2013). The whole algorithm is shown in Algorithm 1.

Analysis

In this section, we present the out-of-sample extension and complexity analysis.

Algorithm 1 The Deep Hyper-Network Embedding (DHNE)

Require: the hyper-network $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with adjacency matrix $\mathbf{A}$, the parameter α

Ensure: Hyper-network Embeddings E and updated Parameters $\theta = \{\mathbf{W}^{(i)}, \mathbf{b}^{(i)}, \hat{\mathbf{W}}^{(i)}, \hat{\mathbf{b}}^{(i)}\}_{i=1}^{3}$

1: initial parameters θ by random process
2: **while** the value of objective function do not converge **do**
3: generate next batch from the hyperedge set $\mathbf{E}$
4: sample negative hyperedge randomly
5: calculate partial derivative $\partial\mathcal{L}/\partial\theta$ with back-propagation algorithm to update θ.
6: **end while**

Out-of-sample extension For a newly arrived vertex v, we can easily obtain its adjacency vector by its connections to existing vertexes. We feed its adjacency vector into the specific autoencoder corresponding to its type, and apply Equation 8 to get representation for vertex v. The complexity for such steps is $\mathcal{O}(d_v d)$, where d_v is the degree of vertex v and d is the dimensionality of the embedding space..

Complexity analysis During the training procedure, the time complexity of calculating gradients and updating parameters is $O((nd + dl + l)bI)$, where n is the number of nodes, d is the dimension of embedding vectors, l is the size of latent layer, b is the batch size and I is the number of iterations. Parameter l is usually related to the dimension of embedding vectors d but independent with the number of vertexes n. The batch size is normally a small number. The number of iterations is also not related with the number of vertexes n. Therefore, the complexity of training procedure is linear to the number of vertexes.

Experiment

In this section, we evaluate our proposed method on several real world datasets and multiple application scenarios.

Datasets

In order to comprehensively evaluate the effectiveness of our proposed method, we use four different types of datasets, including a GPS network, a social network, a medicine network and a semantic network. The detailed information is shown as follows.

- GPS (Zheng et al. 2010): The dataset describes a user joins in an activity in certain location. The (user, location, activity) relations are used for building the hyper-network.
- MovieLens (Harper and Konstan 2016): This dataset describes personal tagging activity from MovieLens[1]. Each movie is labeled by at least one genres. The (user, movie, tag) relations are considered as the hyperedges to form a hyper-network.

[1]https://movielens.org/

Table 2: Statistics of the datasets.

datasets	node type			#(V)			#(E)
GPS	user	location	activity	146	70	5	1436
MovieLens	user	movie	tag	2113	5908	9079	47957
drug	user	drug	reaction	12	1076	6398	171756
wordnet	head	relation	tail	40504	18	40551	145966

- drug[2]: This dataset is obtained from FDA Adverse Event Reporting System (FAERS). It contains information on adverse event and medication error reports submitted to FDA. We construct hyper-network by (user, drug, reaction) relationships, i.e., a user who has certain reaction and takes some drugs will lead to adverse event.
- wordnet (Bordes et al. 2013): This dataset consists of a collection of triplets (synset, relation type, synset) extracted from WordNet 3.0. We can construct the hyper-network by regarding head entity, relation, tail entity as three types of nodes and the triplet relationships as hyperedges.

The detailed statistics of the datasets are summarized in Table 2.

Parameter Settings

We compared DHNE against the following six widely-used algorithms: DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), LINE (Tang et al. 2015), node2vec (Grover and Leskovec 2016), Spectral Hypergraph Embedding (SHE) (Zhou, Huang, and Schölkopf 2006), Tensor decomposition (Kolda and Bader 2009) and HyperEdge Based Embedding (HEBE) (Gui et al. 2016).

In summary, DeepWalk, LINE and node2vec are conventional pairwise network embedding methods. In our experiment, we use clique expansion in Figure 1(c) to transform a hyper-network in a conventional network, and then use these three methods to learn node embeddings from the conventional networks. SHE is designed for homogeneous hyper-network embeddings. Tensor method is a direct way for preserving high-order relationship in heterogeneous hyper-network. HEBE learns node embeddings for heterogeneous event data. Note that DeepWalk, LINE , node2vec and SHE can only measure pairwise relationship. In order to make them applicable to network reconstruction and link prediction in hyper-networks, without loss of generality, we use the mean or minimum value among all pairwise similarities in a candidate hyperedge to represent the tuplewise similarity of the hyperedge. For DeepWalk and node2vec, we set window size as 10, walk length as 40, walks per vertex as 10. For LINE, we set the number of negative samples as 5.

We uniformly set the representation size as 64 for all methods. Specifically, for DeepWalk and node2vec, we set window size as 10, walk length as 40, walks per vertex as 10. For LINE, we set the number of negative samples as 5.

For our model, we use one-layer autoencoder to preserve hyper-network structure and one-layer fully connected layer to learn tuplewise similarity function. The size of hidden

[2]http://www.fda.gov/Drugs/

Table 3: AUC value for network reconstruction.

methods		GPS	MovieLens	drug	wordnet
DHNE		**0.9598**	**0.9344**	**0.9356**	**0.9073**
mean	deepwalk	0.6714	0.8233	0.5750	0.8176
	line	0.8058	0.8431	0.6908	0.8365
	node2vec	0.6715	0.9142	0.6694	0.8609
	SHE	0.8596	0.7530	0.5486	0.5618
min	deepwalk	0.6034	0.7117	0.5321	0.7423
	line	0.7369	0.7910	0.7625	0.7751
	node2vec	0.6578	0.9100	0.6557	0.8387
	SHE	0.7981	0.7972	0.6236	0.5918
tensor		0.9229	0.8640	0.7025	0.7771
HEBE		0.9337	0.8772	0.8236	0.7391

layer of autoencoder is set as 64 which is also the representation size. The size of fully connect layer is set as sum of the embedding length from all types, 192. We do grid search from $\{0.01, 0.1, 1, 2, 5, 10\}$ to tune the parameter α which is shown in *Parameter Sensitivity* section. Similar to LINE (Tang et al. 2015), the learning rate is set with the starting value $\rho_0 = 0.025$ and decreased linearly with the times of iterations.

Network Reconstruction

A good network embedding method should preserve the original network structure well in the embedding space. We first evaluate our proposed algorithm on network reconstruction task. We use the learned embeddings to predict the links of origin networks. The AUC (Area Under the Curve) (Fawcett 2006) is used as the evaluation metric. The results are show in Table 3.

From the results, we have the following observations:

- Our method achieves significant improvements on AUC values over the baselines on all four datasets. It demonstrates that our method is able to preserve the origin network structure well.
- Compared with the baselines, our method achieves higher improvements in sparse drug and wordnet datasets than those on GPS and MovieLens datasets. It indicates the robustness of proposed methods on sparse datasets.
- The results of DHNE perform better than DeepWalk, LINE and SHE which assume that high-order relationships are decomposable. It demonstrates the importance of preserving indecomposable hyperedge in hyper-network.

Link Prediction

Link prediction is a widely-used application in real world especially in recommendation systems. In this section, we fulfil two link prediction tasks on all the four datasets. The two tasks evaluate the overall performance and the performance with different sparsity of the networks, respectively. We calculate AUC value as in network reconstruction task to evaluate the performance.

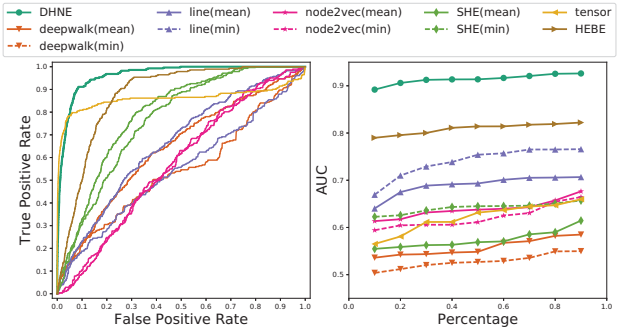

Figure 3: left: ROC curve on GPS; right: Performance for link prediction on networks of different sparsity.

For the first task, we randomly hide 20 percentage of existing edges and use the left network to train hyper-network embedding models. After training, we obtain embedding for each node and similarity function for N nodes and apply the similarity function to predict the held-out links. For GPS dataset which is a small and dense dataset, we can draw ROC curve for this task to observe the performance at various threshold settings, as shown in Figure 3 left. The AUC results on all datasets are shown in Table 4. The observations are illustrated as follows:

- Our method achieves significant improvements over the baselines on all the datasets. It demonstrates the learned embeddings of our method have strong predictive power for unseen links.
- By comparing the performance of LINE, DeepWalk, SHE and DHNE, we can observe that transforming the indecomposable high-order relationship into multiple pairwise relationship will damage the predictive power of the learned embeddings.
- As Tensor and HEBE can somewhat address the indecomposibility of hyperedges, the large improvement margin of our method over these two methods clearly demonstrates the importance of second-order proximities in hyper-network embedding.

For the second task, we change the sparsity of network by randomly hiding different ratios of existing edges and repeat the previous task. Particularly, we conduct this task on the drug dataset as it has the most hyperedges. The ratio of remained edges is selected from 10% to 90%. The results are shown in Figure 3 right.

We can observe that DHNE has a significant improvements over the best baselines on all sparsity of networks. It demonstrates the effectiveness of DHNE on sparse networks.

Classification

In this section, we conduct multi-label classification (Bhatia et al. 2015) on MovieLens dataset and multi-class classification in wordnet, because only these two datasets have label or category information. After deriving the node embeddings from different methods, we choose SVM as the

Table 4: AUC value for link prediction.

methods		GPS	MovieLens	drug	wordnet
DHNE		**0.9166**	**0.8676**	**0.9254**	**0.8268**
mean	deepwalk	0.6593	0.7151	0.5822	0.5952
	line	0.7795	0.7170	0.7057	0.6819
	node2vec	0.5835	0.8211	0.6573	0.8003
	SHE	0.8687	0.7459	0.5899	0.5426
min	deepwalk	0.5715	0.6307	0.5493	0.5542
	line	0.7219	0.6265	0.7651	0.6225
	node2vec	0.5869	0.7675	0.6546	0.7985
	SHE	0.8078	0.8012	0.6508	0.5507
tensor		0.8646	0.7201	0.6470	0.6516
HEBE		0.8355	0.7740	0.8191	0.6364

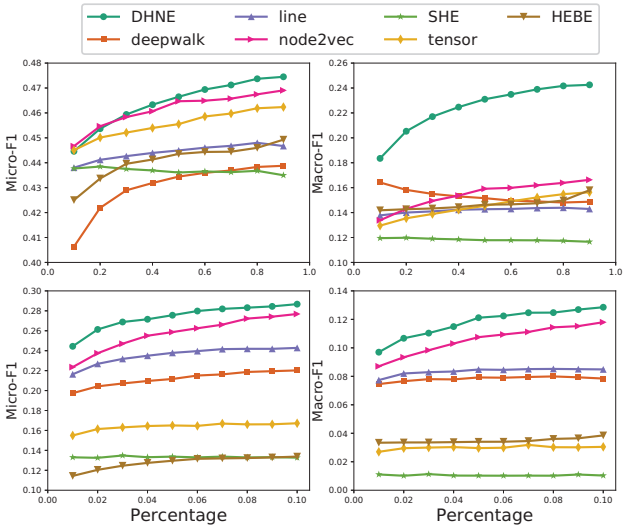

Figure 4: top: multi-label classification on MovieLens dataset; bottom: multi-class classification on wordnet dataset.

classifier. For MovieLens dataset, we randomly sample 10% to 90% of the vertexes as the training samples and use the left vertexes to test the performance. For wordnet dataset, the portion of training data is selected from 1% to 10%. Besides, we remove the nodes without labels on these two datasets. Averaged Macro-F1 and Micro-F1 are used to evaluate the performance. The results are shown in Figure 4.

From the results, we have following observations:

- In both Micro-F1 and Macro-F1 curves, our method performs consistently better than baselines. It demonstrates the effectiveness of our proposed method in classification tasks.
- When the labelled data becomes richer, the relative improvement of our method is more obvious than baselines. Besides, as shown in Figure 4 bottom, when the labeled data is quite sparse, our method still outperforms the baselines. This demonstrates the robustness of our method.

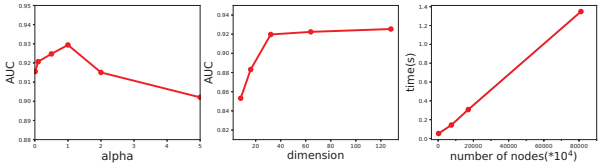

Figure 5: left, middle: Parameter w.r.t. embedding dimensions d, the value of α; right: training time per batch w.r.t. embedding dimensions.

Parameter Sensitivity

In this section, we investigate how parameters influence the performance and training time. Especially, we evaluate the effect of the ratio of first-order proximity loss and second-order proximity loss α and the embedding dimension d. For brevity, we report the results by link prediction task with drug dataset.

Effect of embedding dimension We show how the dimension of embedding space affects the performance in Figure 5 left. We can see that the performance raises firstly when the number of embedding dimension increases. This is reasonable because higher embedding dimensions can embody more information of a hyper-network. After the embedding dimension is larger than 32, the curve is relatively stable, demonstrating that our algorithm is not very sensitive to embedding dimension.

Effect of parameter α The parameter α measures the trade-off of the first-order proximity and second-order proximity of a hyper-network. We show how the values of α affect the performance in Figure 5 middle. When α equals 0, only the first-order proximity is taken into account in our method. The performance with α between 0.1 and 2 is better than that of $\alpha = 0$, demonstrating the importance of second-order proximity. The fact that the performance with α between 0.1 and 2 is better than that of $\alpha = 5$ can demonstrate the importance of the first-order proximity. In summary, both the first-order proximity and the second-order proximity are necessary for hyper-network embedding.

Training time analysis To testify the scalability, we test training time per batch, as shown in Figure 5 right. We can observe that the training time scales linearly with the number of nodes. This results conform to the above complexity analysis and indicate the scalability of our model.

Conclusion

In this paper, we propose a novel deep model named DHNE to learn the low-dimensional representation for hyper-networks with indecomposible hyperedges. More specifically, we theoretically prove that any linear similarity metric in embedding space commonly used in existing methods cannot maintain the indecomposibility property in hyper-networks, and thus propose a new deep model to realize a non-linear tuplewise similarity function while preserving both local and global proximities in the formed embedding space. We conduct extensive experiments on four different

types of hyper-networks, including a GPS network, an online social network, a drug network and a semantic network. The empirical results demonstrate that our method can significantly and consistently outperform the state-of-the-art algorithms.

Acknowledgments

This work is supported by National Program on Key Basic Research Project No. 2015CB352300, National Natural Science Foundation of China Major Project No. U1611461; National Natural Science Foundation of China No. 61772304, 61521002, 61531006, 61702296; NSF IIS-1650723 and IIS-1716432. Thanks for the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and the Young Elite Scientist Sponsorship Program by CAST. Peng Cui, Xiao Wang and Wenwu Zhu are the corresponding authors. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Agarwal, S.; Branson, K.; and Belongie, S. 2006. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, 17–24. ACM.

Belkin, M., and Niyogi, P. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, 585–591.

Bhatia, K.; Jain, H.; Kar, P.; Varma, M.; and Jain, P. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, 730–738.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Chang, S.; Han, W.; Tang, J.; Qi, G.-J.; Aggarwal, C. C.; and Huang, T. S. 2015. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 119–128. ACM.

Deng, C.; Tang, X.; Yan, J.; Liu, W.; and Gao, X. 2016. Discriminative dictionary learning with common label alignment for cross-modal retrieval. *IEEE Transactions on Multimedia* 18(2):208–218.

Fawcett, T. 2006. An introduction to roc analysis. *Pattern recognition letters* 27(8):861–874.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. ACM.

Gui, H.; Liu, J.; Tao, F.; Jiang, M.; Norick, B.; and Han, J. 2016. Large-scale embedding learning in heterogeneous event data. In *ICDM*.

Harper, F. M., and Konstan, J. A. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5(4):19.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

Liu, Y.; Shao, J.; Xiao, J.; Wu, F.; and Zhuang, Y. 2013. Hypergraph spectral hashing for image retrieval with heterogeneous social contexts. *Neurocomputing* 119:49–58.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Ng, A. Y.; Jordan, M. I.; Weiss, Y.; et al. 2001. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, 849–856.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *Proceedings of the 33rd annual international conference on machine learning. ACM*.

Ou, M.; Cui, P.; Wang, F.; Wang, J.; and Zhu, W. 2015. Non-transitive hashing with latent similarity components. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 895–904. ACM.

Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric transitivity preserving graph embedding. In *KDD*, 1105–1114.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Rendle, S., and Schmidt-Thieme, L. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, 81–90. ACM.

Roweis, S. T., and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290(5500):2323–2326.

Sun, L.; Ji, S.; and Ye, J. 2008. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 668–676. ACM.

Symeonidis, P. 2016. Matrix and tensor decomposition in recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 429–430. ACM.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077. ACM.

Tenenbaum, J. B.; De Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290(5500):2319–2323.

Wang, X.; Cui, P.; Wang, J.; Pei, J.; Zhu, W.; and Yang, S. 2017. Community preserving network embedding.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1225–1234. ACM.

Wu, F.; Han, Y.-H.; and Zhuang, Y.-T. 2010. Multiple hypergraph clustering of web images by mining word2image correlations. *Journal of Computer Science and Technology* 25(4):750–760.

Zheng, V. W.; Cao, B.; Zheng, Y.; Xie, X.; and Yang, Q. 2010. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, volume 10, 236–241.

Zhou, D.; Huang, J.; and Schölkopf, B. 2006. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, volume 19, 1633–1640.