Multi-Dimensional Network Embedding with Hierarchical Structure

Yao Ma*†
Data Science and Engineering Lab
Michigan State University
mayao4@msu.edu.com

Zhaochun Ren†
Data Science Lab
JD.com
renzhaochun@jd.com

Ziheng Jiang
Data Science Lab
JD.com
jiangziheng@jd.com

Jiliang Tang
Data Science and Engineering Lab
Michigan State University
tangjili@msu.edu.com

Dawei Yin‡
Data Science Lab
JD.com
yindawei@acm.org

ABSTRACT

Information networks are ubiquitous in many applications. A popular way to facilitate the information in a network is to embed the network structure into low-dimension spaces where each node is represented as a vector. The learned representations have been proven to advance various network analysis tasks such as link prediction and node classification. The majority of existing embedding algorithms are designed for the networks with one type of nodes and one dimension of relations among nodes. However, many networks in the real-world complex systems have multiple types of nodes and multiple dimensions of relations. For example, an e-commerce network can have users and items, and items can be viewed or purchased by users, corresponding to two dimensions of relations. In addition, some types of nodes can present hierarchical structure. For example, authors in publication networks are associated to affiliations; and items in e-commerce networks belong to categories. Most of existing methods cannot be naturally applicable to these networks. In this paper, we aim to learn representations for networks with multiple dimensions and hierarchical structure. In particular, we provide an approach to capture independent information from each dimension and dependent information across dimensions and propose a framework MINES, which performs **M**ulti-d**I**mension **N**etwork **E**mbedding with hierarchical **S**tructure. Experimental results on a network from a real-world e-commerce website demonstrate the effectiveness of the proposed framework.

KEYWORDS

Network Embedding, Multi-dimensional Networks, Hierarchical Structure

ACM Reference Format:
Yao Ma, Zhaochun Ren, Ziheng Jiang, Jiliang Tang, and Dawei Yin. 2018. Multi-Dimensional Network Embedding with Hierarchical Structure. In *WSDM 2018: WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining , February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We are living in a connected world where information networks are ubiquitous. Some examples of information networks include social networks, publication networks, the World Wide Web and e-commerce networks. Network embedding, aiming to learn vector representations for nodes, has attracted increasing attention in recent years. Many advanced network embedding algorithms have emerged such as Deepwalk [26], LINE [29] and Metapath2vec [11], which have been proven to help numerous network analysis tasks such as link prediction [18], node classification [5][33] and network visualization [21][28].

Most of existing embedding algorithms are designed for networks with one type of nodes and one dimension of relations among nodes. However, many networks in real-world complex systems contain multiple dimensions of relations among nodes. For example, in social networking sites such as Facebook, two users could be connected by friend relations, and via various social interactions; in the transportation network [3], two cities could be connected via various means of transportations such as train, highway and airplane; while in e-commerce networks, items can be viewed and purchased by users, corresponding to two dimensions of relations between users and items. In addition, some of the nodes can present certain hierarchical structure. For example, in publication networks, authors are associated to affiliations; while in e-commerce networks, items are organized by categories. A typical example of multi-dimensional networks with hierarchical structure is illustrated in Figure 1 where there are two types of nodes $\mathcal{U} = \{u_1, u_2, u_3, u_4\}$ and $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, and $C = \{c_1, c_2, c_3\}$ is the set of parent nodes. The relations of nodes in $\mathcal{U}$, nodes in $\mathcal{T}$ and nodes between $\mathcal{U}$ and $\mathcal{T}$ are two-dimensional; while each node in $\mathcal{U}$ is associated to one parent in C. The vast majority of

*Work performed during an internship at Data Science Lab, JD.com.
†These two authors contributed equally.
‡Corresponding author

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

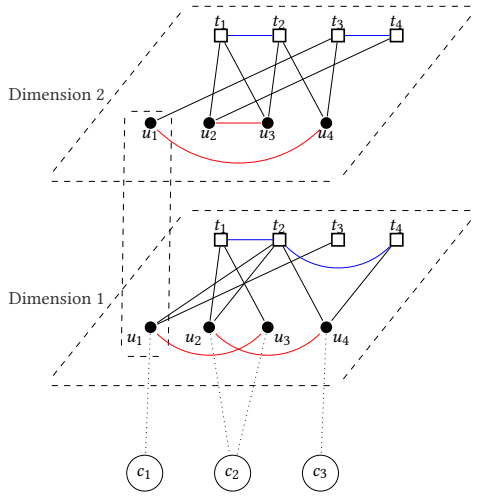

Figure 1: An illustrative example of a multi-dimensional network with hierarchical structure

existing embedding algorithms cannot be naturally applicable to multi-dimensional networks with hierarchical structure as shown in Figure 1.

In this paper, we aim to learn representations of nodes in networks with multiple dimensions and hierarchical structure. In particular, we study approaches (1) to mathematically capture multi-dimensional information and hierarchical structure; and (2) to incorporate such information simultaneously for embedding. Consequently, we propose a framework MINES for **M**ulti-d**I**mensional **N**etwork **E**mbedding with hierarchical **S**tructure. Our major contributions are summarized as follows:

- Providing a principled approach to model multi-dimensional networks, which can capture independent information from each dimension and dependent information across dimensions;
- Proposing a framework MINES, which incorporates multi-dimensional relations and hierarchical structure into a coherent model for node representation learning; and
- Validating the effectiveness of the proposed framework in a real-world e-commerce network.

The rest of this paper is arranged as follows. In Section 2, we review some works that are related to our problem. The problem of embedding networks with multiple dimensions and hierarchical structure to vector space is formally defined in Section 3. The approach to model networks with multiple dimensions and hierarchical structure and the proposed framework with an optimization method are introduced in Section 4. The experiments on a real-world e-commerce network with discussions are presented in Section 5. The conclusion and future work are presented in Section 6.

2 RELATED WORK

Our work is related to multi-dimensional network analysis and network embedding. In this section, we briefly review them.

2.1 Multi-dimensional Network Analysis

Network analysis has been extensively studied for many years [34][35][6][14][2][7]. Multidimensional networks, which are quite ubiquitous in the real-world applications, have attracted increasing attention. In [3], the authors introduced a few examples of real-world multidimensional networks, and they also defined measures such as degree, neighbors for the multidimensional networks. More measures for the multidimensional networks are introduced in [22]. The classic link prediction problem has been extended to multidimensional networks with the new problem "what is the probability that a new link between two nodes will form in a specific dimension?" [27]. Multidimensional versions of the Common Neighbors and Adamic-Adar have been introduced to solve this problem [27]. In [4], the authors studied the community discovery problem in the multidimensional network setting. In [16], the authors investigated friendship maintenance and prediction in multidimensional social networks.

2.2 Network Embedding

Networks can be represented by adjacency matrices; however, these representations are too sparse and high-dimensional. Many classic methods such as Laplacian eigenmap [1] and IsoMap [30] have been proposed to learn low-dimensional representations. These methods work fine on small size networks but cannot be scaled to very large networks. Inspired by word2vec [23][25], DeepWalk and LINE are proposed recently which can be applied to very large scale networks. DeepWalk regards the nodes in the network as the "words" of an artificial language and uses random walk to generate the "sentences" for this language. Then, following the procedure of word2vec, the representations for the nodes can be learned. LINE tries to capture both the first order and second order proximity in the representations. node2vec [13] extends DeepWalk by adding parameters to introduce the biased random walk. These network embedding methods have shown effectiveness in various tasks on many homogeneous networks. In [11], the authors extended DeepWalk method to heterogeneous networks by introducing meta-path based random walks. In [9], the authors also facilitate the meta-path to learn the heterogeneous network embedding, while they focus on the selection of the meta-path. In [8], the authors facilitate deep architectures to perform heterogeneous network embedding. In [32], a signed network embedding algorithm SiNE is proposed based on the notion that a user should be closer to their "friend" than their "enemy". In [20], the authors try to preserve both local and global information in the network for network embedding. There are also works on attributed network embedding [17][31]. Two recent surveys [10][12] give a comprehensive overview of network embedding algorithms. However, most of the existing methods cannot naturally be applicable to networks with multiple dimensions of relations and hierarchical structure. In this paper, we aim to model the multi-dimensional relations and hierarchical structure and propose a framework to embed these networks to vector space.

3 PROBLEM STATEMENT

In the multi-dimensional networks, we have different types of nodes and multiple dimensions of relations. Assume that there are K types

of nodes in total and let $\mathcal{V}_i = \{v_1^{(i)}, v_2^{(i)}, \ldots, v_{N_i}^{(i)}\}$ be the set of the i-th type with N_i nodes. Let $\mathcal{V}$ denote the set of all the nodes $\mathcal{V} = \bigcup_{i=1}^{K} \mathcal{V}_i$. Some types of nodes in the network might present hierarchical structure. In other words, these nodes are associated with categories. For simplicity, we assume all the types of nodes have hierarchical structures with a depth of 2, and we name the parent nodes as categories in this case. Note that though in this work, we focus on the hierarchical structures with a depth of 2, it is straightforward to apply the proposed framework for deeper hierarchical structures. We set $C_i = \{c_1^{(i)}, c_2^{(i)}, \ldots, c_{M_i}^{(i)}\}$ as the set of M_i categories for the i-th type of nodes, and set $\mathbf{T^{(i)}} \in \mathbb{R}^{N_i \times Mi}$ as the matrix that describes the category information, for the i-th type of nodes.

Two nodes could be connected via multiple relations, and we regard each type of relations as a dimension. Thus, nodes from the same type or different types can be connected in the same dimension. These connections can be described by adjacency matrices (for the same type nodes) and the interaction matrices (for different types of nodes). Let $\mathbf{A_d^{(i)}} \in \mathbb{R}^{N_i \times N_i}$ be the adjacency matrix of the i-th node type and $\mathbf{H_d^{(i,j)}} \in \mathbb{R}^{\mathbf{N_i} \times \mathbf{N_j}}$ be the interaction matrix between the i-th and j-th types of node in the d-th dimension. We target to learn representations for each node in each dimension of the network. Let $\mathcal{U}_d{}^{(i)} = \{\mathbf{u_d}_1^{(i)}, \mathbf{u_d}_2^{(i)}, \ldots, \mathbf{u_d}_{N_i}^{(i)}\}$ denote the representations of i-th type of nodes in the dimension d ($d = 1, \ldots D$) where D is the number of dimensions.

With the aforementioned notations and definitions, our problem can be formally defined as follow:

Given

- *K different sets of nodes, i.e., $\mathcal{V}_i = \{v_1^{(i)}, v_2^{(i)}, \ldots, v_{N_i}^{(i)}\}$ ($i = 1, \ldots, K$);*
- *multi-dimensional relations among the nodes, i.e., $\mathbf{A}_d^{(i)}$ ($i = 1, \ldots, K$) and $\mathbf{H}_d^{(i,j)}$ ($i, j = 1, \ldots, K; i \neq j; d = 1, \ldots, D$);*
- *the hierarchical structure information, i.e., $\mathbf{T_i}$ ($i = 1, \ldots, K$).*

We aim to learn a set of representations for all nodes, i.e.,

$$\mathcal{U}_d{}^{(i)} = \{\mathbf{u}_{d1}^{(i)}, \mathbf{u}_{d2}^{(i)}, \ldots, \mathbf{u}_{dN_i}^{(i)}\} \quad (i = 1, \ldots, K)$$

in each dimension d ($d = 1, \ldots D$).

4 THE MULTI-DIMENSIONAL EMBEDDING FRAMEWORK WITH HIERARCHICAL STRUCTURE

In this section, we will first introduce how to model multi-dimensional relations and hierarchical structure; and then discuss the proposed framework with an optimization method.

4.1 Capturing Multi-Dimensional Relations

In a multi-dimensional network, all dimensions share the same set of nodes, while having their own network structures in each dimension. A straightforward way to learn representations for each dimension is to perform the network embedding for each dimension, separately. This strategy treats each dimension independently and completely ignores information across dimensions. Hence the learned representations for different dimensions are not related. However, dimensions are inherently related since they share the same set of nodes. Thus, in this subsection, we study how to model multi-dimensional relations.

Intuitively, each dimension should have its independent information individually; while all dimensions should share dependent information across dimensions. Therefore, the learned representations for each dimension should not only preserve independent information from the dimension but also keep dependent information across dimensions. To achieve this goal, for a given dimension d, the representation $\mathbf{u}_d$ for a node contains two components – (1) one component $\mathbf{u}$ for the information shared across dimensions; and (2) one component $\mathbf{e}_d$ specific to the dimension d. With these two components, we can rewrite $\mathbf{u}_d$ as:

$$\mathbf{u}_d = f(\mathbf{u}, \mathbf{e}_d), \tag{1}$$

where f is a function to combine the shared component $\mathbf{u}$ and the specific component $\mathbf{e}_d$. The shared component $\mathbf{u}$ not only captures dependent information across dimensions but also helps the learned representations of all dimensions to be related. The specific component $\mathbf{e}_d$ preserves independent information from the dimension d.

4.2 Capturing Hierarchical Structure

For these nodes which have the hierarchical structure, we also need to model its category information (or parents). The category information is actually shared by all the dimensions, hence it should be indicated in the shared component of representations. For the nodes in the same category, they should also share the similar characteristics. Therefore, to model the hierarchical structure, the shared component of the node representation should further contain two components – (1) one component $\mathbf{c}_u$ indicates category information which is shared by all the nodes in the category, and (2) one component $\mathbf{s}_u$ is specific to the node. With the defined components, we can further rewrite $\mathbf{u}$ in Eq. (1) as:

$$\mathbf{u} = g(\mathbf{c}_u, \mathbf{s}_u) \tag{2}$$

where g is the function to combine the category shared information $\mathbf{c}_u$ and the node specific information $\mathbf{s}_u$. Note that Eq. (2) can be easily extended to deeper hierarchical structure by further decomposing the category shared information $\mathbf{c}_u$.

4.3 The Proposed Framework

With approaches to capture multi-dimensional relations and hierarchical structure, in this subsection, we introduce the embedding framework MINES.

To learn the embeddings for the nodes in each dimension, we follow the idea of skip-gram model [24], which is an effective and efficient way to learn distributed representations of words. The skip-gram model predicts surrounding context given a center word, which can be formulated as follows:

$$p(N(w_c)|w_c), \tag{3}$$

where $N(w_c)$ is the set of words that surround word w_c.

Similarly, we can use the skip-gram to model the network in a given dimension d. For a node v, we define all nodes connected to

v as the "context" of v, which is formally defined as:

$$N_d(v) = \bigcup_{i=1}^{K} N_d^{(i)}(v); \tag{4}$$

where $N_d^{(i)}$ is the set of i-th type of nodes that are connected to node v in the dimension d. Note that the "context" of v consists of different types of nodes, and we treat them differently, which will be further explained later.

Then, given a center node v, we need to predict its "context" as:

$$p_d(N_d(v)|v) = \prod_{i=1}^{K} p(N_d^{(i)}(v)|v) = \prod_{i=1}^{K} \prod_{v_j^{(i)} \in N_d^{(i)}(v)} p_d(v_j^{(i)}|v); \tag{5}$$

where $p(v_j^{(i)}|v)$ can be modeled using a softmax function as:

$$p_d(v_j^{(i)}|v) = \frac{\exp(\mathbf{u}_d^T \mathbf{u}_{d_j}^{(i)})}{\sum_{v^{(i)} \in V_i} \exp(\mathbf{u}_d^T \mathbf{u}_{d}^{(i)})}. \tag{6}$$

In (6), the softmax function is over the i-th type nodes $\mathcal{V}_i$ instead of the whole nodes set $\mathcal{V}$.

To learn the representations for the dimension d, we model this problem as a maximum likelihood problem. In other words, we need to maximize the probability that $N_d(v)$ is the "context" of node v for all the nodes $v \in \mathcal{V}$. Hence, we need to maximize:

$$P_d = \prod_{v \in \mathcal{V}} p_d(N_d(v)|v). \tag{7}$$

With all D dimensions, we need to jointly maximize the following term:

$$P = \prod_{d=1}^{D} P_d. \tag{8}$$

Instead of maximizing Eq. (8), we equivalently minimize its negative logarithm with respect to the representations $\mathcal{U}_d^{(i)}$ as:

$$\begin{aligned} &\min_{\{\mathcal{U}_d^{(i)}\}_{i=1,\dots,K}^{d=1,\dots,D}} -\log P \\ \Leftrightarrow &\min_{\{\mathcal{U}_d^{(i)}\}_{i=1,\dots,K}^{d=1,\dots,D}} -\sum_{d=1}^{D} \log P_d \\ \Leftrightarrow &\min_{\{\mathcal{U}_d^{(i)}\}_{i=1,\dots,K}^{d=1,\dots,D}} -\sum_{d=1}^{D} \sum_{v \in \mathcal{V}} \sum_{i=1}^{K} \sum_{v_j^{(i)} \in N_d^{(i)}(v)} \log p_d(v_j^{(i)}|v). \end{aligned} \tag{9}$$

4.4 An Optimization Method

There are two challenges to address when optimizing Eq. (9). First, the minimization of Eq. (9) is computationally expensive due to summation over the whole set of nodes $\mathcal{V}_i$ when calculating each term $\log p_d(v_j^{(i)}|v)$. Second, how to choose the functions of f in Eq. (1) and g in Eq. (2).

To solve computational challenge, we adopt the negative sampling approach proposed in [25]. By using the negative sampling method, we replace each $\log p_d(v_j^{(i)}|v)$ with

$$O_d(v, v_j^{(i)}) = \log \sigma(\mathbf{u}_d^T \mathbf{u}_{d_j}^{(i)}) + \sum_{n=1}^{N_e} \log \sigma(-\mathbf{u}_d^T \mathbf{u}_{d_{(n)}}^{(i)}); \tag{10}$$

where $\sigma(x) = 1/(1 + \exp(x))$ is the sigmoid function, and N_e is the number of negative samples. The negative samples are randomly sampled from some noise distribution. For $\log p_d(v_j^{(i)}|v)$, the negative samples are sampled from node set $\mathcal{V}_i$ according to $P_{i(v)}^{(i)}(v^{(i)}) \sim d_{v^{(i)}}^{3/4}$, as proposed in [25], where $d_{v^{(i)}}$ is the in-degree of $v^{(i)}$ corresponding to the $i(v)$-th type of nodes and $i(v)$ indicates the type of node v. Note again, for i-th type of node $v_j^{(i)}$, we sample the negative samples from the i-th type of nodes set $\mathcal{V}_i$ instead of the whole nodes set $\mathcal{V}$.

We adopt mini-batch Stochastic Gradient Descent (SGD) to optimize the problem. In each step, a mini-batch of edges of the same type are sampled according to their weights. Here, by "same" type of edge, we mean, these edges have same types of nodes for source and target nodes respectively and also the relations between them is in the same dimension. For each sampled edge, the source node is treated as v and the target node is treated as $v_j^{(i)}$ in (10). The derivatives for v, $v_j^{(i)}$ and $v_{(n)}^{(i)}$ are

$$\begin{aligned} \frac{\partial O_d(v, v_j^{(i)})}{\partial u_d} &= (1 - \sigma(u_d^T u_{d_j}^{(i)}))u_{d_j}^{(i)} - \sum_{n=1}^{Ne} (1 - \sigma(-u_d^T u_{d_{(n)}}^{(i)}))u_{d_{(n)}}^{(i)}; \\ \frac{\partial O_d(v, v_j^{(i)})}{\partial u_{d_j}^{(i)}} &= (1 - \sigma(u_d^T u_{d_j}^{(i)}))u_d; \\ \frac{\partial O_d(v, v_j^{(i)})}{\partial u_{d_{(n)}}^{(i)}} &= -(1 - \sigma(-u_d^T u_{d_{(n)}}^{(i)}))u_d, n = 1, \dots, Ne. \end{aligned} \tag{11}$$

Next we discuss how to choose f and g functions. In fact, f is used to combine the dimension shared component $\mathbf{u}$ and dimension-specific component $\mathbf{c}_d$. It can be a linear function, a non-linear function (e.g., exponential functions) or even can be automatically learned (e.g., neural networks). In this work, we choose a linear function f. In other words, we define as – $f(\mathbf{u}, \mathbf{e}_d) = \mathbf{u} + \mathbf{c}_d$. We also use a similar function for g. We would like to leave the investigation of other choices of f and g as one future direction. With choices of f and g, the representations for nodes can be rewritten as:

$$\mathbf{u}_d = \mathbf{c}_u + \mathbf{s}_u + \mathbf{e}_d; \tag{12}$$

$$\mathbf{u}_{d_j}^{(i)} = \mathbf{c}_{u_j}^{(i)} + \mathbf{s}_{u_j}^{(i)} + \mathbf{e}_{d_j}^{(i)}; \tag{13}$$

$$\mathbf{u}_{d_{(n)}}^{(i)} = \mathbf{c}_{u_{(n)}}^{(i)} + \mathbf{s}_{u_{(n)}}^{(i)} + \mathbf{e}_{d_{(n)}}^{(i)}. \tag{14}$$

We need to update $\mathbf{c}_u, \mathbf{s}_u, \mathbf{e}_d, \mathbf{c}_{uj}, \mathbf{s}_{uj}, \mathbf{e}_{dj}, \mathbf{c}_{u(n)}, \mathbf{s}_{u(n)}$ and $\mathbf{e}_{d(n)}$. We update these representations using Gradient Decent (GD).

To update the representations for v, we need to update its three components $\mathbf{c}_u, \mathbf{s}_u$ and $\mathbf{c}_d$ according to (15).

$$
\begin{aligned}
c_u &\leftarrow c_u + \rho \cdot \frac{\partial O_d(v, v_h^{(i)})}{\partial u_d};\\
s_u &\leftarrow s_u + \rho \cdot \frac{\partial O_d(v, v_h^{(i)})}{\partial u_d};\\
e_d &\leftarrow e_d + \rho \cdot \frac{\partial O_d(v, v_h^{(i)})}{\partial u_d}.
\end{aligned} \tag{15}
$$

Similarly, to update the representations for $v_j^{(i)}$, we need to update its three components $\mathbf{c}_{u_j}^{(i)}, \mathbf{s}_{u_j}^{(i)}$ and $\mathbf{c}_{d_j}^{(i)}$ according to (16).

$$
\begin{aligned}
c_{u_j}^{(i)} &\leftarrow c_{u_j}^{(i)} + \rho \cdot \frac{\partial O_d(v, v_j^{(i)})}{\partial u_{d_j}^{(i)}};\\
s_{u_j}^{(i)} &\leftarrow s_{u_j}^{(i)} + \rho \cdot \frac{\partial O_d(v, v_j^{(i)})}{\partial u_{d_j}^{(i)}};\\
e_{d_j}^{(i)} &\leftarrow e_{d_j}^{(i)} + \rho \cdot \frac{\partial O_d(v, v_j^{(i)})}{\partial u_{d_j}^{(i)}}.
\end{aligned} \tag{16}
$$

Finally, to update the representations for $v_{(n)}^{(i)},\ n = 1, \ldots, Ne$, we need to update their three components $\mathbf{c}_{u_{(n)}}^{(i)}, \mathbf{s}_{u_{(n)}}^{(i)}$ and $\mathbf{c}_{d_{(n)}}^{(i)}$ according to (17) respectively.

$$
\begin{aligned}
\mathbf{c}_{u_{(n)}}^{(i)} &\leftarrow \mathbf{c}_{u_{(n)}}^{(i)} + \rho \cdot \frac{\partial O_d(v, v_j^{(i)})}{\partial \mathbf{u}_{d_{(n)}}^{(i)}}, n = 1, \ldots Ne;\\
\mathbf{s}_{u_{(n)}}^{(i)} &\leftarrow \mathbf{s}_{u_{(n)}}^{(i)} + \rho \cdot \frac{\partial O_d(v, v_j^{(i)})}{\partial \mathbf{u}_{d_{(n)}}^{(i)}}, n = 1, \ldots Ne;\\
\mathbf{e}_{d_{(n)}}^{(i)} &\leftarrow \mathbf{e}_{d_{(n)}}^{(i)} + \rho \cdot \frac{\partial O_d(v, v_j^{(i)})}{\partial \mathbf{u}_{d_{(n)}}^{(i)}}, n = 1, \ldots Ne.
\end{aligned} \tag{17}
$$

We summarize the optimization procedure in Algorithm 1. In the algorithm, the input includes the number of mini-batch size m, the training size S, the dimension of representations dim, the number of negative samples Ne, the learning rate ρ and the set of all the edges $\mathcal{E}$ in the network. In line 1, we initialize all the components for all the representations. Then, we sample a set of same type edges SE from $\mathcal{E}$ in line 4. In line 6, for each edge, we sample Ne negative samples. We calculate the gradients and update the components in lines 7 and 8, respectively. Finally, we combine the components to form the representations for each node in each dimension in line 12.

To efficiently sample the edges and negatives samples, we adopt the alias methods proposed in [15], which can generate a random variable from a discrete distribution in constant time $O(1)$. The optimization with negative sampling takes $O(dim \cdot (Ne + 2) + Ne)$ time, where Ne is the number of negative samples. hence, each step of MINES takes $O(dim \cdot Ne)$ operations. If the training size is S, the overall time complexity of MINES is $O(S \cdot dim \cdot Ne)$.

Algorithm 1: Optimization procedure

Input: Ne, m, S, ρ, dim, $\mathcal{E}$
Output: $\{\mathcal{U}_d^{(i)}\}_{i=1,\ldots,K}^{d=1,\ldots,D}$

1 Initialize $\mathbf{c}_{u_j}^{(i)}, \mathbf{s}_{u_j}^{(i)}$ and $\mathbf{e}_{d_j}^{(i)}$, as dim dimension vectors randomly, for $d = 1, \ldots, D$, $i = 1, \ldots, K$ and $j = 1, \ldots, N_i$;
2 $s = 0$;
3 **while** $s < S$ **do**
4 Sample a set of m edges of the same type SE from $\mathcal{E}$;
5 **for** $e = (v, v_j^{(i)}) \in SE$ **do**
6 Sample a set of Ne negative samples $\{v_{(n)}^{(i)}\}_{n=1,\ldots,Ne}$;
7 Calculate the gradients according to (11);
8 Update the corresponding vectors according to (15), (16) and (17);
9 **end**
10 $s \leftarrow s + m$.
11 **end**
12 $\mathbf{u}_{d_j}^{(i)} = \mathbf{c}_{u_j}^{(i)} + \mathbf{s}_{u_j}^{(i)} + \mathbf{e}_{d_j}^{(i)}$; for $d = 1, \ldots, D$, $i = 1, \ldots, K$ and $j = 1, \ldots, N_i$;
13 **return** $\{\mathcal{U}_d^{(i)}\}_{i=1,\ldots,K}^{d=1,\ldots,D}$.

5 EXPERIMENTS

In this section, we present the experimental details to verify the effectiveness of the proposed framework. We first introduce the dataset we will use in the evaluation. Then, we describe the experimental settings. Finally, we present the experimental results with discussions and study the key parameter in the proposed framework.

5.1 Dataset

In our experiments, we sample data from JD.com, which is one of the largest e-commerce companies. In our dataset, we have two types of nodes: users and items. The items have hierarchical structure and each item belongs to some predefined categories. Users can perform various behaviors on items such as "view", "save", and "purchase". In this work, we collect two behaviors, i.e., "view" and "purchase", to construct two-dimensional relations between users and items. In addition, we collect two other relations: one is the "view session" of a user, while another is the "purchase basket" of the user.

A view session is a sequence of items that are viewed by a user within a period of time. It is intuitively to understand the items that are viewed within a short period by the same user should be similar. To incorporate these relations into the network, we construct an item-item view network by connecting the items that are viewed w items before or after a given item in a session with this item, where w is the window size. In this work, we set the window size to 5. These edges are in the "view" dimension and they are weighted where the weight is the co-occurrence frequency.

A purchase basket is a set of items that are purchased by a user at the same time. Items that are purchased in the same basket are supposed to be related to each other. To incorporate these relations, we construct an item-item purchase network. In particular, we connect two items if they are purchased in the same basket. These

# items	401,922
# users	17,806
# categories	2,788
# item-item (view)	6,402,586
# user-user (view)	13,651,206
# user-item (view)	962,362
# item-item (purchase)	3,211,660
# user-user (purchase)	6,870,510
# user-item (purchase)	485,656

Table 1: The statistics of the network

edges are weighted where the weight is the frequency of the two items presenting in the same basket. In the other way around, users that have "viewed" or "purchased" the same item also shows similarity. In each dimension, we connected users that have "viewed" or "purchased" the same items.

To sum up, in the constructed multidimensional e-commerce network, we have two types of nodes, i.e., the users and the items, and the items have hierarchical structure. There are two dimensions, i.e., the "view" dimension and the "purchase" dimension. We can conclude that the constructed network has all the characteristics of networks we want to study in this work; hence it is suitable for us to use the dataset to evaluate the proposed framework. Some statistics of the network are shown in Table 1.

5.2 Experimental Setting

Following the common way to assess network embedding algorithms [13], we choose link prediction as the evaluation task. The intuition is that a better embedding algorithm should learn better node representations, which will lead to better link prediction performance.

In the link prediction task, a certain fraction of edges are removed, and we would like to predict whether these "missing" edges exist.

In our evaluation, we perform the link prediction task on the two dimensions, separately. For each dimension, we remove the *user-item* edges and use them as parts of the testing set. We set up 3 groups of experiments, where 10%, 30% and 50% of the *user-item* edges are removed, respectively. To form the training set, we first put all the remaining *user-item* edges into the training set, and then, for each *user-item* edge in the training set, we randomly sample an item that is not connected to this user and use this user and non-connected item pair as the negative sample in the training set. We form the testing set in the same way.

After removing the edges, we use the remaining network to learn the representations for all the nodes. Then, to perform the link prediction task, the representations for the edges (or the user item pairs) should be learned. We use two different ways to combine the representations of two nodes as the representation of the edge (or user item pair) as used in [13].

- **Element-wise addition** Given two dim dimension representations of two nodes, we add them element-wisely and get a new dim dimension vector as the representation for this pair of nodes.
- **Element-wise multiplication** Given two dim dimension representations of two nodes, we multiply them element-wisely and get a new dim dimension vector as the representation for this pair of nodes.

For all the methods, we use both ways to form the representations for the pairs of nodes and report the results for each method.

After we form the representations for the pairs of the nodes in the training set and the testing set, we train a binary classifier using logistic regression on the training set and perform link prediction on the testing set. In this work, we will use Micro-F_1, Macro-F_1 and AUC as the metric to evaluate the link prediction performance.

5.3 Performance Comparison

To evaluate the performance of our algorithm, we compare the performance of our algorithm with the following representative baselines:

- **LINE** [29]: As LINE can only work for one-dimensional network, we apply LINE to the two dimensions separately and learn one set of representations for each dimension, respectively. We treat categories as nodes, and add *item-category* edges into the networks for LINE.
- **DeepWalk** [26]: We apply DeepWalk to the two dimensions separately and learn two sets of representations. We treat categories as nodes, and add *item-category* edges into the networks for DeepWalk. DeepWalk can only work for unweighted networks, hence, we convert our network to unweighted network by ignoring the weights.
- **Non-negative Matrix Factorization (NMF)** [19]: We apply it to the user-item interaction matrix and use the factorized two matrices as the embeddings for the users and items. NMF is also applied to the two dimensions, separately.
- **Co-NMF**: In Co-NMF, we perform a co-factorization on the multi-dimensional networks and learn unified user representations for all dimensions. Basically Co-NMF assumes all dimensions share the same embeddings, which completely ignores independent information from each dimension.
- **MINES(S)**: This is a variant of our framework MINES. Instead of using all three components $\mathbf{c}_u$, $\mathbf{s}_u$ and $\mathbf{e}_d$, we only use the shared components $\mathbf{c}_u$ and $\mathbf{s}_u$ to form the representation for a node v.

We summarize the experiments results for the "view" dimension and "purchase" dimension in Table 2, and Table 3, respectively. We make the following observations from Table 2:

- For all methods, using the **Element-wise multiplication** is better than **Element-wise addition**, which is consistent with the observation in [13].
- The performance of Co-NMF is worse than that of NMF, which indicates that the independent information from each dimension is very important to accurately predict links in that dimension.
- MINES shows better performance than MINES(S), which further shows the importance of the dimension specific information.
- As we remove more percent of edges, the performance of all methods decrease in all three measures when using the **Element-wise multiplication**.

		Addition			Multiplication		
	% removed edges	10%	30%	50%	10%	30%	50%
Micro-F_1(%)	MINES	**72.77**	**72.76**	**72.65**	**83.89**	**82.57**	**81.32**
	LINE	71.00	70.83	70.72	79.42	78.26	76.84
	DeepWalk	69.39	69.03	68.87	76.45	76.10	74.94
	NMF	59.80	59.79	59.86	78.16	78.16	77.98
	Co-NMF	56.66	56.83	56.93	76.96	77.00	76.87
	MINES(S)	67.22	66.78	66.82	76.98	76.28	75.78
Macro-F_1(%)	MINES	**72.74**	**72.72**	**72.58**	**83.75**	**82.35**	**80.98**
	LINE	70.94	70.82	70.78	79.17	77.95	76.44
	DeepWalk	69.28	68.92	68.79	76.22	75.79	74.63
	NMF	59.77	59.77	59.86	78.08	78.07	77.90
	Co-NMF	56.66	56.83	56.93	76.84	76.87	76.74
	MINES(S)	67.21	66.77	68.81	76.96	76.25	75.77
AUC	MINES	**0.8037**	**0.8040**	**0.8036**	**0.9261**	**0.9180**	**0.9146**
	LINE	0.7757	0.7757	0.7732	0.8879	0.8759	0.8636
	DeepWalk	0.7478	0.7434	0.7392	0.8522	0.8517	0.8351
	NMF	0.6516	0.6527	0.6529	0.8741	0.8739	0.8729
	Co-NMF	0.5912	0.5927	0.5933	0.8603	0.8605	0.8596
	MINES(S)	0.7315	0.7271	0.7278	0.8530	0.8456	0.8409

Table 2: Link Prediction Performance Comparison: The View Dimension

		Addition			Multiplication		
	% removed edges	10%	30%	50%	10%	30%	50%
Micro-F_1(%)	MINES	**78.48**	**77.94**	**77.62**	**90.93**	**89.74**	**88.57**
	LINE	71.55	70.84	70.63	87.88	86.94	85.69
	DeepWalk	67.08	67.46	67.53	82.76	82.35	80.54
	NMF	68.20	68.11	68.18	83.43	83.65	83.44
	Co-NMF	56.28	56.36	56.31	76.98	76.83	76.75
	MINES(S)	70.36	70.77	70.54	83.63	82.97	82.20
Macro-F_1(%)	MINES	**78.47**	**77.92**	**77.59**	**90.89**	**89.66**	**88.49**
	LINE	71.48	70.83	70.51	87.85	86.87	85.58
	DeepWalk	67.03	67.38	67.52	82.70	82.27	80.41
	NMF	68.19	68.10	68.18	83.36	83.59	83.36
	Co-NMF	56.24	56.33	56.27	76.84	76.67	76.60
	MINES(S)	70.34	70.75	70.52	83.61	82.96	82.19
AUC	MINES	**0.8662**	**0.8644**	**0.8623**	**0.9762**	**0.9725**	**0.9690**
	LINE	0.7892	0.7791	0.7759	0.9614	0.9518	0.9455
	DeepWalk	0.7109	0.7134	0.7128	0.9278	0.9256	0.9252
	NMF	0.7544	0.7528	0.7538	0.9346	0.9347	0.9342
	Co-NMF	0.5826	0.5834	0.5820	0.8592	0.8585	0.8568
	MINES(S)	0.7740	0.7775	0.7758	0.9146	0.9036	0.9101

Table 3: Link Prediction Performance Comparison: The Purchase Dimension

- The performance of DeepWalk is worse than LINE and NMF. This is mainly because DeepWalk can only work for unweighted networks and cannot take advantage of the edge weights.
- The proposed framework MINES obtains the best performance. For example, MINES boosts the performance 3% – 5% compared to the best baseline when 10% – 50% edges are removed. The major reason is the proposed framework has two components to capture the multi-dimensional relations and the hierarchical structure.

We have similar observations for the "purchase" dimension – (1) using the **Element-wise multiplication** is better than using

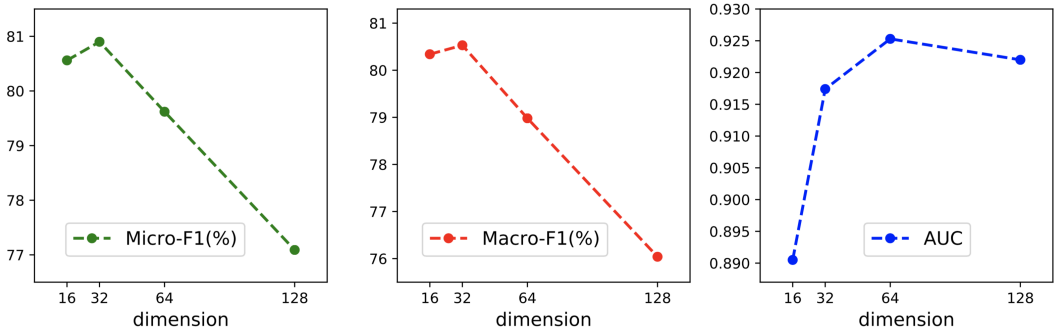

Figure 2: Parameter Analysis: The View Dimension

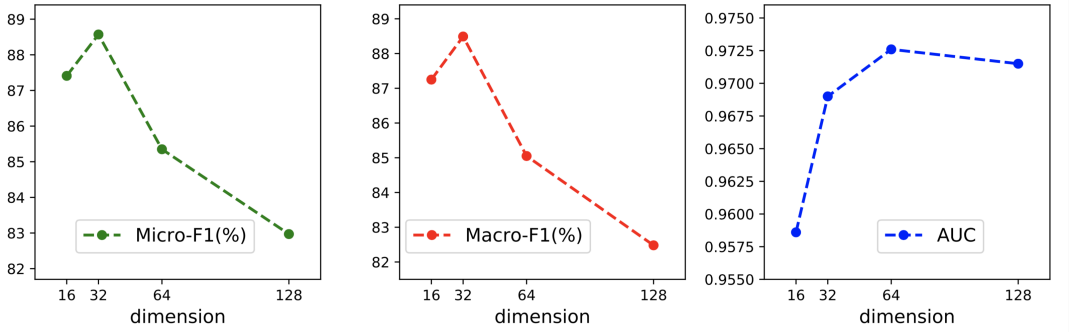

Figure 3: Parameter Analysis: The Purchase Dimension

Element-wise addition and (2) MINES outperforms all the baselines; for example, MINES obtains over 2% improvement in terms of all the measures compared to the best baseline.

5.4 Parameter Analysis

In this section, we analyze how the dimension of the learned representations in our method affects the performance of the link prediction task. In particular, we set the dimension of the representations to $\{16, 32, 64, 128\}$ with the setting of 50% edges removed. The results are reported in in Figure 2 and Figure 3 for view and purchase dimensions, separately. Note that we ignore the results with other settings since we can make similar observations.

As shown in Figure 2 and Figure 3, in both view and purchase dimensions, the Micro-F_1 and Macro-F_1 first increase as the dimension of the learned representations gets large, and then decrease. Both the Micro-F_1 and Macro-F_1 reach the maximum when the dimension of the representations is 32 in both view and purchase dimension. AUC also increases first and then decrease as the dimension of the representations gets larger. However, the AUC score reaches the maximum when the dimension of the representation is 64.

In summary, the performance first increases and then decreases as the dimension of the representations gets larger. The dimension of the representations affects the different measures differently.

6 CONCLUSION

In this paper, we propose an approach to model multi-dimensional networks, which can capture independent information from each dimension and dependent information across dimensions. Based on this approach, we propose the MINES framework which can embed multi-dimensional network with hierarchical structure to low-dimensional vector spaces. We can learn a set of node representations for each dimension using this framework. The learned representations for each dimension will contain the hierarchical information, the independent information from the specific dimension and also dependent information across dimensions. We evaluate the effectiveness of our framework on a multi-dimensional e-commerce network. The results of our experiments show the advancement of our framework.

In this work, we utilize linear functions to model the across dimension information and the hierarchical structure information. In our future work, more complicated non-linear functions such as exponential functions or even the neural networks can be used. Meanwhile, as a limitation in our work, we only focus on hierarchical structures with depth of 2 in this paper. As another direction in our future work, we would like to investigate the proposed framework with deeper hierarchical structures. Real-world networks typically evolve such as addition of new nodes and links, and deletion of old nodes and links. Therefore, multi-dimensional network embedding with dynamics should provide new insights in future.

ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers for their helpful comments. Yao Ma and Jiliang Tang are supported by the National Science Foundation (NSF) under grant number IIS-1714741 and IIS-1715940.

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*. 585–591.

[2] Michael GH Bell, Yasunori Iida, et al. 1997. Transportation network analysis. (1997).

[3] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. 2013. Multidimensional networks: foundations of structural analysis. *World Wide Web* 16, 5-6 (2013), 567–593.

[4] Michele Berlingerio, Fabio Pinelli, and Francesco Calabrese. 2013. Abacus: frequent pattern mining-based community discovery in multidimensional networks. *Data Mining and Knowledge Discovery* 27, 3 (2013), 294–320.

[5] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.

[6] Ronald L Breiger, Scott A Boorman, and Phipps Arabie. 1975. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of mathematical psychology* 12, 3 (1975), 328–383.

[7] Carter T Butts. 2009. Revisiting the foundations of network analysis. *science* 325, 5939 (2009), 414–416.

[8] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 119–128.

[9] Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang (Eds.). ACM, 295–304.

[10] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2017. A Survey on Network Embedding. *arXiv preprint arXiv:1711.08752* (2017).

[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. (2017).

[12] Palash Goyal and Emilio Ferrara. 2017. Graph Embedding Techniques, Applications, and Performance: A Survey. *arXiv preprint arXiv:1705.02801* (2017).

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

[14] Gueorgi Kossinets and Duncan J Watts. 2006. Empirical analysis of an evolving social network. *science* 311, 5757 (2006), 88–90.

[15] Richard A Kronmal and Arthur V Peterson Jr. 1979. On the alias method for generating random variables from a discrete distribution. *The American Statistician* 33, 4 (1979), 214–218.

[16] Ka-Wei Roy Lee and Ee-Peng Lim. 2016. Friendship maintenance and prediction in multiple social networks. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*. ACM, 83–92.

[17] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. *arXiv preprint arXiv:1706.01860* (2017).

[18] David Liben-Nowell and Jon M. Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*. ACM, 556–559.

[19] Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation* 19, 10 (2007), 2756–2779.

[20] Yao Ma, Suhang Wang, ZhaoChun Ren, Dawei Yin, and Jiliang Tang. 2017. Preserving Local and Global Information for Network Embedding. *arXiv preprint arXiv:1710.07266* (2017).

[21] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[22] MARGGF Magnani, Anna Monreale, Giulio Rossetti, and Fosca Giannotti. 2013. On multidimensional network measures. In *Italian conference on Sistemi Evoluti per le Basi di Dati (SEBD)*.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013).

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 701–710.

[27] Giulio Rossetti, Michele Berlingerio, and Fosca Giannotti. 2011. Scalable link prediction on multidimensional networks. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*. IEEE, 979–986.

[28] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. Visualization Large-scale and High-dimensional Data. *CoRR* abs/1602.00370 (2016).

[29] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). ACM, 1067–1077.

[30] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.

[31] Suhang Wang, Charu C. Aggarwal, Jiliang Tang, and Huan Liu. [n. d.]. Attributed Signed Network Embedding. In *Proceedings of CIKM*.

[32] Suhang Wang, Jiliang Tang, Charu C. Aggarwal, Yi Chang, and Huan Liu. 2017. Signed Network Embedding in Social Media. In *Proceedings of SDM*. 327–335.

[33] Suhang Wang, Jiliang Tang, Charu C. Aggarwal, and Huan Liu. 2016. Linked Document Embedding for Classification. In *Proceedings of CIKM*. 115–124.

[34] Stanley Wasserman and Katherine Faust. 1994. *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press.

[35] Barry Wellman. 1983. Network analysis: Some basic principles. *Sociological theory* (1983), 155–200.