Deep Multi-dimensional Network Embedding

ABSTRACT

It is crucial to obtain good features for nodes in networks when dealing with network related tasks such as node classification and link prediction. Recently, network embedding, which aims to learn low-dimensional node representations, has been proved to advance the performance of many of these tasks. Most of the existing network embedding algorithms have been designed for single dimensional networks where only one type of relation exists between nodes. However, many real world networks consist of multiple relations. They can be naturally modeled as multidimensional networks with each type of relation as a dimension. Multidimensional networks bring more complexity due to the introducing of multiple dimensions and complex relations between them. Network embedding algorithms designed for single dimensional networks are insufficient for multi-dimensional networks. In this paper, we propose a deep framework DMNE which performs **D**eep **M**ulti-dimensional **N**etwork **E**mbedding to learn node representations for multidimensional networks. Experimental results on two real-world multi-dimensional networks demonstrate the effectiveness of the proposed framework.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

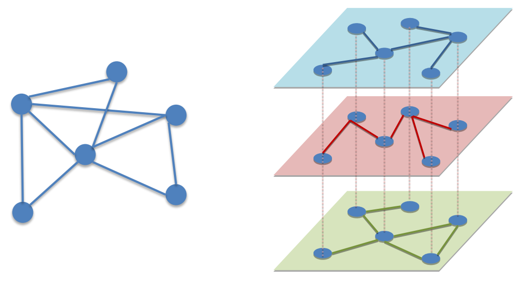
ACM proceedings, LaTeX, text tagging

ACM Reference Format:

. 1997. Deep Multi-dimensional Network Embedding. In *Proceedings of ACM Woodstock conference (WOODSTOCK'97)*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Networks, which encode relations between entities, can naturally represent data in many areas. Social networks, academic networks and transportation networks are some of the popular and important examples. It is of great interest to analyze the network and perform network related tasks to the nodes and links in the network. To achieve good performance on these tasks, it is crucial to find appropriate representations for nodes. Therefore, network embedding, which aims to learn low-dimensional representations, has attracted increasing attention in recent years [7, 21, 27].



(a) Single dimension networks **(b) Multidimensional networks**

Figure 1: Single dimension networks vs. multidimensional networks

Most of existing network embedding algorithms have been designed for single-dimension networks. As shown in Figure 1a, in a single dimension network, a pair of nodes can only have one type of relation. However, in many real-world networks, there are multiple types of relations between a pair of nodes [3]. For example, in social networking sites such as Facebook, two users could be connected by friend relations, and by various social interactions; and in product review sites such as Epinions, two users can be connected by trust, distrust relations, and via relations through products. We consider each relation type as a dimension and then the networks with multiple relations can be naturally represented as multidimensional networks as shown in Figure 1b. In a multidimensional network, the nodes are shared by all the dimensions, while each dimension has a unique relation type and distinct network structure. Multiple dimensions bring additional degree of complexity to the multi-dimensional networks. For example, a multidimensional network allows us to understand relations among its nodes from different perspectives by distinguishing relations in different dimensions [34]; and relations in different dimensions relate to each other for individual pairs [10]. Meanwhile, dimensions in multidimensional networks are not related to each other equally and some dimensions might be more similar than others [2]. Given the added complexity, multidimensional networks pose tremendous challenges to algorithms designed for single dimensional networks, which calls for dedicated efforts for multidimensional network embedding.

These added complexities also bring many opportunities for network analysis and mining [3]. Thanks to the availability of different dimensions, numerous network analysis and mining tasks have been facilitated. For example, multidimensional networks can help find better communities from noisy and sparse social media networks [29]; distinguishing relations from different dimensions can significantly boost the performance of node ranking and recommendations [26]; and scalable link prediction can be achieved by exploiting cross-dimensional information [25]. Hence, it has great potential to facilitate these additional multi-dimensional relations to learn better node representations.

To design network embedding algorithms for multi-dimensional networks, there are two major challenges: 1) How to model the multi-dimensional relations and capture the information from each dimension; and 2) How to capture the correlations between different dimensions. Some recent attempts of multi-dimensional network embedding [17, 23] use linear functions to capture the complex relations in multi-dimensional networks. However, the underlying structure of the network is highly non-linear [16], and the added complexity of multidimensional networks further increases the non-linearity and complexity. Therefore, to solve the aforementioned two challenges, in this paper, we propose deep models for multidimensional network embedding. Our major contributions can be summarized as follows:

- We provide a principled approach to model the complex relations in multi-dimensional networks and to capture the correlations between different dimensions mathematically;
- We propose novel multidimensional embedding deep learning frameworks, which captures multi-dimensional relations and dimension correlations into a coherent model; and
- We demonstrate the effectiveness of our frameworks by performing network related tasks on two real-world multi-dimensional networks.

The rest of this paper is arranged as follows. We introduce the approach to model the multi-dimensional network and the proposed embedding learning frameworks in Section 2. In Section 3, we conduct node classification and link prediction tasks on two real-world datasets to illustrate the effectiveness of the proposed frameworks. In Section 4, we review some works related to our problem. Finally, we conclude our work with future directions in Section 5.

2 THE PROPOSED FRAMEWORK

In this section, we introduce the deep architectures to perform multi-dimensional network embedding. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a multi-dimensional network, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ denotes a set of N nodes and $\mathcal{E} = \{e_1, \ldots, e_M\}$ represents a set of M edges. Each edge can be represented as a triplet (v_i, v_j, d), where v_i, v_j are two nodes connected by the edge and $d \in \{1 \ldots, D\}$ indicates the edge is in the d-th dimension of the network. D is the number of dimensions in the multi-dimensional network. Given the multi-dimensional network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, we aim to learn representations of nodes in $\mathcal{V}$ that can be used for various network related tasks. Next we introduce the deep architecture to model the multidimensional relations, followed by the approach to capture dimension correlations and the method to optimize the proposed framework.

2.1 Modeling the multi-dimensional relations

In a multi-dimensional network, nodes are shared by all the dimensions, while each dimension has different network structures. To capture the network information in each dimension, a natural way is to learn node representations for each dimension. To learn node representations in each dimension, a simple way is to regard each dimension as an independent network and learn node representations for each dimension separately. However, this method ignores the information across dimensions. The representations for the same node in different dimensions should share some properties, since they belong to the same node. To capture the shared information across dimensions, we propose a general node representation $\mathbf{u} \in \mathbb{R}^l$ for a given node v, where l is the length of the representation vector. The general representation $\mathbf{u}$ is supposed to capture the general information from all the dimensions. The general representations for all nodes can be denoted as a matrix $\mathbf{U} \in \mathbb{R}^{l \times N}$, where each column corresponds to a representation vector for a node. In different dimensions, different network structures encode different information to the node. To capture such information, we propose to describe the unique dimension property of each dimension using a dense vector. We can view each dimension as a different "context", where the "context" information can be described by a dense vector. We call this dense vector as "context representation" and denote the "context representation" for a dimension d as $\mathbf{e}_d \in \mathbb{R}^s$, where s is the length of the representation vector. The context representations for all the D dimensions can be represented as a matrix $\mathbf{E} \in \mathbb{R}^{s \times D}$.

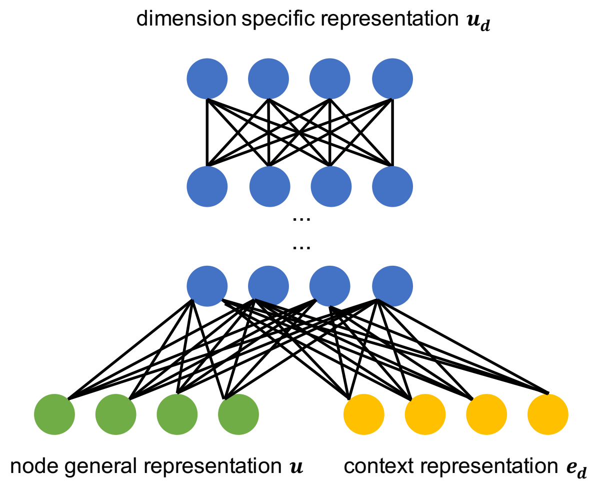

dimension specific representation $\boldsymbol{u_d}$
...
...
node general representation $\boldsymbol{u}$
context representation $\boldsymbol{e_d}$

Figure 2: Using neural network to model $g(\cdot, \cdot)$ function

Given the general representation and the context representation, we can now model the node representation for each dimension. The node representation of a given node, under a certain dimension, can be represented by "forcing" the "context" information into the node's general representation. We call the specific representation in dimension d for a given node v as dimension specific representation, which can be denoted as $\mathbf{u}_d \in \mathbb{R}^h$ where h is the length of the dimension-specific representation. The dimension specific representations of all nodes in dimension d can be denoted as a matrix $\mathbf{U}_d \in \mathbb{R}^{h \times N}$. To mathematically represent the dimension specific representation by "forcing" the context information into the general node representation $\mathbf{u}$, we use the following function $g(\cdot, \cdot)$ as:

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

Given the inherent complexity of multi-dimensional networks, in this work, we propose to use a neural network to model the function $g(\cdot, \cdot)$. The structure of the neural network is shown in Figure 2. The input of the neural network is a general representation $\mathbf{u}$ for node v and a context representation $\mathbf{e}_d$ of a given dimension d. The first layer of the neural network fuses the two input vectors into a hidden layer, and then several fully connected layers follow. The output of the neural network is the dimension specific representation of the node v in the dimension d, which is denoted as $\mathbf{u}_d$.

The advantages of modeling the dimension specific representation in this approach are four-fold. First, the model can capture the general information across all dimensions as well as the dimension specific information. Second, the applications of the learned representations are flexible since we can use different node representations for different tasks. For example, we can use general representation for dimension independent tasks such as node classification, and dimension specific representation for dimension-related tasks such as link prediction. Third, the representations needed to be learned and stored are the general representations and the context representations. Thus, the number of embedding parameters of the proposed approach are much less than existing multi-dimensional embedding algorithms and more stable with the increase of the number of dimensions. For example, the algorithm MINES in [17] has $N \times h \times D$ parameters, which is much lager than ours $N \times h + D \times s$ given that D is much smaller than N; while the number of parameters only increases s for ours as compared to $N \cdot h$ for MINES when we increase the dimension from D to $D + 1$. Also the number of embedding parameters only slightly increase compared to single-dimensional embedding algorithms. For example, a typical single-dimensional embedding algorithm has $N \times h$ parameters compared to ours $N \times h + D \times s$. Note that we do not consider the parameters from the function g for the proposed approach, which are shared by all nodes and much smaller than the embedding parameters. Fourth, the context representation naturally paves us a way to capture the correlations between dimensions and we will discuss more details in Section 2.3.

2.2 The DMNE framework

In this subsection, we introduce the framework DMNE to learn the representations by considering the multiple types of relations. To preserve the network information in the representations, we propose to use the linkage information. The idea is that good dimension specific representations can well preserve the network structure in the given dimension so that the edge can be predicted using the dimension specific representations. Following this idea, we propose a discriminative model, which takes a pair of nodes and a dimension as input and predict whether there is an edge between the nodes in the given dimension.

As shown in Figure 3, the DMNE framework consists of three major components. The two components in the bottom are the same neural networks modeling $g(\cdot,\cdot)$ function for two nodes. As described in the last section, each of these two components takes into a general node representation and a context representation and generate the dimension specific representation. These two components share weight parameters. Note that in this framework, the context representation in the two components are the same, as there are no across-dimensional edges; however, it is easy to extend it for networks with cross-dimensional links by inputting the corresponding context representations and we would like to leave it as one future work. The third component which is on the top of the previous two components is the prediction component.

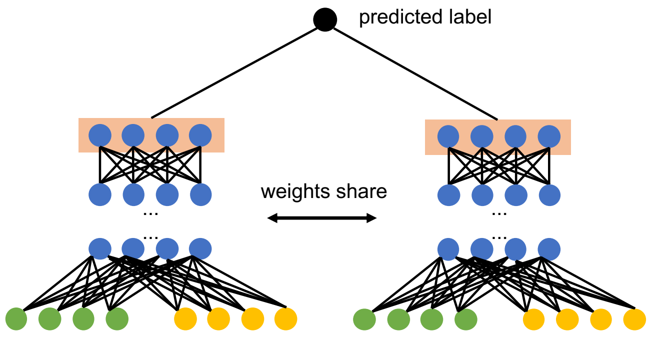


Figure 3: The DMNE framework

This component takes into two dimension specific representations and then makes the prediction about the existence of an edge in the given dimension. Ideally, the model should predict 1 if there is an edge between the two given nodes in the specific dimension, otherwise 0. The parameters in this neural network includes the weights for different layers, which we denote as $\mathbf{W}$, the general node representations $\mathbf{U}$ and the context representations $\mathbf{E}$.

To learn the parameters, we model the problem in a maximum likelihood fashion. Given a triplet (v_i, v_j, d), the probability of an edge existing between v_i and v_j in dimension d can be modeled as:

$$p(1|v_i, v_j, d) = \sigma(f(g(\mathbf{u}_i, \mathbf{e}_d), g(\mathbf{u}_j, \mathbf{e}_d))) \tag{2}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function and $f(\cdot,\cdot)$ is a function that takes into two representations and outputs a value. Different $f(\cdot,\cdot)$ functions including simple linear function, complex non-linear function or even neural networks can be used. In this paper, we empirically find that inner product works well. Accordingly, the probability of no edge exiting between them in dimension d can be modeled as

$$\begin{aligned} p(0|v_i, v_j, d) &= 1 - \sigma(f(g(\mathbf{u}_i, \mathbf{e}_d), g(\mathbf{u}_j, \mathbf{e}_d))) \\ &= \sigma(-f(g(\mathbf{u}_i, \mathbf{e}_d), g(\mathbf{u}_j, \mathbf{e}_d))) \end{aligned} \tag{3}$$

Given the set of training samples $\mathcal{T}$, which consists of positive samples $\mathcal{T}_p$ and negative samples $\mathcal{T}_n$, the likelihood of $\mathbf{E}$, $\mathbf{U}$ and $\mathbf{W}$ given $\mathcal{T}$ can be modeled as

$$\begin{aligned} P(\mathcal{T}|\mathbf{E},\mathbf{U},\mathbf{W}) &= \prod_{(v_i,v_j,d)\in\mathcal{T}_p} p(1|v_i,v_j,d) \cdot \prod_{(v_i,v_j,d)\in\mathcal{T}_n} p(0|v_i,v_j,d) \\ &= \prod_{(v_i,v_j,d)\in\mathcal{T}_p} \sigma(f(g(\mathbf{u}_i,\mathbf{e}_d), g(\mathbf{u}_j,\mathbf{e}_d))) \\ &\cdot \prod_{(v_i,v_j,d)\in\mathcal{T}_n} \sigma(-f(g(\mathbf{u}_i,\mathbf{e}_d), g(\mathbf{u}_j,\mathbf{e}_d))) \end{aligned} \tag{4}$$

where $\mathbf{W}$ denotes all the parameters of the neural network. Details about the training samples can be found in Section 2.4. The parameters $\mathbf{E}$, $\mathbf{U}$ and $\mathbf{W}$ can be learned by minimizing the negative logarithm of the likelihood in (4). The negative logarithm of the likelihood can be written as

$$\begin{aligned} L &= -\log P(\mathcal{T}|\mathbf{E}, \mathbf{U}, \mathbf{W}) \\ &= -\sum_{(v_i, v_j, d) \in \mathcal{T}_p} \log \sigma(f(g(\mathbf{u}_i, \mathbf{e}_d), g(\mathbf{u}_j, \mathbf{e}_d))) \\ &\quad - \sum_{(v_i, v_j, d) \in \mathcal{T}_n} \log \sigma(-f(g(\mathbf{u}_i, \mathbf{e}_d), g(\mathbf{u}_j, \mathbf{e}_d))) \end{aligned} \quad (5)$$

2.3 Capturing the dimension correlations

We have introduced the embedding learning framework DMNE, which mainly captures multiple types of relations. In DMNE, we treat each dimension equally assuming that dimensions affect each other equally. However, this might not be always true, as some dimensions could be more similar than others. We use the context representation to represent each dimension. Thus, dimensions with similar network structure should have similar context representations in the embedding space. If we have prior knowledge or model assumptions about these correlations, we can enforce them into the context representations. However, in many applications, it is unlikely that we can have such information. Therefore, it is desired to learn the dimension correlations from the data automatically.

To learn the correlations among different dimensions, we use a matrix-variate normal distribution [8] to model the structure of context representations $\mathbf{E}$. More specifically, we assume $\mathbf{E}$ follows the distribution

$$p(\mathbf{E}) = \mathcal{MN}(\mathbf{E}|\mathbf{0}_{s\times D}, \mathbf{I}_{s\times s} \otimes \Omega_{D\times D}) \quad (6)$$

where $\mathcal{MN}(\mathbf{X}|\mathbf{M}_{s\times D}, \mathbf{A}_{s\times s} \otimes \mathbf{B}_{D\times D})$ denotes a matrix-variate normal distribution [8] with mean $\mathbf{M} \in \mathbb{R}^{s\times D}$, row co-variance matrix $\mathbf{A} \in \mathbf{R}^{s\times s}$ and column co-variance matrix $\mathbf{B} \in \mathbb{R}^{D\times D}$. The probability density function of this distribution is defined as

$$p(\mathbf{X}|\mathbf{M}, \mathbf{A}, \mathbf{B}) = \frac{\exp(-\frac{1}{2} tr(\mathbf{A}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{B}^{-1}(\mathbf{X}-\mathbf{M})^T))}{(2\pi)^{sD/2}|\mathbf{A}|^{D/2}|\mathbf{B}|^{s/2}}. \quad (7)$$

We assume that the context representation matrix has a $\mathbf{0}$ mean, thus $\mathbf{M}_{s\times D} = \mathbf{0}_{s\times D}$. The row co-variance matrix $\mathbf{A}$ models the relationship between features, and in our case, features are assumed to be independent to each other and follow an identical distribution, thus $\mathbf{A}_{s\times s} = \mathbf{I}_s$. The co-variance matrix $\Omega_{D\times D}$ captures the relationships between different dimensions, and this is the part we need to estimate from the data.

We then model a joint probability using eq (4) and eq (6)

$$P = P(\mathcal{T}|\mathbf{E}, \mathbf{U}, \mathbf{W}) \cdot P(\mathbf{E})^{2\lambda}, \quad (8)$$

where λ controls the importance of the second term.

The negative logarithm of eq (8) after removing some constants is below:

$$\begin{aligned} L_2(\mathbf{E}, \mathbf{U}, \mathbf{W}, \Omega) = &-\sum_{(v_i, v_j, d) \in \mathcal{T}_p} \log \sigma(f(g(\mathbf{u}_i, \mathbf{e}_d), g(\mathbf{u}_j, {}_d))) \\ &- \sum_{(v_i, v_j, d) \in \mathcal{T}_n} \log \sigma(-f(g(\mathbf{u}_i, e_d), g(\mathbf{u}_j, e_d))) \\ &+ \lambda \cdot tr(\mathbf{E}\Omega^{-1}\mathbf{E}^T) + 2s\lambda \log|\Omega| \end{aligned} \quad (9)$$

As Ω is the column co-variance matrix of $\mathbf{E}$, we add the constraint $\Omega \succeq 0$ to make sure that the matrix Ω is positive semi-definite. The last term in eq (9) penalize the complexity of Ω and it is concave when $\lambda > 0$, which makes the problem hard to solve [33]. Hence, we replace it with $tr(\Omega) = k$ to control the complexity as in [33]. We empirically set k to be the number of dimensions D in this work. We then have the new formulation and we denote the framework capturing both multiple types of relations and dimension correlations as the DMNE++ framework:

$$\begin{aligned} L_2(\mathbf{E}, \mathbf{U}, \mathbf{W}, \Omega) = &-\sum_{(v_i, v_j, d) \in \mathcal{T}_p} \log \sigma(f(g(\mathbf{u}_i, \mathbf{e}_d), g(\mathbf{u}_j, {}_d))) \\ &- \sum_{(v_i, v_j, d) \in \mathcal{T}_n} \log \sigma(-f(g(\mathbf{u}_i, e_d), g(\mathbf{u}_j, e_d))) \\ &+ \lambda \cdot tr(\mathbf{E}\Omega^{-1}\mathbf{E}^T) \\ &s.t. \quad \Omega \succeq 0, tr(\Omega) = k \end{aligned} \quad (10)$$

where λ is the hyper-parameter to control the importance of the correlation term in the learning procedure. Note that the DMNE framework is a special case of DMNE++ with $\lambda = 0$. Next we introduce the optimization algorithm to learn general node representation $\mathbf{U}$, the context representations $\mathbf{E}$, the column co-variance matrix Ω and the neural network parameters $\mathbf{W}$.

2.4 An Optimization Method

To optimize Eq. (10), we adopt an alternating updating approach as introduced in [33]. We update $\mathbf{U}$, $\mathbf{E}$ and $\mathbf{W}$ and the co-variance matrix Ω alternately. Next, we introduce the two main procedures in the alternating optimization.

To update the representations $\mathbf{E}, \mathbf{U}$ and the neural network parameters $\mathbf{W}$ fixing Ω, we adopt *ADAM* [12] to optimize the loss function (10). The training set $\mathcal{T}$ consists of positive and negative samples. The positive samples are generated from the network – they are the edge set $\mathcal{T}_p = \mathcal{E}$. The negative samples are generated upon the positive samples. For each positive sample (v_i, v_j, d), we fix v_i and d and randomly sample n nodes without links to v_i to form n negative samples. We denote the set of negative samples as $\mathcal{T}_n = \mathcal{E}_{ne}$ and then we have $\mathcal{T} = \mathcal{T}_p \cup \mathcal{T}_n$.

To update Ω given the representations $\mathbf{E}$, $\mathbf{U}$ and parameters $\mathbf{W}$, the loss function (10) can be simplified as

$$\begin{aligned} &\min_{\Omega} tr(\mathbf{E}\Omega^{-1}\mathbf{E}^T) \\ &s.t. \quad \Omega \succeq 0, tr(\Omega) = k \end{aligned} \quad (11)$$

where $\mathbf{E}$ is fixed.

According to [6, 33], the solution of (11) is given by

$$\Omega = \frac{k \cdot (\mathbf{E}^T\mathbf{E})^{\frac{1}{2}}}{tr((\mathbf{E}^T\mathbf{E})^{\frac{1}{2}})} \quad (12)$$

The optimization procedure is summarized in Algorithm 1. We randomly initialize $\mathbf{E}, \mathbf{U}$ and $\mathbf{W}$ in line 1 and initialize $\Omega = \mathbf{I}$ in line 2, assuming context representation do not affect each other initially. The epoch parameter controls the number of epochs to run in the learning procedure. Inside each epoch, we alternately perform the two updates. In line 5, we update $\mathbf{U}$, $\mathbf{E}$ and $\mathbf{W}$ by optimizing (10) on the training set $\mathcal{T}$. Then Ω is updated according to (12) in line 7.

Algorithm 1: Optimization procedure

```
Input: $\lambda$, $\mathcal{T}$, epoch
Output: U, E, W, $\Omega$
1 Initialize U, E and W randomly;
2 Initialize $\Omega$ as I;
3 Initialize $ep = 0$;
4 while ep<epoch do
5     Shuffle $\mathcal{T}$;
6     Optimize (10) using ADAM while fixing $\Omega$ on $\mathcal{T}$;
7     Update $\Omega$ according to (12);
8     ep $\leftarrow$ ep +1
9 end
```

3 EXPERIMENTS

In this section, we perform several network related tasks to verify the effectiveness of our frameworks. We first introduce the two datasets we use in our experiments, then discuss the results of node classification and link prediction tasks. We study the impact of important parameters and also provide a case study to show the effectiveness of the learned context representations.

3.1 Datasets

We use the following two datasets for both node classification and link prediction tasks. Some important statistics of the two datasets are shown in Table 1.

- **Epinions**: Epinions is a general customer review site. Users in this site can write reviews to products, and rate helpfulness of reviews written by other users. There are also trust and distrust relations in this site. We form a 5-dimensional network on this dataset, the 5 relations are – 1) co-review: if two users review some common items, we create a link between them in the co-review dimension; 2) helpfulness rating: if a user rate the other user's review, we create a link between them in the helpfulness rating dimension; 3) co-rating: if two users rate some common reviews, we create a link between them in the co-rating dimension; 4) trust: the trust relation naturally forms a dimension; and 5) distrust: the distrust relation naturally forms a dimension. We also label each user in the dataset based on the items the user reviewed. Each item is associated with a category. We label each user as the category he/she reviewed most items from. There are 15 different labels in total.
- **DBLP**: DBLP is an academic publication site. We collect all the publications during 1998 – 2017. Then, we form a co-authorship multi-dimensional network with each year as a dimension. We create am edge between two authors if they co-author a paper in the given year. Each author is labeled according to their publications. More specifically, each paper is associated with a venue, while each venue belongs to a high-level category such as "machine learning" and "theoretical computer science". The author is assigned the high-level category where he/she published most of his/her publications. There are 10 different high-level categories in total.

Table 1: Statistics of datasets

	DBLP	Epinions
number of nodes	138,072	15,108
number of edges	2,015,650	485,154
number of dimensions	20	5
number of labels	10	15

3.2 Baseline Methods

There are two possible strategies to handle multi-dimensional networks for single-dimensional network embedding algorithms. First, we can convert a multi-dimensional network to a single-dimension network by aggregating edges from all dimensions (or not distinguishing edges in different dimensions) and then apply single-dimensional network embedding algorithms to the aggregated single-dimensional network. Second, we can treat it as a set of single-dimension networks by considering each dimension separately, and apply single-dimensional network embedding algorithms to each dimension. In this work, we focus on the first strategy when we apply single-dimensional network embedding algorithms to multi-dimensional networks since the second strategy is likely to suffer from the data sparsity problem. In particular, we compare our frameworks with some classic and state of the art network embedding algorithms as follows:

- Non-negative matrix factorization(NMF) [15]: We aggregate the multi-dimensional network to a single dimension network and apply NMF to the aggregated network. We factorize the adjacency matrix of the aggregated single dimensional network to get the node representations.
- LINE [28] : LINE can only be applied to single dimension network, so we aggregate the multidimensional network to a single dimension network and apply LINE on it. We follow the default setting of LINE and set the number of samples to 800 million.
- node2vec [7]: Similar to LINE, we also apply node2vec to the aggregated single dimensional network. We follow the default setting for node2vec and follow [7] to perform a grid search on 10 percents of the testing data for parameters p and q.
- MINES [17]: MINES is a recent proposed multi-dimensional network embedding method. We apply MINES without the hierarchical structure to the multidimensional network and use the general embedding as the node representation in the node classification task.
- DMNE++: We apply DMNE++ to the multi-dimensional network which captures both multiple types of relations and dimension correlations.
- DMNE: DMNE is a special case of DMNE++, where we set the hyper-parameter of the correlation term λ to 0. It only captures multiple types of relations while ignoring the dimension correlations.
- DMNE0: This is a variant of DMNE, where we do not use the context representations. In other words, we do not distinguish representations for different dimensions.

3.3 Node Classification

In the node classification task, we observe the labels of a fraction of nodes in the network. The task is to predict the labels for the remaining nodes. We perform node classification on both data sets. We use F_1-macro score and F_1-micro score as metrics to evaluate the performance as in [7, 21].

We follow a similar procedure as in [21] to perform the node classification task. We randomly sample a portion (R_t) of the nodes, and use them as training set and use the remaining nodes as testing set. Specially, we set the portion of training node to 10%-90% with a step size of 20%. We use logistic regression classifier in the experiments. For each setting, we repeat the experiment for 10 times and report the average values for the two metrics. We set the length of the representations in NMF, LINE and node2vec to 64. For our proposed algorithm and its variants, we set the length of the representations to 64 ($l = h = s = 64$) for a fair comparison and we use the general node representation for the node classification task. We set $\lambda = 100$ for DMNE++ and we will discuss more details about parameter analysis for DMNE++ in the following subsection.

3.3.1 Results. The node classification results are summarized in Figure 4 and Figure 5 for Epinions and DBLP, respectively. Note that the performances of NMF on both datasets are not comparable with the other baselines. Specially, it only achieves F1-macro: 0.2113 and F1-micro 0.4012 on the Epinions data set under the 90% training ratio setting and F1-macro: 0.2380 and F1-micro 0.3580 on the DBLP data set under the 90% training ratio setting. Hence, we do not include NMF in the figures. We can make the following observation from the two figures:

- MINES outperforms the single dimension methods LINE and node2vec most of the time, which shows that multi-dimensional relations bring richer information and utilizing them can improve the quality of the learned representations.
- Both DMNE++ and DMNE outperform DMNE0. These results indicate that it is necessary to distinguish representations from different dimensions and the proposed models can effectively capture the multi-dimensional relations via general and context representations.
- DMNE obtains better performance than MINES, which suggests that the deep architecture can help capture the complexity of multi-dimensional networks.
- DMNE++ obtains 1%-2% improvement in most of the settings compared to DMNE. These results suggest that it is important to capture the dimension correlations, which is complementary to capturing multiple types of relations.

3.4 Link Prediction

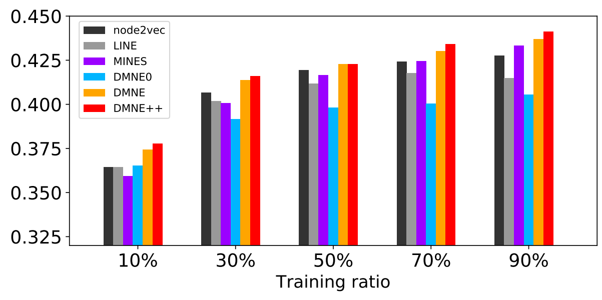
In the link prediction task, we try to predict whether a non-existing edge would emerge between nodes given a network. In the setting for single dimensional networks, we randomly remove some fraction of edges while ensuring the remaining network is connected and use the remaining network to learn representations [7]. We adjust this setting for multidimensional networks. In multi-dimensional networks, we perform link prediction for different dimensions, separately. Given a dimension, we remove some fractions of edges in this dimension while the remaining network in this dimension is connected. Then if the removed edges exist in other dimensions, we also remove those edges in other dimensions for a fair comparison with single dimensional network embedding baselines. After removing the edges, we use the remaining network to learn the representations for all nodes. Then, following [7], we formulate the link prediction task as a binary classification task, where positive samples are the real existing edges, and negative samples are randomly generated node pairs. We use element-wise multiplication to combine the node representations of the given two nodes and use it as the input of the binary classifier. To form the training set, we put all the remaining edges into the training set and then randomly generate an equal number of node pairs as negative samples. We form the testing set in a similar way, using the removed edges as positive samples.

For Epinions, we perform link prediction for each dimension, separately, and report the results for each dimension. We also report the average performance over all the dimensions. In each dimension, we remove 20% of links and perform the multi-dimensional embedding and link prediction as mentioned above. For DBLP, we only perform link prediction for the last dimension (Year 2017) since it is unreasonable to use information of 2017 to predict links in years before 2017. We perform link prediction on two settings with 50% and 70% of the edges removed. The metric we use in the link prediction task is the AUC score as in [7]. For our framework DMNE++, DMNE and DMNE0, we use the dimension specific embedding to perform the link prediction task. We set the length of the representations to 64 for all the methods.

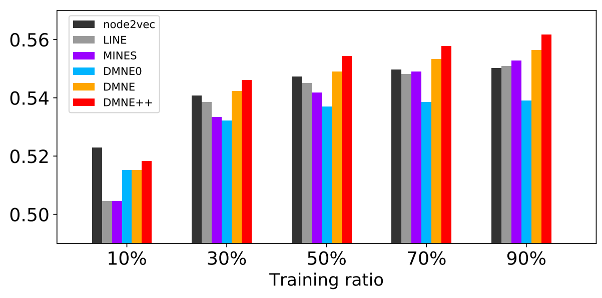
3.4.1 Results. The results of link prediction on the DBLP data set are shown in Figure 6. We make the following observations from the figure:

- DMNE, DMNE++ and MINES obtain better performance than MMF, LINE and node2vec. DMNE, DMNE++ and MINES capture multiple types of relations; while NMF, LINE and node2vec perform on the aggregated network. This result demonstrate that it is important to distinguish relations from different dimensions, which can help learn better node representations for link prediction.
- DMNE0 performs worse than DMNE and DMNE++. It supports the importance of context representations to capture differences between dimensions.
- MINES performs a little bit worse to DMNE and DMNE++, while it requires much more parameters as discussed in Section 2.1.
- DMNE++ obtains better performance than DMNE. DMNE only captures multiple types of relations; while DMNE++ models both multiple types of relations and dimension correlations. This result further demonstrates the importance of dimension correlations in embedding.

The results of link prediction on the Epinions dataset are shown in Table 2. Note that in the table "dim 0", "dim 1", "dim 2", "dim 3", and "dim 4" denote the co-review dimension, helpfulness dimension, co-rating dimension, trust dimension and dis-trust dimension, respectively. The average performance over all dimensions is reported in the last column of the table. All methods do not perform equally across different dimensions, which suggests that different dimensions indeed have different structures. DMNE and DMNE++

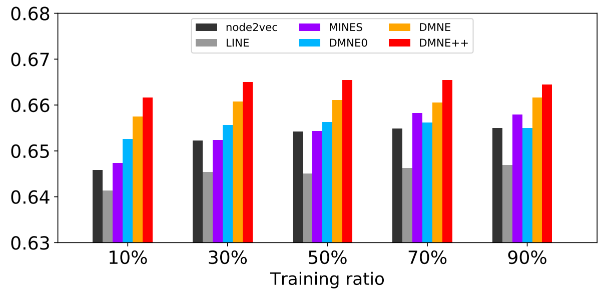


(a) Epinions: F1-macro

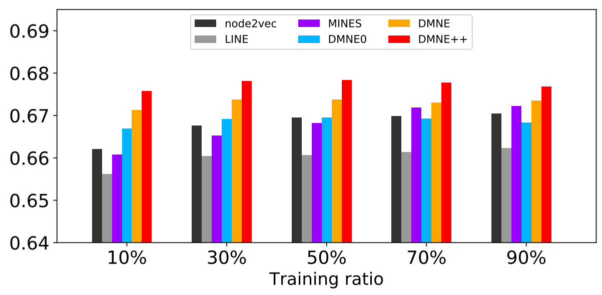


(b) Epinions: F1-micro

Figure 4: Performance Comparison of Node classification on Epinions dataset



(a) DBLP: F1-macro



(b) DBLP: F1-micro

Figure 5: Performance Comparison of Node classification on DBLP dataset

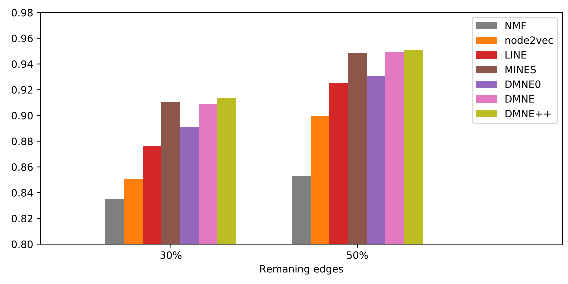


Figure 6: Performance comparison of link prediction in terms of AUC on DBLP dataset

obtain the dimension-specific representations by fusing information from all dimensions via the deep architecture to combine the general and the context representations. Their performance is more stable across dimensions. In terms of the average performance, we make similar observations to these on DBLP as shown in Figure 6.

3.5 Parameter Analysis

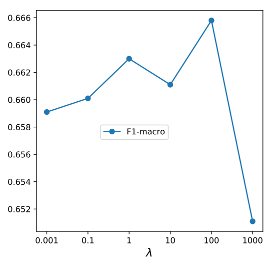
The major parameters in our algorithm is the length of the representations and the hyper-parameter λ controlling the importance of the correlation term. We only demonstrate parameter sensitive analysis for node classification task on the DBLP data set with the

method	dim 0	dim 1	dim 2	dim 3	dim 4	average
NMF	0.9463	0.8092	0.9381	0.8909	0.9066	0.8982
LINE	0.8612	0.7747	0.8100	0.8088	0.8888	0.8287
node2vec	0.8713	0.7866	0.8773	0.8121	0.8732	0.8441
MINES	0.9621	0.8268	0.9572	0.8036	0.8711	0.8842
DMNE++	0.9582	0.8107	0.9539	0.9109	0.9402	0.9148
DMNE	0.9403	0.8018	0.9424	0.9180	0.9323	0.9070
DMNE0	0.8643	0.7270	0.9131	0.8432	0.8559	0.8407

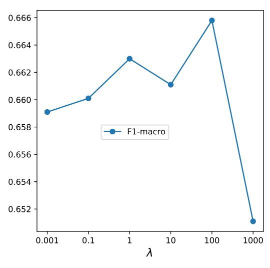
Table 2: Performance comparison of link prediction in terms of AUC on Epinions. Note that in the table "dim 0", "dim 1", "dim 2", "dim 3", and "dim 4" denote the co-review dimension, helpfulness dimension, co-rating dimension, trust dimension and dis-trust dimension, respectively. The average performance over all dimensions is reported in the last column of the table.

training ratio to 80% since we make similar observations with other settings.

To investigate how DMNE++ is affected by the hyper-parameter λ, we fix the length of the representation to 64 and vary the values of λ. The results are shown in Figure 7. As we can observe, with the increase of λ, the performance follows a similar trend – it first increases, reaches the peak performance ($\lambda = 100$) and then decreases. λ balances the contributions from two types of information, i.e., multiple types of relations and dimension correlations. Too large or too small will result in overfitting to one type of information. This analysis also suggests the importance to capture both types of information for multi-dimensional network embedding.



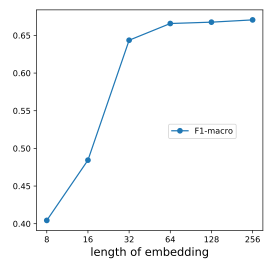
(a) DBLP: F1-macro



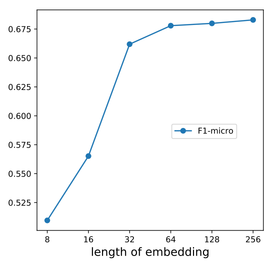
(b) DBLP: F1-micro

Figure 7: The effect of different values of λ on DBLP dataset

To investigate how DMNE++ is affected by the length of the representations, we fix the hyper-parameter λ to 100 and vary the length of the representations. The results are shown in Figure 8. We note that the performance increases very fast when the length of the representation is small (smaller than 32). Too smaller length is not sufficient to capture the network structures. When the length of the representation is larger than 32, the performance becomes relatively stable.



(a) DBLP: F1-macro



(b) DBLP: F1-micro

Figure 8: The effect of different length of embedding on DBLP dataset

3.6 Case Study

In this subsection, we show a case study on the DBLP dataset since the dimension in DBLP denotes the temporal information. We try to investigate the dimension correlations via the learned co-variance matrix $\mathbf{\Omega}$ and the context representations $\mathbf{E}$. In this case study, the representation length is set to 64 and the hyper-parameter λ is set to 100.

Firstly, we visualize the column co-variance matrix $\mathbf{\Omega}$ in Figure 9. We can observe that the color near the diagonal is darker than other parts, which suggests higher correlations. Furthermore, the color gets lighter and lighter as the distance to the diagonal increases, which suggests lower correlations. Intuitively, close years are more related than distant years, which is consistent with observations from $\mathbf{\Omega}$. Therefore, the matrix can indeed capture the dimension correlations.

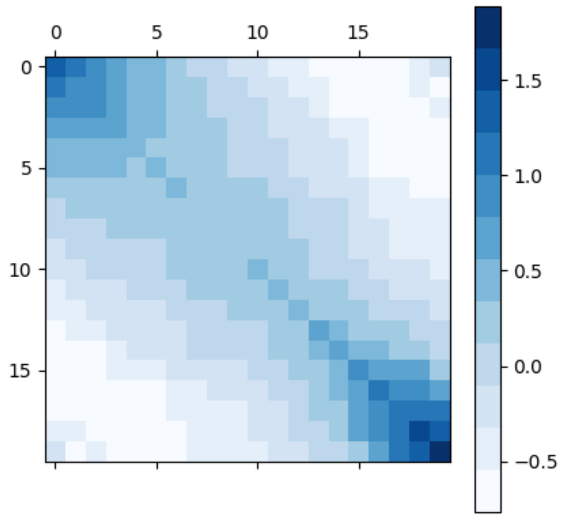


Figure 9: The co-variance matrix Ω

We also project the context representation for each dimension to a vector of length 3 using Principal Component Analysis (PCA) [11]. Then, we plot the context representations in a 3 dimensional scatter plot, where each dot represents a context representation. The distribution of these dots is demonstrated in Figure 10. Note that the notions 0-19 mean the dimension 0 to dimension 19, which are corresponding to Year 1998 to Year 2017. As we can observe from Figure 10, consecutive years naturally stay closer than distant years, which indicates that the context representations of consecutive years are more similar than distant years. This is consistent with the results of the co-variance matrix. Hence, via capturing the dimension correlation, we can fuse dimension correlation information into the context representation.

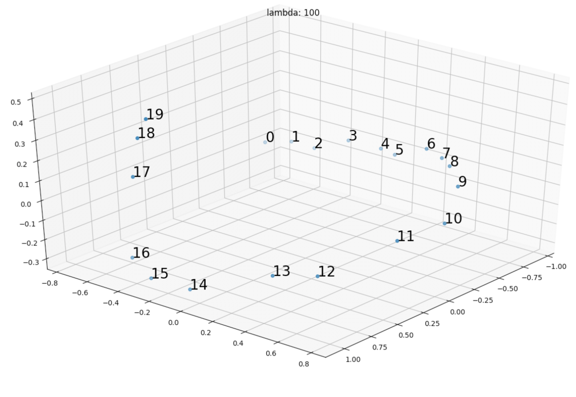


Figure 10: The context embedding in a 3d plot

4 RELATED WORK

Obtaining appropriate node representations in a network is an essential and crucial step for various network related data mining and analysis tasks. Traditional methods such as Laplacian Eigenmap [1] and IsoMap [30] utilize the adjacency matrix to perform dimension reduction on it. These methods involves calculating eigenvalues of the adjacency matrix and are not be able to scale to very large scale networks. More scalable algorithms such as DeepWalk [21] and LINE [28] are introduced recently. These methods which are commonly known as network embedding aim to learn low-dimensional node representations while preserving proximity. DeepWalk is inspired by word2vec [19, 20], and it utilizes random walk to capture the proximity in the network. node2vec [7] extends Deepwalk by introducing biased random walk. Recent works [14, 22] connecting network embedding methods with matrix factorization show a theoretical perspective of some network embedding algorithms. There are also some attempts such as HNE [5] and SDNE [31] to use deep models to learn node representations. Aside from preserving proximity, some network embedding algorithms try to preserve other information in the representations. For example, struct2vec [24] tries to preserve structural identity; SiNE [32] tries to utilize the sign of link to improve the representation in signed networks; PRUNE [13] and LOG [18] try to include global ranking information in the representation learning procedure. Comprehensive overviews on network embedding algorithms can be found in two recent surveys [4, 9].

These methods are mainly designed for single dimensional networks. However, many networks with multiple relations between nodes can be naturally represented as multidimensional network. Examples and analysis of multidimensional network can be found in [3]. Numerous network analysis and mining tasks have been facilitated thanks to the availability of multiple dimensions. For example, multidimensional networks can help find better communities from noisy and sparse social media networks [29]; and scalable link prediction can be achieved by exploiting across-dimensional information [25]. The availability of multiple dimensions shows great potential to advance the existing network embedding algorithms. Recently, there are a few attempts to perform network embedding for multidimensional networks [17, 23]. In [17], the authors model the multidimensional relations by linear function. In [23], the authors propose to learn embedding for each node in each dimension and then linearly combine them based on attention mechanism. The attentions in this model are learned in a weakly supervised way. Adding more complexities to the non-linear structured network, linear function is likely to be not capable to well capture the information. In this paper, we propose a deep model using neural networks to learn representations for multi-dimensional networks.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a deep model to model the multidimensional relations. Particularly, we propose to learn context representation for each dimension while learning general node representations. We propose to model the dimension specific representations using the general node representations and context representations. The proposed framework DMNE can learn the general node representations, the context representations and the function to combine them to generate the dimension specific representations. DMNE++ further enhances DMNE by incorporating the dimension correlation into the framework. Comprehensive experiments on two real world data sets with node classification and link prediction tasks show the effectiveness of our frameworks.

In this work, we assume there are no cross-dimensional edges, thus we first want to extend DMNE and DMNE++ with cross-dimensional edges. Furthermore, we only utilize the network structure, while many real-world networks are associated rich side information. For example, in many networks, nodes are associated with rich attributes. Therefore, investigating multi-dimensional network embedding with node attributes would be an interesting future direction. Finally, networks are also naturally evolving, incorporating the dynamic information into multi-dimensional network embedding could be another promising direction.

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] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. 2011. Foundations of multidimensional network analysis. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE, 485–489.

[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] Hongyun Cai, Vincent W Zheng, and Kevin Chang. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[5] 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.

[6] Hongliang Fei and Jun Huan. 2013. Structured feature selection and task relationship inference for multi-task learning. *Knowledge and information systems* 35, 2 (2013), 345–364.

[7] 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.

[8] Arjun K Gupta and Daya K Nagar. 1999. *Matrix variate distributions*. Vol. 104. CRC Press.

[9] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *arXiv preprint arXiv:1709.05584* (2017).

[10] Theus Hossmann, George Nomikos, Thrasyvoulos Spyropoulos, and Franck Legendre. 2012. Collection and analysis of multi-dimensional network data for opportunistic networking research. *Computer Communications* 35, 13 (2012), 1613–1625.

[11] Ian T Jolliffe. 1986. Principal component analysis and factor analysis. In *Principal component analysis*. Springer, 115–128.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh, and Shou-De Lin. 2017. PRUNE: Preserving Proximity and Global Ranking for Network Embedding. In *Advances in Neural Information Processing Systems*. 5263–5272.

[14] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.

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

[16] Dijun Luo, Feiping Nie, Heng Huang, and Chris H Ding. 2011. Cauchy graph embedding. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 553–560.

[17] Yao Ma, Zhaochun Ren, Ziheng Jiang, Jiliang Tang, and Dawei Yin. 2018. Multi-Dimensional Network Embedding with Hierarchical Structure. (2018).

[18] 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).

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

[20] 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.

[21] 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.

[22] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2017. Network Embedding as Matrix Factorization: UnifyingDeepWalk, LINE, PTE, and node2vec. *arXiv preprint arXiv:1710.02971* (2017).

[23] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An Attention-based Collaboration Framework for Multi-View Network Representation Learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1767–1776.

[24] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 385–394.

[25] 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.

[26] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 93–102.

[27] 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.

[28] 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*. International World Wide Web Conferences Steering Committee, 1067–1077.

[29] Lei Tang, Xufei Wang, and Huan Liu. 2009. Uncoverning groups via heterogeneous interaction analysis. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 503–512.

[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] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[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] Yu Zhang and Dit-Yan Yeung. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536* (2012).

[34] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. 2011. Graph cube: on warehousing and OLAP multidimensional networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 853–864.