MULTI-TASK NETWORK EMBEDDING WITH ADAPTIVE LOSS WEIGHTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Network embedding is to learn low-dimensional representations of nodes which mostly preserve the network topological structure. In real-world networks, however, nodes are often associated with a rich set of attributes and labels which are potentially valuable in seeking more effective vector representations. To properly utilize this information, we propose a Joint Autoencoders framework for Multi-task network Embedding (JAME), which aims to encode a shared representation of local network structure, node attributes, and available node labels. Jointly embedding via multi-task learning is strongly dependent on the relative weighting between each task's loss function. Tuning these weights by hand is an expensive and difficult process, making multi-task learning prohibitive in practice. Therefore, we define an adaptive loss weighting layer capable of learning an optimal combination of loss weights during representation learning. Empirical evaluations on real-world datasets show effectiveness and efficiency of our JAME model compared to relevant baseline methods.

1 INTRODUCTION

Network embedding techniques project nodes in a network to a low-dimensional vector space while preserving network structure and inherent properties. It has attracted increasing attention recently due to significant progress in downstream graph mining tasks such as node classification (Grover & Leskovec, 2016; Tang et al., 2015; Perozzi et al., 2014; Wang et al., 2016; Bojchevski & Günnemann, 2017; Tsitsulin et al., 2018), link prediction (Grover & Leskovec, 2016; Ou et al., 2016; Bojchevski & Günnemann, 2017; Tsitsulin et al., 2018), community detection (Cavallari et al., 2017), and shortest distance approximation (Rizi et al., 2018).

Traditionally, network embedding is mainly focused on preserving plain network structure, mapping nodes in the same neighborhood close to each other in the vector space. In real-world scenarios, however, nodes are often associated with a set of auxiliary information such as contents (e.g. profile attributes or textual features) or labels (e.g. community or affiliation group) which form labeled attributed networks. It has been shown that labels are strongly influenced by and inherently correlated to both network structure and attribute information (Huang et al., 2017b). Therefore, modeling and integrating node attribute proximity and labels into network embedding is potentially helpful in learning more effective and meaningful vector representations.

Recently, various efforts have been devoted to learn low-dimensional vector representations for attributed networks (Gao & Huang, 2018; Huang et al., 2017a; Meng et al., 2019; Yang et al., 2018; Zhang et al., 2019). These methods considered network structure and attributes, but ignored abundant labels associated to nodes that could potentially benefit network representation learning and subsequent analytic tasks. To the best of our knowledge, only LANE (Huang et al., 2017b) smoothly incorporates label information into the attributed network embedding while preserving their correlations. More precisely, LANE learns three types of latent representations via spectral techniques from attribute similarity matrix, network adjacent matrix, and label similarity matrix, and project them into a common embedding space. Indeed, three embedding tasks jointly encode the network information into a shared representation through optimizing three objective functions. Despite the great novel algorithm, LANE faces several drawbacks, like, 1) expensive computations for matrix decomposition, 2) manually tuning weights for multiple objective functions, and 3) relatively low performance in node classification and link prediction.

To overcome the problems of the existing methods, we need to design an efficient multi-task learning framework with adaptive loss weighting. It has been widely shown that multi-task learning improves the overall performance of each task relative to learning them separately (Caruana, 1997; Chen et al., 2009; 2011). Specifically, combining autoencoders as a unified framework has demonstrated promising performance to solve multiple classification and regression tasks simultaneously (Meir et al., 2017; Banijamali et al., 2017; Cadena et al., 2016; Zhuang et al., 2015). The structure of autoencoders facilitates the integration of multiple information sources towards an efficient joint representation learning. Therefore, we propose a Joint Autoencoders model for Multi-task network Embedding, abbreviated as JAME, which encodes shared representations for nodes in the network via optimizing multiple objectives. In detail, our end-to-end model consists of three autoencoders: a graph convolutional network (GCN) (Kipf & Welling, 2016) autoencoder and two standard autoencoders. The GCN autoencoder captures graph local connectivity by applying spectral filters followed by nonlinear activation functions. Two standard autoencoders perform supervised classification tasks which infer the embeddings of nodes involving both labels and attributes. Overall, our joint autoencoders handle three embedding tasks by optimizing three respective objective functions. However, the performance is highly dependent on an appropriate choice of weighting between each task's loss. To search for an optimal weighting, we define an adaptive loss weighting layer where inputs of the layer are loss values. During joint representation learning, weights within the loss weighting layer are contributed to the gradient and get updated. In summary, the main contributions of this paper are as follows:

- We propose a novel end-to-end model for labeled attributed networks, JAME, which aims at encoding joint representations of nodes via multi-task learning.
- We define an adoptive loss weighting layer which simultaneously optimizes the weighted combination of losses during representation learning.
- We empirically evaluate effectiveness and efficiency of JAME on real-world networks, showing its superior performance to the state-of-the-art baselines.

The rest of the paper is organized as follows. Section 2 presents related work. We present notations and the proposed model in Section 4. Section 5 presents empirical evaluation. In Section 6, we conclude and discuss future work.

2 Related works

Network embedding via deep neural networks has become an efficient tool to deal with today's large-scale networks. Recently, there has been numerous embedding methods which focused on preserving the topological structure of plain networks (Goyal & Ferrara, 2018) along with various real-world applications (Jia & Saule, 2018; Zhou, 2019; Rizi & Granitzer, 2019; Nikolentzos et al., 2017). However, today's networks are often associated with a rich set of attributes and abundant label information, which can be encoded within a joint embedding framework. Gao & Huang (2018) proposed DANE, a joint autoencoders model which takes graph adjacency matrix and attributes as input to learn a shared representation via optimizing four objective functions, i.e. first-order proximity loss, high-order proximity loss, semantic proximity loss, and complementary loss. Their final embeddings preserve first-order and high-order proximity of the network which result superior performance in typical graph mining tasks. Another work by Meng et al. (2019) named CAN employs two variational autoencoders containing an inference and a generative model. The inference model encodes features of nodes and attributes into Gaussian distributions while the generative model reconstructs the original edges and attributes. During representation learning, a reparameterization method is applied to transform embeddings from the Gaussian random variables to the deterministic variables which helps to measure affinities between nodes and attributes. CAN finally learns separate embeddings for attributes and nodes in the same semantic space, mapping nodes nearby their relevant attributes.

Some recent works like BANE (Yang et al., 2018) and LANE (Huang et al., 2017b) applied matrix factorization techniques to attributed networks to jointly embed nodes and attributes into a shared vector space. BANE first applies Weisfeiler-Lehman proximity to aggregate adjacency and attribute matrices into a unified proximity matrix. The proximity matrix then is factorize by Weisfeiler-Lehman factorization approach to obtain final binary representations. LANE first integrate attributes and labels in to same proximity matrix applying a spectral technique persevering node and attribute proximities. Then, it uses graph Laplacian (Chung & Graham, 1997) to smoothly embed label matrix and proximity matrix uniformly into the final vector representations. Although, LANE observes abundant labels during representation learning, the performance is relatively low with an expensive runtime complexity.

Multi-task learning conducts multiple related learning tasks simultaneously such that useful information can be shared among tasks. During multi-task learning usually multiple objectives are jointly optimized to improve efficiency and accuracy of all relevant tasks. Here, the main challenge is how to automatically obtain optimal weights for these objectives within model training. Kendall et al. (2018) propose a multi-task loss function based on maximising the Gaussian likelihood with homoscedastic uncertainty which optimizes combination of multiple objectives during joint training. The presented model can learn multi-task weighting and outperforms separate models trained individually on each task. Similarly, BenTaieb & Hamarneh (2017) define a multi-loss objective function integrating uncertainty in each loss term which is trainable during learning. Their convolutional autoencoder classifies medical images while learning how to optimally combine multiple objectives. To our best knowledge, we are the first employing multiple loss weighting into graph embedding to improve graph analysis tasks.

3 Problem Definition and Preliminaries

Let $G = (V, E)$ be a labeled attributed network, where V is a set of n nodes and E is the set of edges. Each node in the network is associated with a set of attributes and label information indicating its affiliation group. We denote $\mathbf{A} \in \{0, 1\}^{n\times n}$ as adjacency matrix, $\mathbf{X} \in \{0, 1\}^{n\times f}$ as the attribute matrix with f attributes, and $\mathbf{Y} \in \{0, 1\}^{n\times k}$ as label matrix with k categories. In the attribute matrix $\mathbf{X}$, every row x_i describes the attributes associated with node i. In the label matrix, $\mathbf{Y}_{ij} = 1$ indicates node i belongs to category j. The final embedding matrix is denoted as $\mathbf{U} \in \mathbb{R}^{n\times d}$ where $d \ll n$ is the embedding dimension size.

Based on the terminologies described above, we formally define the problem of multi-task network embedding as follows: Given a labeled attributed network G, the goal is to learn a mapping function $\Theta : \{\mathbf{A}, \mathbf{X}, \mathbf{Y}\} \mapsto \mathbf{U}$ in an supervised manner such that the network structure $\mathbf{A}$, attribute $\mathbf{X}$, and label information $\mathbf{Y}$ are preserved running the following learning tasks:

- **Network structure embedding** encodes network topological structure into vector representation:
$$\Theta_a : \{\mathbf{A}\} \mapsto \mathbf{U} \tag{1}$$
- **Attributed network embedding** integrates attributes of the nodes into vector space:
$$\Theta_x : \{\mathbf{A}, \mathbf{X}\} \mapsto \mathbf{U} \tag{2}$$
- **Labeled network embedding** incorporates node labels into vector embeddings:
$$\Theta_y : \{\ \mathbf{A}, \mathbf{Y}\} \mapsto \mathbf{U} \tag{3}$$

4 Proposed Approach

4.1 Framework

In this paper, we propose JAME, a Joint Autoencoders model for Multi-task network Embedding with adoptive loss weights. The framework of JAME is shown in Figure 1. In general, JAME consists of three major parts: network structure autoencoder, attributed network autoencoder, and labeled network autoencoder. These three autoencoders handle three embedding tasks which are optimized at the same time. The network structure autoencoder employs GCN (Kipf & Welling, 2016) layers to encode local links by reconstruction of the adjacency matrix $\mathbf{A}$. The attributed network autoencoder receives the adjacency matrix as input and tries to predict their associated attributes $\mathbf{X}$. The labeled network autoencoder aims at predicting nodes labels $\mathbf{Y}$ during joint training. For a multi-task learning problem containing multiple objective functions, JAME offers an adaptive loss weighting layer L capable of learning optimal weighted combination of several objectives. In the following subsections, we will introduce our proposed JAME model in detail.

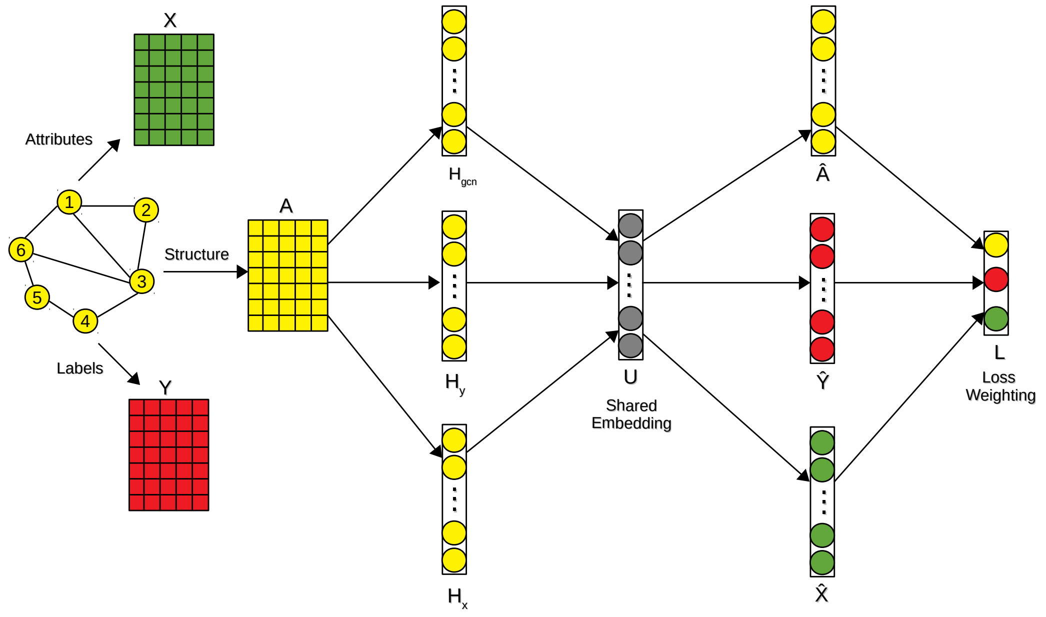


Figure 1: The framework of our end-to-end JAME model. The model is fitted with the adjacency matrix $\mathbf{A}$ as input and outputs adjacency matrix $\mathbf{A}$, attribute matrix $\mathbf{X}$ and binary label matrix $\mathbf{Y}$, respectively. Shared Embedding layer aggregates information from structure, attribute and labels while Loss Weighting layer learns optimal weights for each embedding task.

4.2 Network Structure Embedding

We employ GCN (Kipf & Welling, 2016) layers into basic autoencoders to encapsulate non-linear structure of the input graph. In Figure 1, the GCN encoder gets latent representations for each node while the decoder computes the pair-wise distance between node latent representations produced by the encoder. After applying a non-linear activation function, the decoder reconstructs the original adjacency matrix. More formally, our GCN encoder is defined as:

$$\mathbf{H}_{gcn} = \sigma(\tilde{\mathbf{A}}\,\mathrm{ReLU}(\tilde{\mathbf{A}}W^{(0)} + b^{(0)})W^{(1)} + b^{(1)}), \tag{4}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_n)\mathbf{D}^{-\frac{1}{2}})$ is a symmetrically normalized adjacency matrix with self-connections, $\mathbf{I}_n$ is the identity matrix, and $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. $\mathbf{H}_{gcn} \in \mathbb{R}^{n\times s}$ is the hidden representation from the GCN encoder with dimension size s. $W^{(l)}$ is the trainable weight matrix and $b^{(l)}$ is the bias of l^{th} layer. Sigmoid $\sigma(.)$ and ReLU are non-linear element-wise activation functions.

In Figure 1, the bottleneck features in the shared embedding layer constitutes an informative compact representation of the data. More formally, shared embedding is the summation of representations from the GCN encoder $\mathbf{H}_{gcn}$, attribute encoder $\mathbf{H}_x$ and label encoder $\mathbf{H}_y$:

$$\mathbf{U} = \sigma((\mathbf{H}_{gcn} + \mathbf{H}_x + \mathbf{H}_y)W^{(2)} + b^{(2)}), \tag{5}$$

The graph decoder reconstructs the graph adjacency matrix:

$$\hat{\mathbf{A}} = \sigma(\mathrm{ReLU}(\mathbf{U}W^{(3)} + b^{(3)})), \tag{6}$$

where $\hat{\mathbf{A}}$ is the reconstruction of the adjacency matrix, and $\mathbf{U} \in \mathbb{R}^{n\times d}$ is the shared representation matrix with dimension size d. Inspired by Tran (2018), we learn model parameters by minimizing a masked cross-entropy loss formulation:

$$\mathcal{L}_{ce} = -\mathbf{a}_i \log(\sigma(\hat{\mathbf{a}}_i)) - (1 - \mathbf{a}_i)\log(1 - \sigma(\hat{\mathbf{a}}_i)), \tag{7}$$

$$\mathcal{L}_a = \frac{\sum_i m_i \odot \mathcal{L}_{ce}}{\sum_i m_i}. \tag{8}$$

where $\mathbf{a}_i \in \mathbb{R}^n$ is an adjacency vector of $\mathbf{A}$, and $\hat{\mathbf{a}}_i \in \mathbb{R}^n$ is the reconstruction output vector of $\hat{\mathbf{A}}$. $\odot$ is the Hadamard (element-wise) product, and m_i is a Boolean mask: $m_i = 1$ if $\mathbf{a}_i = 1$, else $m_i = 0$.

4.3 Attributed Network Embedding

To capture attribute proximity associated to nodes, we employ a standard autoencoder. The encoder gets latent representations for each node while the decoder reconstructs the attributes of nodes. The hidden representation of the attribute encoder is defined as:

$$\mathbf{H}_x = \sigma(\mathrm{ReLU}(\mathbf{A}W^{(0)} + b^{(0)})W^{(1)} + b^{(1)}), \tag{9}$$

here $\mathbf{A}$ is the adjacency matrix, and $\mathbf{H}_x \in \mathbb{R}^{n \times s}$ is the hidden representation from the attribute encoder with dimension size s.

The attribute decoder rebuilds the attributes associated to nodes:

$$\hat{\mathbf{X}} = \sigma(\mathrm{ReLU}(\mathbf{U}W^{(3)} + b^{(3)})), \tag{10}$$

where $\hat{\mathbf{X}}$ is the reconstruction of the attribute matrix, and $\mathbf{U} \in \mathbb{R}^{n \times d}$ is the final shared representation with dimension size d. Due to the binary valued representation of the attribute matrix, we optimize the binary cross-entropy to learn model weights:

$$\mathcal{L}_x = -\mathbf{x}_i \log(\sigma(\hat{\mathbf{x}}_i)) - (1 - \mathbf{x}_i) \log(1 - \sigma(\hat{\mathbf{x}}_i)), \tag{11}$$

where, $\mathbf{x}_i \in \mathbb{R}^n$ is an attribute vector of $\mathbf{X}$, and $\hat{\mathbf{x}}_i \in \mathbb{R}^n$ is a reconstruction output vector of $\hat{\mathbf{X}}$.

4.4 Label Informed Network Embedding

Label information plays an essential role in determining the inscape of each node with strong intrinsic correlations to network structure and node attributes. We propose to model labels information via a supervised autoencoder. The encoder gets latent representations for each node while the decoder predicts the labels of nodes. The label encoder transformation is defined as:

$$\mathbf{H}_y = \sigma(\mathrm{ReLU}(\mathbf{A}W^{(0)} + b^{(0)})W^{(1)} + b^{(1)}), \tag{12}$$

where $\mathbf{H}_y \in \mathbb{R}^{n \times s}$ is the hidden representation from the label encoder with dimension size s.

The label decoder predicts the abundant labels of nodes:

$$\hat{\mathbf{Y}} = \sigma(\mathrm{ReLU}(\mathbf{U}W^{(3)} + b^{(3)})), \tag{13}$$

where $\hat{\mathbf{Y}}$ is the reconstruction of the binary label matrix, and $\mathbf{U} \in \mathbb{R}^{n \times d}$ is the final shared representation matrix with dimension size d. In multi-class setting, we minimize the categorical cross entropy loss to update model parameters:

$$\mathcal{L}_y = \sum_{i=1}^{k} -\mathbf{y}_i \log(\sigma(\hat{\mathbf{y}}_i)), \tag{14}$$

here $\mathbf{y}_i \in \mathbb{R}^n$ is a label vector of $\mathbf{Y}$, $\hat{\mathbf{y}}_i \in \mathbb{R}^n$ is a reconstruction output vector of $\hat{\mathbf{Y}}$, and k is the number of label categories.

4.5 Multi-Task Weighting

JAME handles three embedding tasks through optimizing three objective functions. To learn joint vector representations, we need to combine multiple loss functions where each loss has a weight. Since searching for the optimal weightings is expensive and difficult, we propose to learn a weighted linear sum of losses using a loss weighting layer:

$$L = \mathrm{ReLU}(w_a \mathcal{L}_a + w_x \mathcal{L}_x + w_y \mathcal{L}_y), \tag{15}$$

where w_a, w_x, and w_y are weights for the network structure, attribute, and label embedding loss respectively. The parameters in the layer L are trained jointly with autoencoders, hence the weights are contributing to the gradient and get updated.

4.6 JAME TRAINING

Our goal is to minimize the multi-task loss function $L = \sum_i w_i \mathcal{L}_i$, where the sum runs over all T tasks. We thus propose an adaptive method, in which w_i can vary at each training step $t : w_i = w_i(t)$. To optimize the weights $w_i(t)$ in $L(t) = \sum_i w_i(t)\mathcal{L}_i(t)$, we propose a simple algorithm that w_i directly involved to the backpropagated gradient from each task. The detail of our method is described in Algorithm 1. At each training step t, the algorithm receives a mini-batch of nodes and updates loss weights and model weights until convergence.

Algorithm 1: JAME Training with Adaptive Loss Weights

```
Initialize loss weights $w_i(0) = 1 \; \forall i$
Initialize model weights $W^{(l)} \; \forall l$
Pick weights of the shared layer $W^{(U)}$
for $t = 0$ to max_iterations do
    Input batch to compute $\mathcal{L}_i(t) \; \forall i$ and $L(t) = \sum_i w_i \mathcal{L}_i(t)$ [standard forward pass]
    Compute $G^{(i)}_{W^{(U)}}(t) = \nabla_{W^{(U)}} w_i(t)\mathcal{L}_i(t) \; \forall i$
    Compute $L_{grad} = \sum_i G^{(i)}_{W^{(U)}}(t)$
    Compute loss weights gradients $\nabla_{w_i} L_{grad}$
    Compute standard gradients $\nabla_{W^{(l)}} L(t)$
    Update $w_i(t) \mapsto w_i(t+1)$ using $\nabla_{w_i} L_{grad}$
    Update $W^{(l)}(t) \mapsto W^{(l)}(t+1)$ using $\nabla_{W^{(l)}} L(t)$ [standard backward pass]
end
```

4.7 TRAINING COMPLEXITY

It is not difficult to see that the runtime complexity of JAME is $O(ncdI)$, where n is the number of nodes in the graph, d is the maximum dimension of the hidden layers, I is the number of iterations. c is the average number of non-zero entries in each row of matrix $\mathbf{A}$ (average degree of the network) which is a constant in scale-free networks (Barabási & Bonabeau, 2003). Parameter d is related to the dimension of embedding vectors but not related to the number of nodes. I is also a constant and independent from n. Therefore, cdI is independent from n, and thus the overall training complexity is $O(n)$ which linear to the number of nodes in the network.

5 EMPIRICAL EVALUATION

In this section, we conduct experiments to evaluate the effectiveness of JAME on node classification, link prediction and attribute inferring tasks. Single task evaluation can reveal whether our shared embeddings retain certain type of information. We first introduce the datasets, baseline methods, and experimental settings before presenting the main experiments. We then investigate how loss weighting effects the performance of each individual task. Last, we study the efficiency of JAME framework compared to the baseline methods. Our reference code and data are available at `https://www.dropbox.com/sh/zyuwgi067ojn427/AAAzfwpfml3BpMI9vbY7TGzOa?dl=0`. [1]

5.1 DATASETS

We conduct our experiments on the following labeled attributed network datasets: Cora (Sen et al., 2008), Citeseer (Sen et al., 2008) and Facebook (Leskovec & Mcauley, 2012). Cora and Citeseer are citation networks, where nodes are publications and edges are citation links. Attributes of each node

[1] We will share the code on Github in future.

are bag-of-words representations of the corresponding publications. The labels indicate research topics of the publications. The Facebook network is built from relation data of 10 ego networks, while attributes are constructed from profile information. We set the ego number of each node as its group label. The statistics of the datasets are provided in Table 1.

Table 1: Statistics of datasets.

Dataset	Nodes	Edges	Attributes	Labels
Cora	2,708	5,429	1,433	7
Citeseer	3,312	4,660	3,703	6
Facebook	4,049	88,234	576	10

5.2 Baselines

We compare JAME with the following relevant network embedding baselines:

- **LANE:** It is a labeled attributed network embedding model that learns node representations based on eigen-decomposition of the graph affinity, attribute affinity and label affinity matrices (Huang et al., 2017b).
- **CAN:** It learns network representations via a variational auto-encoder algorithm, which consists of an inference model for encoding attributes into Gaussian distributions and a generative model for reconstructing both real edges and attributes (Meng et al., 2019).
- **attri2vec:** It integrates the network structure and node attributes together seamlessly by a defined transformation function (Zhang et al., 2019). To preserve the network structure, DeepWalk (Perozzi et al., 2014) mechanism is employed, which makes nodes sharing similar neighbors embedded closely in the attribute subspace.
- **DANE:** It captures the network non-linearity and preserves the proximity of both topological structure and node attributes via a joint autoencoder model (Gao & Huang, 2018).

5.3 Parameter Settings

For all of the baselines, we follow the authors suggested hyperparameter settings available in the original papers as follows. DANE needs to construct a high-order adjacency matrix where the window size is $w = 10$, the walk length is $l = 80$, and the number of walks is $\gamma = 10$. For CAN, the hyperparameters of prior distributions, $\sigma_{\mathcal{V}}, \sigma_{\mathcal{A}}$, are set as 1, and the model is trained in 200 iterations. In attri2vec, $w = 10$, $l = 100$, $\gamma = 40$, and the number of iterations is 100 million. LANE has three objective functions which are manually weighted as 1, $\alpha_1 > 1$ and $\alpha_2 > 10$ to achieve the optimal performance. We train our JAME model for 8 iterations using the Adam optimizer (Kingma & Ba, 2014) with the learning rate being 0.01. The dimension size of each hidden layer is $s = 300$.

5.4 Node Classification

Node classification is a widely adopted task for validating quality of network embeddings (Goyal & Ferrara, 2018). To investigate the contribution of integrating node labels, we partially observe node labels during representation learning, then we conduct node classification. Among baselines, only LANE (Huang et al., 2017b) is able to leverage node labels into vector representations, hence we set it as our main baseline. For both methods, we choose embedding dimension $d = 128$, and we employ Macro-F1 to measure the performance of node classification. To make a comprehensive evaluation, we randomly observe 10% to 100% of the labels during representation learning. From Table 2, we can find that our proposed JAME consistently outperforms LANE on Citeseer and Facebook. In Cora, observing more than 50% of labeled nodes shows better performance. It shows JAME fed with more labels achieves higher performance since multiple tasks potentially compensate each other by learning from the shared information. In Table 3, once more labels are contributed in multi-task learning, more attributes and links are also involved in shared representations. Overall, the presented results indicate effectiveness and robustness of JAME to handle and encode additional label information associated to the network.

Table 2: Performance of node classification observing different percentages of labels during representation learning.

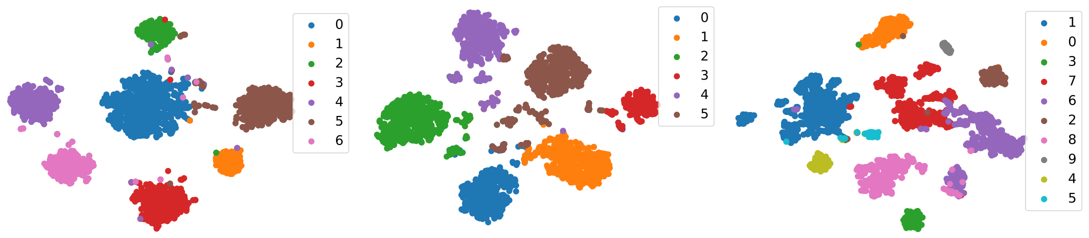
Dataset	**Model**	**Macro-F1**				
		10%	30%	50%	80%	100%
Cora	JAME	0.621 ± 0.002	0.663 ± 0.002	**0.729 ± 0.002**	**0.892 ± 0.002**	**0.981 ± 0.002**
	LANE	**0.663 ± 0.001**	**0.684 ± 0.001**	0.726 ± 0.001	0.769 ± 0.001	0.808 ± 0.001
Citeseer	JAME	**0.616 ± 0.002**	**0.651 ± 0.002**	**0.731 ± 0.002**	**0.810 ± 0.002**	**0.936 ± 0.002**
	LANE	0.538 ± 0.001	0.607 ± 0.001	0.616 ± 0.001	0.635 ± 0.001	0.677 ± 0.001
Facebook	JAME	**0.740 ± 0.002**	**0.769 ± 0.002**	**0.813 ± 0.002**	**0.902 ± 0.002**	**0.983 ± 0.002**
	LANE	0.522 ± 0.001	0.534 ± 0.001	0.559 ± 0.001	0.578 ± 0.001	0.581 ± 0.001

Table 3: Impact of observing labels on multi-task weighting during representation learning.

Dataset	**Task Weights**								
	50%			80%			100%		
	w_a	w_x	w_y	w_a	w_x	w_y	w_a	w_x	w_y
Cora	0.163	0.206	0.633	0.225	0.316	0.652	0.333	0.383	0.678
Citeseer	0.120	0.109	0.528	0.166	0.135	0.556	0.251	0.237	0.588
Facebook	0.153	0.138	0.346	0.182	0.150	0.368	0.281	0.224	0.398

5.5 Network Visualization

To further show capability of JAME for label encoding, we learn representations $d = 128$ observing full labeled networks, and visualize them by t-SNE (Maaten & Hinton, 2008). The visualization results for all datasets are shown in Figure 2. Different node colors indicate different users groups. We see learned embeddings from different classes are clearly compact and separated.



(a) Cora (b) Citeseer (c) Facebook

Figure 2: Network visualization of JAME embeddings observing full labeled networks.

5.6 Link Prediction

Link prediction is a commonly used task to demonstrate the meaningfulness of the vector representations for preserving local connections in the network. To carry out the link prediction, we follow Grover & Leskovec (2016) experiments which gather positive and negative edges. The positive examples are obtained by randomly removing 50% of the existing edges from the original graph whereas negative examples are node pairs which are not connected by edges (non-existing edges). We construct edge features from node representations by the Hadamard product (Grover & Leskovec, 2016) to feed a logistic regression. Figure 3 shows the link prediction performance of JAME and the baselines for different embedding sizes. As shown, in the Citeseer dataset, JAME consistently performs better than any of the baseline models. In Cora and Facebook datasets, JAME obtains competitive AUC scores as CAN, since both employ GCN to encode local connections in the network.

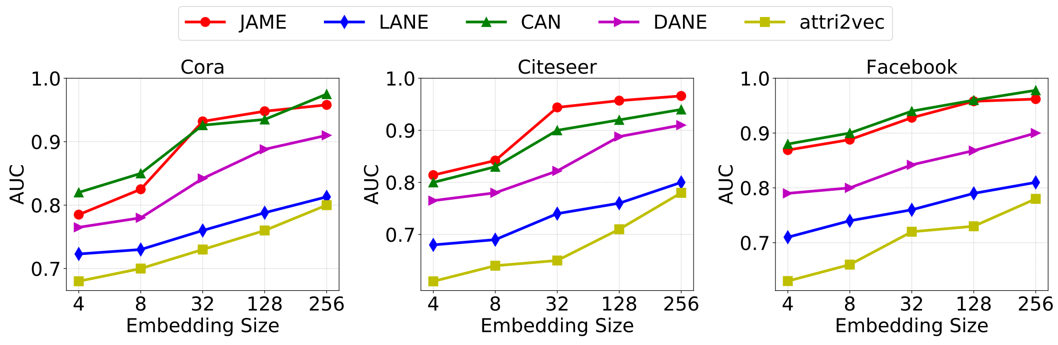


Figure 3: Link prediction performance of different methods for different embedding sizes

5.7 Attribute Inference

Attribute inference aims at predicting the value of attributes associated to the nodes in the network. We follow the CAN (Meng et al., 2019) experiments for attribute inference which conduct a binary classification task using a logistic regression. To evaluate our learned embeddings, we randomly divide nodes into a training (80%), and a test set (20%). We employ the AUC to measure the attribute inference performance since attributes of the nodes are 0/1-valued. Figure 4 presents the performance of JAME against the baseline models on the three attributed networks. We can find that JAME and CAN show competitive performance in the Citeseer and Facebook datasets as both employ joint autoencoders. CAN employs two variational autoencoders to simultaneously reconstruct both adjacency and attribute matrices that leads to more accurate attribute persevering in final vector embeddings.

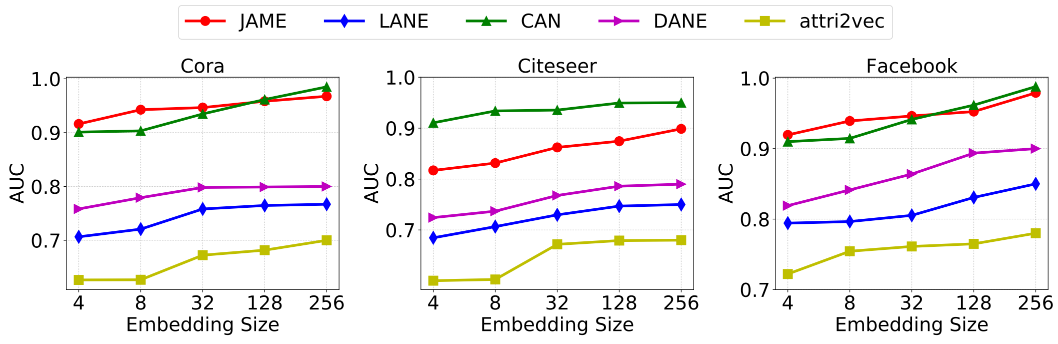


Figure 4: Attribute inference performance of different methods for different embedding sizes

5.8 Loss Weighting for Multi-Task Learning

Multi-task learning concerns the problem of optimizing a joint model with respect to multiple objectives. Training the model with different loss weights can effect the performance in multiple tasks. We initialize all weights in the loss weighting layer to 1.0 then the model adjusts these weights to the optimal values during representation learning. Table 4 demonstrates the training steps for learning optimal weights on the Cora dataset. We observe that the loss weighting layer learns the optimal combination of weights to have the highest performance on each task.

5.9 Efficiency Evaluation

To study the efficiency of JAME, we compare it with all the baselines on the Facebook dataset. Among baselines, attri2vec (Zhang et al., 2019) requires around 10944.35 sec to learn embeddings for the Facebook network. Due to the high runtime of attri2vec, we compare JAME to the other

Table 4: Performance of Link Prediction (LP), Attribute Inference (AI), and Node Classification (NC) applying weighted combination of multiple embedding tasks. Results are reported on the Cora network dataset.

Epoch	**Task Weights**			**LP**	**AI**	**NC**
	w_a	w_x	w_y	AUC	AUC	Macro-F1
1	0.920	0.930	0.916	0.746	0.811	0.615
2	0.837	0.856	0.831	0.829	0.895	0.818
3	0.755	0.777	0.763	0.887	0.929	0.913
4	0.671	0.698	0.733	0.922	0.938	0.960
5	0.587	0.619	0.714	0.937	0.946	0.964
6	0.503	0.542	0.704	0.944	0.948	0.972
7	0.418	0.459	0.689	0.947	0.954	0.978
8	**0.333**	**0.383**	**0.678**	**0.951**	**0.956**	**0.981**

baselines which learn representations in more reasonable time. From the results in Figure 5, we observe that JAME takes much less running time than LANE, DANE and CAN consistently. As the number of input nodes increases, the performance difference in time also raises up. Although DANE and CAN employ autoencoders, they need a lot more iterations to converge which increases the computation time. In summary, all these observations illustrate the efficiency and scalability of our proposed joint model.

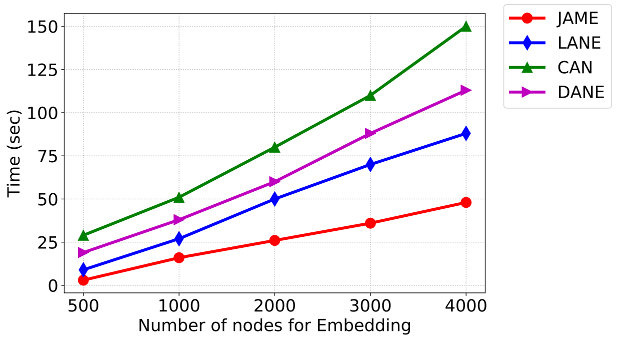


Figure 5: Running time comparison of baselines with respect to the number of nodes on Facebook.

6 Conclusion and future work

Labeled data is an essential information source available in a variety of attributed networks, which is beneficial to network representation learning. Incorporating labels into network representation learning is promising however challenging. To this end, we propose a novel framework JAME, which jointly projects labels and attributes into a unified vector space by running multiple embedding tasks. Our proposed model learns a weighted combination of multiple objective functions via a loss weighting layer. Via experiments on three real-world networks, we demonstrate (1) JAME can effectively incorporate the network structure, attributes, and labels into final representations, (2) learning embeddings by JAME is more efficient than the baselines as training complexity is linear to the number of nodes. As to future work, we aim to extend our JAME model to a heterogeneous one, where the network contains multiple types of nodes and edges.

References

Ershad Banijamali, Amir-Hossein Karimi, Alexander Wong, and Ali Ghodsi. Jade: Joint autoencoders for dis-entanglement. *arXiv preprint arXiv:1711.09163*, 2017.

Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003.

Aïcha BenTaieb and Ghassan Hamarneh. Uncertainty driven multi-loss fully convolutional networks for histopathology. In *Intravascular Imaging and Computer Assisted Stenting, and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pp. 155–163. Springer, 2017.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

Cesar Cadena, Anthony R Dick, and Ian D Reid. Multi-modal auto-encoders as joint estimators for robotics scene understanding. In *Robotics: Science and Systems*, volume 5, pp. 1, 2016.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 377–386. ACM, 2017.

Jianhui Chen, Lei Tang, Jun Liu, and Jieping Ye. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 137–144. ACM, 2009.

Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 42–50. ACM, 2011.

Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

Hongchang Gao and Heng Huang. Deep attributed network embedding. In *IJCAI*, volume 18, pp. 3364–3370, 2018.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 855–864, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2.

Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM international conference on data mining*, pp. 633–641. SIAM, 2017a.

Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 731–739. ACM, 2017b.

Haofeng Jia and Erik Saule. Graph embedding for citation recommendation. *arXiv preprint arXiv:1812.03835*, 2018.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pp. 539–547, 2012.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Baruch Epstein Meir, Tomer Michaeli, et al. Joint auto-encoders: a flexible multi-task learning framework. *arXiv preprint arXiv:1705.10494*, 2017.

Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. Co-embedding attributed networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 393–401. ACM, 2019.

Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114. ACM, 2016.

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

Fatemeh Salehi Rizi and Michael Granitzer. Predicting event attendance exploring social influence. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 2131–2134. ACM, 2019.

Fatemeh Salehi Rizi, Joerg Schloetterer, and Michael Granitzer. Shortest path distance approximation using deep learning techniques. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 1007–1014. IEEE, 2018.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pp. 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-3469-3. doi: 10.1145/2736277.2741093.

Phi Vu Tran. Multi-task graph autoencoders. *arXiv preprint arXiv:1811.02798*, 2018.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference*, pp. 539–548. International World Wide Web Conferences Steering Committee, 2018.

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

Hong Yang, Shirui Pan, Peng Zhang, Ling Chen, Defu Lian, and Chengqi Zhang. Binarized attributed network embedding. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 1476–1481. IEEE, 2018.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Attributed network embedding via subspace discovery. *arXiv preprint arXiv:1901.04095*, 2019.

Sida Zhou. Empirical effect of graph embeddings on fraud detection/risk mitigation. *arXiv preprint arXiv:1903.05976*, 2019.

Fuzhen Zhuang, Dan Luo, Xin Jin, Hui Xiong, Ping Luo, and Qing He. Representation learning via semi-supervised autoencoder for multi-task learning. In *2015 IEEE International Conference on Data Mining*, pp. 1141–1146. IEEE, 2015.