Weak Supervision Network Embedding for Constrained Graph Learning

Ting Guo[1], Xingquan Zhu[2(✉)], Yang Wang[1], and Fang Chen[1]

[1] Data Science Institute, University of Technology Sydney, Sydney, Australia
{ting.guo,yang.wang,fang.chen}@uts.edu.au
[2] Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, USA
xzhu3@fau.edu

Abstract. Constrained learning, a weakly supervised learning task, aims to incorporate domain constraints to learn models without requiring labels for each instance. Because weak supervision knowledge is useful and easy to obtain, constrained learning outperforms unsupervised learning in performance and is preferable than supervised learning in terms of labeling costs. To date, constrained learning, especially constrained clustering, has been extensively studied, but was primarily focused on data in the Euclidean space. In this paper, we propose a weak supervision network embedding (WSNE) for constrained learning of graphs. Because no label is available for individual nodes, we propose a new loss function to quantify the constraint-based loss, and integrate this loss in a graph convolutional neural network (GCN) and variational graph auto-encoder (VGAE) combined framework to jointly model graph structures and node attributes. The joint optimization allows WSNE to learn embedding not only preserving network topology and content, but also satisfying the constraints. Experiments show that WSNE outperforms baselines for constrained graph learning tasks, including constrained graph clustering and constrained graph classification.

Keywords: Weak supervision · GNN · Network embedding

1 Introduction

Graph-structured data are becoming increasingly common in many real-world applications, such as social networks, citation networks, knowledge graphs, telecommunication networks, and biological networks [4]. In graph-structured data, nodes represent individual entities, and edges represent relationships and interactions between entities. For example, in citation networks, each document denotes a node, and a citation link between two documents is treated as an edge. Learning node embeddings is one of the most important and active research topics in network feature representation learning. Many models [24] have been proposed to embed each node into a continuous vector space. Because embedding

K. Karlapalem et al. (Eds.): PAKDD 2021, LNAI 12712, pp. 488–500, 2021.
https://doi.org/10.1007/978-3-030-75762-5_39

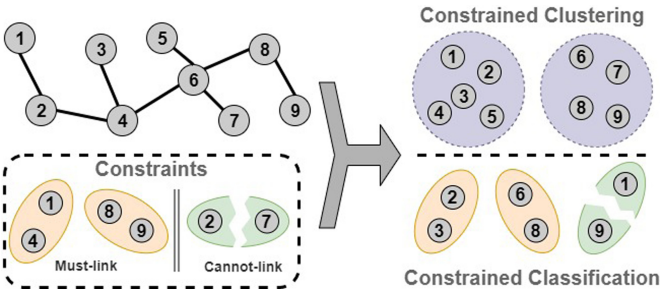

Fig. 1. Constrained graph learning. The weak supervision, must-link and cannot-link, specifies pairwise node constraints. Constrained graph clustering clusters node into groups, and constrained classification classifies a pair of nodes as either same-group or different-group.

vectors preserve topology and node content information, they can be directly used in downstream tasks, such as clustering or classification.

Depending on the availability of node labels, existing node embedding methods fall into two categories: (1) *Strong-supervision:* labels are provided to individual nodes (or a portion of nodes) for finding embeddings with maximum separability. (2) *Non-supervision:* no labels are provided to the learning task, so embedding aims to find a mapping to an output space without labeled responses. For strong-supervision methods, they require labeled training data, which often imply high labeling costs and obstacles. This bottleneck effect manifests itself in various ways, including the insufficient quantity of labeled data, insufficient subject-matter expertise to label data, and insufficient time to label and prepare data. On the other hand, for non-supervision methods, because no labels are given to the learning algorithms to differentiate samples, it is difficult for them to find structures satisfying users' requirements.

In addition to the above strong- *vs.* non-supervision scenarios, *Weak supervision* in machine learning provides a new setting where no labels are provided for individual instances, but some noisy, low-quality, conditional constraints over unlabeled data are available for model learning [1]. Weak supervision learning is intended to decrease labeling costs and increase the efficiency of human efforts expended in hand-labeling data while makes the outputs usable and comprehensive to specific problems. For example, pairwise constraints, *must-link* and *cannot-link*, are a means of weak supervision that constrain a pair of data points to belong to the same cluster (must-link) or different clusters (cannot-link). By integrating such pairwise constraints, constrained clustering is able to learn cluster structures much better than pure clustering methods [2]. A variety of studies have attempted to learn models using weak supervision [5,12,13,26], some works [25] have also recently advanced deep learning to weak supervision, but existing methods are primarily focused on data in the Euclidean space. While both weakly supervised learning and graph data have been extensively studied,

to the best of our knowledge, there is no existing work on weakly supervised learning for graph-structured data, especially for constrained graph learning.

For constrained graph learning, the purpose is to integrate constraints as weak supervision knowledge to learn graph models. One example is constrained graph clustering (as shown in Fig. 1, which aims to cluster nodes of an attributed graph where each node is associated with a set of feature attributes. In specific applications such as the clustering of faces in videos [23] and the assessing of interpatient similarity [22] when class labels are not available, constraints are particularly important for enhancing performance. Another example is constrained graph classification (as shown in Fig. 1), which learns a binary classifier to identify whether two given nodes belong to the same group or not. Like recommendation systems, the predicted pairwise constraints (pairwise association rules) are very important in suggesting relevant items to users (as known as pairwise preference learning) [8].

The above observations motivate our research to propose a weak supervision node embedding model (WSNE) for constrained graph learning. We consider pairwise node constraints as weak supervision, and the main idea is to learn optimal node embedding by simultaneously integrating constraint loss and graph reconstruction loss in a deep graph convolution network (GCN) and variational graph auto-encoder (VGAE) combined framework. Different from existing strong-supervision graph embedding methods (including supervised/semi-supervised), our approach can utilize both constrained and unconstrained nodes to derive a high-order embedding evaluation criterion. This evaluation criterion estimates the effectiveness of node embedding based upon the high-level distances of must-link/cannot-link among constrained nodes and the average squared distances between unconstrained nodes. The GCN framework is used to learn a target node's representation by propagating neighbour information in an iterative manner until a stable fixed point is reached. And the VGAE framework is used to learn latent node representations through reconstructing graph topology information such as the graph adjacency matrix. To make better use of the constraints in graph learning, we also propose a novel topology optimization to fully utilize the potential constraint information during the message passing process in GCN as the given network topology may induce a performance degradation if it is directly employed in classification/clustering tasks.

In summary, the main contribution of the paper, compared to existing methods in the field, is threefold:

- We formulate a new weak-supervision network embedding task to utilize weak supervision knowledge to find effective latent vector space.
- We propose a new pairwise constraint evaluation criterion to efficiently evaluate the quality of embedding vector on constrained and unconstrained graph data.
- We develop a new constrained topology optimization method for graph convolution layers which take must-link and cannot-link into consideration.

2 Problem Definition

An undirected connected attributed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{X}\}$ consists of a set of nodes $\mathcal{V}$ with $|\mathcal{V}| = n$, a set of edges $\mathcal{E}$ with $|\mathcal{E}| = m$, the adjacency matrix $\mathcal{A}$, and node attribute matrix $\mathcal{X}$. If there is an edge between node i and node j, the entry $\mathcal{A}_{ij}$ denotes the weight of the edge; otherwise, $\mathcal{A}_{ij} = 0$. For unweighted graphs, we simply set $\mathcal{A}_{ij} = 1$.

For each node, its content (features) is represented as a vector $x \in \mathbb{R}^n$, where x_i denotes feature values of node i (Node attributes, node content, and node features are equivalent terms in this paper). Therefore, $\mathcal{X} \in \mathbb{R}^{n \times d}$ denotes the node attribute matrix of the graph, and the columns of $\mathcal{X}$ are the d features of the graph.

Network embedding aims to embed a graph $\mathcal{G}$ in a low-dimensional space $\mathcal{Z} \in \mathbb{R}^{n \times m}$, where $m << d$ and the columns of $\mathcal{Z}$ are the m embedded signals of the graph. In the context of constrained graph learning, we consider two pairwise node-level constraints as weak supervision knowledge:

- **Must-link**: Two nodes belong to be in the same cluster/group, *i.e.* $\mathcal{C}^+ = \{< i, j > | \mathcal{C}_i = \mathcal{C}_j\}$, where $\mathcal{C}_k$ means the cluster that node k belongs to.
- **Cannot-link**: Two nodes do not belong to the same cluster/group, *i.e.* $\mathcal{C}^- = \{< i, j > | \mathcal{C}_i \neq \mathcal{C}_j\}$.

Given graph $\mathcal{G}$ and weak supervision constraints $\{\mathcal{C}^+, \mathcal{C}^-\}$, constrained graph learning **aims** to solve learning sub-tasks as follows:

- **Constrained graph clustering:** incorporate pairwise node constraints to cluster nodes into different groups.
- **Constrained graph classification:** incorporate constraints $\{\mathcal{C}^+, \mathcal{C}^-\}$ to learn a binary classifier to classify a pair of nodes as either same-group or different-group.

For constrained graph learning, our theme is to impose constraints on network embedding process to learn a discriminative representation for each node. The pairwise constraints $\{\mathcal{C}^+, \mathcal{C}^-\}$ define transitive binary relations over the nodes. Consequently, when making use of constraints, we take a transitive closure over the constraints. The full set of derived constraints is then presented to the learning algorithm.

3 CEL: Constraint Embedding Loss

Because class labels are not available for individual nodes, we have to design a new approach to utilize constraints $\{\mathcal{C}^+, \mathcal{C}^-\}$ for node embedding learning. In this section, we address the weak supervision problem discussed in Sect. 1 by finding optimal node embedding of weak supervision from both must-link and cannot-link constraints. From the perspective of weak supervision, we assume that the optimal node embeddings should have the following properties: (1) *Must-link distance*: The pairwise must-link nodes should be close to each other

in the node embedding space; (2) *Cannot-link distance*: The pairwise cannot-link nodes should be far away from each other in the node embedding space; (3) *Separability*: unconstrained nodes should be able to be separated from each other in the node embedding space.

Intuitively, (1) and (2) only consider the constraints from constrained pairs and tend to optimize the node embeddings based on the constraints. This is motivated by the commonly observed phenomenon [17] that nodes close to each other tend to share common content information. Note (3) incorporates the distribution of unconstrained nodes, and tends to optimize the node embeddings that can separate nodes far from each other. It is similar to the PCA's assumption, which is expressed as the average squared distance between unlabelled samples.

Based upon the above properties, we derive a new evaluation criterion $\mathcal{L}_{CEL}(\mathcal{Z})$, for a given node embedding matrix $\mathcal{Z}$ as follow:

$$\begin{aligned}\mathcal{L}_{CEL}(\mathcal{Z}) = \quad & \frac{\gamma^+}{2|\mathcal{C}^+|}\sum_{<i,j>\in\mathcal{C}^+}(\mathcal{Z}_{i\cdot}-\mathcal{Z}_{j\cdot})^2 - \frac{\gamma^-}{2|\mathcal{C}^-|}\sum_{<i,j>\in\mathcal{C}^-}(\mathcal{Z}_{i\cdot}-\mathcal{Z}_{j\cdot})^2 \\ & - \frac{1}{2|\mathcal{C}^u|^2}\sum_{<i,j>\in\mathcal{C}^u}(\mathcal{Z}_{i\cdot}-\mathcal{Z}_{j\cdot})^2\end{aligned} \tag{1}$$

In Eq. (1), $\mathcal{C}^u = \{< i,j > \mid < i,\cdot > \notin \mathcal{C}^+ \cup \mathcal{C}^- \ and \ < j,\cdot > \notin \mathcal{C}^+ \cup \mathcal{C}^-\}$ is the pairwise unconstraint sets. γ^+ and γ^- are two parameters, which control the weights of the three types of constraints. Based on the experiments, we found that applying feature binarization after 0–1 scaling on $\mathcal{Z}$ for calculating $\mathcal{L}_{CEL}$ can accelerate the convergence during the training process. By defining a pairwised constraint matrix $P = [P_{ij}]^{n\times n}$ as

$$P_{ij} = \begin{cases} \gamma^+/|\mathcal{C}^+| & if \ < i,j > \in \mathcal{C}^+ \\ -\gamma^-/|\mathcal{C}^-| & if \ < i,j > \in \mathcal{C}^- \\ -1/2|\mathcal{C}^u|^2 & if \ < i,j > \in \mathcal{C}^u \\ 0 & otherwise \end{cases} \tag{2}$$

We can then rewrite the $\mathcal{L}_{CEL}(\mathcal{Z})$ in Eq. 1 as follow:

$$\mathcal{L}_{CEL}(\mathcal{Z}) = \frac{1}{2}\sum_{i,j}(\mathcal{Z}_{i\cdot}-\mathcal{Z}_{j\cdot})^2 P_{ij} = tr(\mathcal{Z}^\top(D^p - P)\mathcal{Z}) = tr(\mathcal{Z}^\top L^p \mathcal{Z}) \tag{3}$$

where $tr(\cdot)$ is the trace of a matrix, D^p is the diagnal matrix whose entries are column sums of P, i.e. $D^p_{ii} = \sum_j P_{ij}$. $L^p = D^p - P$ is a Laplacian matrix.

Therefore, the framework to optimize the node embeddings for graph learning by considering constraints is to minimize $\mathcal{L}_{CEL}(\mathcal{Z})$ during the training process.

4 WSNE for Constrained Graph Learning

The above constraint embedding loss $\mathcal{L}_{CEL}$ allows us to quantify embedding loss without knowing labels of individual nodes. In this paper, we incorporate

$\mathcal{L}_{CEL}$ into a variational graph encoder framework (GVAE) architecture which uses constrained graph convolution to build the encoder. Our goal is to generate embedding vectors which optimally preserve graph content and topology, as well as comply to the given constraints.

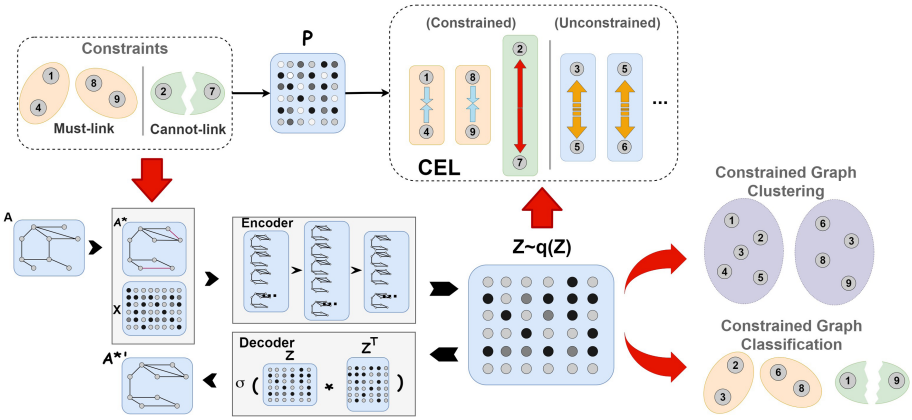

Fig. 2. The architecture of the weak supervision node embedding (WSNE) for constrained graph learning. The lower tier is the graph convolution based variational auto-encoder that reconstructs a graph $\mathcal{A}$ from $\mathcal{Z}$ which is generated by the encoder which exploits graph structure $\mathcal{A}$ and the node content matrix $\mathcal{X}$. The upper tier is the constraint embedding loss that evaluate the quality of $\mathcal{Z}$ on constrained and unconstrained graph data. The generated node embedding $\mathcal{Z}$ can be used for both constrained graph clustering and classification as right tier.

Variational Graph Auto-encoder. Graph auto-encoders (GAE) [11,19,21] are a family of models aiming at embed a graph in a low-dimensional space from which reconstructing (decoding) the graph should be possible. More precisely, the node embedding matrix $\mathcal{Z}$ is usually the output of a graph neural network (GNN) [3,6] processing $\mathcal{A}$. To reconstruct the graph, GAE stack an inner product decoder to this GNN. We have $\bar{\mathcal{A}} = \sigma(\mathcal{Z}\mathcal{Z}^\top)$, with $\sigma(\cdot)$ denoting the sigmoid function: $\sigma(x) = 1/(1+e^{-x})$. Therefore, the larger the inner product $\bar{\mathcal{A}}_{ij}$ in the embedding, the more likely nodes i and j are connected in $\mathcal{G}$ according to the GAE. Weights of the GNN are trained by gradient descent to minimize a reconstruction loss capturing the similarity of $\mathcal{A}$ and $\bar{\mathcal{A}}$, usually formulated as a weighted cross entropy loss.

VGAE extended the variational auto-encoder framework [9] to graph structure, which uses a probabilistic model involving latent variables z_i for each node $i \in \mathcal{V}$, interpreted as node representations in an embedding space. The inference model, i.e. the encoding part of the VAE, is defined as:

$$q(\mathcal{Z}|\mathcal{X},\mathcal{A}) = \prod_{i=1}^{n} q(z_i|\mathcal{X},\mathcal{A}) \tag{4}$$

where $q(z_i|\mathcal{X},\mathcal{A}) = \mathcal{N}(z_i|\mu_i, diag(\sigma_i^2))$. Gaussian parameters are learned from two GNNs, i.e. $\mu = GNN_\mu(\mathcal{X},\mathcal{A})$, with μ the matrix stacking up mean vectors μ_i; likewise, $log\sigma = GNN_\sigma(\mathcal{X},\mathcal{A})$. Latent vectors z_i are samples drawn from this distribution. From these vectors, a generative model aims at reconstructing (decoding) $\mathcal{A}$, leveraging inner products: $p(\mathcal{A}|\mathcal{Z}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\mathcal{A}_{ij}|z_i, z_j)$, where $p(\mathcal{A}_{ij} = 1|z_i, z_j) = \sigma(z_i^T z_j)$. During training, GNN weights are tuned by maximizing a tractable variational lower bound (ELBO) of the model's likelihood by gradient descent, with a Gaussian prior on the distribution of latent vectors, and using the reparameterization trick from [9]. Formally, for VGAE, we minimize the reconstruction error of the graph data by:

$$\mathcal{L}_R(\mathcal{Z}) = \mathbb{E}_{q(Z|\mathcal{X},\mathcal{A})}[\log p(\mathcal{A}|\mathcal{Z})] - D_{KL}[q(\mathcal{Z}|\mathcal{X},\mathcal{A})||p(\mathcal{Z})] \tag{5}$$

where $D_{KL}(\cdot||\cdot)$ is the KL divergence of the approximate from the true posterior.

WSNE: Weak Supervision Network Embedding. To better handle weak-supervision network embedding tasks, we incorporate $\mathcal{L}_{CEL}$ into the VGAE framework, in which the evaluation criterion $\mathcal{L}_{CEL}$ acts on the node embedding of the Encoder as part of the loss function to minimize together with the reconstruction loss. Figure 2 shows an overview of the proposed architecture. Therefore the whole network parameters are jointly trained by minimizing the following loss function as

$$\mathcal{L} = \mathcal{L}_R(\mathcal{Z}) + \lambda\mathcal{L}_{CEL}(\mathcal{Z}) \tag{6}$$

where $\mathcal{L}_R$ and $\mathcal{L}_{CEL}$ are defined in Eq. (3), respectively. Parameter $\lambda \geq 0$ is a tradeoff parameter. It is noted that, when $\lambda = 0$, the model is regressed to an original VGAE.

The proposed architecture can be used for both constrained graph clustering and classification tasks by using the learned node embeddings $\mathcal{Z}$.

Constrained Graph Clustering. We apply the linear kernel $\mathcal{K} = \mathcal{Z}\mathcal{Z}^\top$, and calculate the similarity matrix $\mathcal{S} = \frac{1}{2}(|\mathcal{K}|+|\mathcal{K}^\top|)$, where $|\cdot|$ means taking absolute value of each element of the matrix. Finally, we perform spectral clustering on $\mathcal{S}$ to obtain clustering results by computing the eigenvectors associated with the m largest eigenvalues of $\mathcal{S}$ and then applying the COP-k-means algorithm on the eigenvectors to obtain clusters.

Constrained Graph Classification. Given two nodes u and v, we define a binary operator $\otimes$ over the corresponding node embeddings $\mathcal{Z}_{u\cdot}$ and $\mathcal{Z}_{v\cdot}$ in order to generate a representation $g(u,v) \in \mathbb{R}^m$, which is the representation of the node pair $< u, v >$. We want our operator to be generally defined for any pair of nodes, even if an edge does not exist between the pair since doing so makes the representations useful for must-link/cannot-link prediction where our test set contains both true (must-link) and false (cannot-link) constraints. In this paper, we consider the Hadamard product used in [7] as the binary operator, i.e. $[g(u,v)]_i = [\mathcal{Z}_{u\cdot} \otimes \mathcal{Z}_{v\cdot}]_i = \mathcal{Z}_{ui} \times \mathcal{Z}_{vi}$. The new pairwise representation is then used to learn a binary classifier for constrained graph classification on the given constraints.

5 Constraint Assisted Topology Optimization

The given network topology induce a performance degradation if it is directly employed in classification/clustering, because it may possess high sparsity and certain noises. To make the message passing more efficient, in this section, we optimize the graph topology based upon the constraints and the GCN architecture.

Graph Convolutional Networks. Graph Convolutional Networks (GCNs) [10] achieve promising generalization in various tasks and our work is built upon the GCN module. At layer i, taking graph adjacency matrix $\mathcal{A}$ and hidden representation matrix $H^{(i)}$ as input, each GCN module outputs a hidden representation matrix $H^{(i+1)}$, which is described as:

$$H^{(i+1)} = ReLU(\hat{D}^{-\frac{1}{2}}\hat{\mathcal{A}}\hat{D}^{-\frac{1}{2}}H^{(i)}W^{(i)}) \tag{7}$$

where $H^{(0)} = \mathcal{X}$, $ReLU(a) = max(0, a)$, adjacency matrix with self-loop $\hat{\mathcal{A}} = \mathcal{A}+I$ (I is an identity matrix), $\hat{D}$ is the degree matrix of $\hat{\mathcal{A}}$, and $W^{(i)}$ is a trainable weight matrix. Then the output node embedding $\mathcal{Z} = H^{(K)}$ and $(K+1)$ is the number of layers in the network architecture.

Topology Optimization. Graph convolutional networks collectively aggregate information from graph structure, and model input and/or output consisting of elements and their dependency. The graph structure (edges) used in graph convolution architectures represents a kind of relations between nodes and guides the message passing among nodes. For example, in citation networks, a citation link between two documents is treated as an edge. While in social networks, the edges represent the interactions between users. From this point of view, the must-link/cannot-link constraints $\{\mathcal{C}^+, \mathcal{C}^-\}$ also represent a high-order relation between nodes which can be used to build the graph structure for constrained graph learning.

In order to directly utilize constraints $\{\mathcal{C}^+, \mathcal{C}^-\}$ into graph learning process, we will update the adjacency matrix $\mathcal{A}$ by using must-link and cannot-link constraints and use the updated $\mathcal{A}^*$ for graph convolution. The updated $\mathcal{A}^*$ is as follow:

$$\mathcal{A}^*_{ij} = \begin{cases} 1 & if \ < i, j > \in \mathcal{C}^+ \\ 0 & if \ < i, j > \in \mathcal{C}^- \\ \mathcal{A}_{ij} & otherwise \end{cases} \tag{8}$$

Using updated $\mathcal{A}^*$ in graph covolutions allows direct message passing between must-link nodes as they are in the same cluster and should have node embeddings consisting of elements. Using $\mathcal{A}^*$ also rejects direct message passing between cannot-link nodes as they are in different clusters and are not recommended to share similar information. By using $\mathcal{A}^*$, the GCN module should be updated accordingly as follow:

$$H^{(i+1)} = ReLU(\hat{D}^{*-\frac{1}{2}}\hat{\mathcal{A}}^*\hat{D}^{*-\frac{1}{2}}H^{(i)}W^{(i)}) \tag{9}$$

Algorithm 1: WSNE for constrained graph learning

Data: $\mathcal{A}$: The adjacency matrix; $\mathcal{X}$: The node attribute matrix; $\mathcal{C}^+$ and $\mathcal{C}^-$: The constraints;
Result: $\mathcal{Z}$: The node embedding matrix; $\mathcal{M}$: The cluster partition; and $\mathcal{Y}$: The binary classification;

$P \leftarrow$ Generate constraint matrix P through Eq. (2);
$\mathcal{A}^* \leftarrow$ Constraint assisted topology optimization through Eq. (8);
while *Convergence* **do**
  $\mathcal{Z} \leftarrow$ Generate latent variables $\mathcal{Z}$ using Eq. (4) and graph convolution framework in Eq. (9);
  $\mathcal{L}_R(\mathcal{Z}) \leftarrow \mathbb{E}_{q(Z|\mathcal{X},\mathcal{A})}[\log p(\mathcal{A}|\mathcal{Z})] - KL[q(\mathcal{Z}|\mathcal{X},\mathcal{A})||p(\mathcal{Z})]$;
  $\mathcal{L}_{CEL}(\mathcal{Z}) \leftarrow tr(\mathcal{Z}^\top L^p \mathcal{Z})$;
  $\mathcal{L} \leftarrow \mathcal{L}_R(\mathcal{Z}) + \lambda \mathcal{L}_{CEL}(\mathcal{Z})$ update variational autoencoder with its gradient ;
end
if *Constrained graph clustering* **then**
  Apply the linear kernel $\mathcal{K} = \mathcal{Z}\mathcal{Z}^\top$, and calculate the similarity matrix $\mathcal{S} = \frac{1}{2}(|\mathcal{K}| + |\mathcal{K}^\top|)$;
  Obtain the cluster partition $\mathcal{M}$ by performing spectral clustering on $\mathcal{S}$;
end
if *Constrained graph classification* **then**
  $[g(u, v)] = \mathcal{Z}_{u\cdot} \times \mathcal{Z}_{v\cdot}$;
  Train a binary classifier on training $\mathcal{C}^+$ and $\mathcal{C}^-$ with new representation $[g(u, v)]$ and obtain the classification results $\mathcal{Y}$;
end

In this paper, we use graph convolution to process $\mathcal{A}$ and VGAE to reconstruct the graph structure for minimizing the information loss during the node embedding. Algorithm 1 lists detailed procedures of constrained graph learning.

6 Experiments

We evaluate our method on three benchmark graph datasets for both constrained graph clustering and classification tasks. **Cora**, **Citeseer** and **Pubmed** [10] are citation networks where nodes correspond to publications and are connected if one cites the other. The nodes in **Cora** and **Citeseer** are associated with binary word vectors, and nodes in **Pubmed** are associated with tf-idf weighted word vectors. Table 1 summarizes the details of the datasets. For Cora and Citeseer datasets, we randomly select 400 pairwise constraints (200 must-link pairs and 200 cannot-link pairs, respectively) as weak supervision. While for Pubmed dataset, we randomly select 3,000 pairwise constraints with 1,500 must-link pairs and 1,500 cannot-link pairs, respectively.

6.1 Constrained Graph Clustering

Baselines. As there is no existing constrained graph clustering method. We compare both embedding based approaches as well as approaches directly for graph clustering using constrained k-means for obtaining clustering results.

- **COP-k-means**: the constrained k-means algorithm [20] uses constraints as knowledge to restrict the data assignment process of the original k-means algorithm.
- **Spectral Clustering**: [18] is an effective approach for learning social embedding.
- **DeepWalk**: [16] is a network representation approach which encodes social relations into a continuous vector space.
- **GAE/VGAE**: [11] are (variational) autoencoder-based unsupervised frameworks for graph data, which naturally leverages both topological and content information.
- **ARVGA**: [15] is an adversarially regularized variational graph autoencoder for learning the node embedding.

Metrics. We employ three metrics to validate the clustering results: Accuracy (Acc), Normalized Mutual Information (NMI) and Average Rand index (ARI).

Parameter Settings. For the Cora, Citeseer and Pubmed datasets, we train all autoencoder-related models for 200 iterations and optimize them with the Adam algorithm. The learning rate is set to 0.001 and $\lambda = 0.1$. The parameters in $\mathcal{L}_{CEL}$ are set to $\alpha = \beta = 1$. We construct encoders with a 32-neuron hidden layer and a 16-neuron embedding layer for all the experiments. For the rest of the baselines, we retain the settings described in the corresponding papers.

Experimental Results. The constrained graph clustering results on the Cora, Citeseer and Pubmed data sets are given in Table 2. The results show that by incorporating the effective constraint embedding loss and constraint assisted topology optimization into our variational graph convolutional auto-encoder, WSNE achieve outstanding performance on all three metrics. Compared with the baselines, WSNE increased the Acc score from around 5.6% compared with existing node embeding methods incorporating with COP-k-means and 2.6% increased on the NMI score.

6.2 Constrained Graph Classification

Baselines. Because there is no constrained graph classification method available for comparison, we use each node embedding method, including DeepWalk, GAE, VGAE, ARVGA, and WSNE, to find node embedding. After that, we use embedding to generate vector $g(u, v)$ for node pair $< u, v >$, using constraints $\mathcal{C}^+$ and $\mathcal{C}^-$ (as we described in Sect. 4). Then we train binary classifiers using $g(u, v)$ generated from each embedding method, and report their performance in Table 3.

Table 1. Benchmark network statistics

Dataset	*# Nodes*	*# Edges*	*# Features*	*# Classes*
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3

Table 2. Constrained graph clustering results on Cora, Citeseer and Pubmed.

Methods	CORA			CITESEER			PUBMED		
	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
COP-k-means	0.397	0.233	0.210	0.423	0.212	0.197	0.563	0.289	0.276
Spectral clustering	0.425	0.287	0.197	0.427	0.239	0.166	0.526	0.247	0.251
DeepWalk + COP-k-means	0.449	0.324	0.142	0.371	0.175	0.096	0.604	0.274	0.203
GAE + COP-k-means	0.557	0.406	0.290	0.451	0.277	0.213	0.627	0.269	0.175
VGAE + COP-k-means	0.570	0.424	0.332	0.471	0.259	0.124	0.615	0.193	0.095
ARVGA + COP-k-means	0.617	0.459	**0.373**	0.575	0.330	0.326	0.592	0.307	0.221
WSNE	**0.652**	**0.471**	**0.373**	**0.636**	**0.424**	**0.380**	**0.655**	**0.315**	**0.311**

Metrics. We report the results in terms of AUC score (the area under a receiver operating characteristic curve). The training set for the binary classification tasks are the provided constraints and the testing set contains 1,000 pairwise constraints for Cora and Citeseer datasets, and 5,000 pairwise constraints for Pubmed dataset to verify the performance.

Experimental Results. The constrained graph classification results on the Cora, Citeseer and Pubmed data sets are given in Table 3. The results show that WSNE achieves a significant improvement on the AUC score compared to all other baselines.

Table 3. Constrained graph classification results (AUC) on Cora, Citeseer, and Pubmed networks.

Methods	AUC values		
	Cora	Citeseer	Pubmed
DeepWalk	0.679	0.624	0.703
GAE	0.772	0.694	0.797
VGAE	0.790	0.738	0.823
ARVGA	0.793	0.754	0.836
WSNE	**0.844**	**0.802**	**0.871**

6.3 Embedding Visualization

We also visualize the Cora data in a two-dimensional space by applying the t-SNE algorithm [14] on the learned embedding. The results in Fig. 3 validate that by applying weak supervision constraints, *WSNE* is able to learn a more discriminative embedding vectors from graph data.

Fig. 3. Visualization comparison of embedding vectors on Cora data ($t-$SNE). From left to right: embeddings from DeepWalk, GAE, VGAE, ARVGA, and WSNE. Each point denotes a node. Nodes are color-coded based on the ground-truth class they belonging to (there are 7 classes/groups in total). The digit shows the centroid of each group, reported from $t-$SNE.

7 Conclusion

In this paper, we study a new research problem of weak supervision network embedding for constrained graph learning. We argued that existing network embedding approaches either require label information for individual nodes (strong-supervision) or do not use node labels (non-supervision). Weak supervision, such as constraints, are useful domain knowledge, but cannot be utilized in existing network embedding methods. To address the challenge, we proposed a new constraint embedding loss to quantify latent embedding vectors' loss by using both constrained and unconstrained data. Then we integrated this loss in a graph convolutional neural network and Graph Auto-Encoders combined framework to jointly model graph structures and node attributes to learn discriminative embedding vectors. Experiments and comparisons on real-world tasks show that the proposed method can effectively utilize weak supervision knowledge for constrained graph clustering and classification tasks.

Acknowledgements. This research is sponsored in part by the U. S. National Science Foundation (NSF) through Grant Nos. IIS-1763452 & CNS-1828181.

References

1. Basu, S., Davidson, I.: Clustering with constraints: theory and practice. In: KDD Tutorial (2006)
2. Basu, S., Davidson, I., Wagstaff, K.: Constrained Clustering: Advances in Algorithms, Theory, and Applications. CRC Press, Boca Raton (2008)

3. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
4. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: problems, techniques, and applications. TKDE **30**(9), 1616–1637 (2018)
5. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML, pp. 209–216 (2007)
6. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, pp. 3844–3852 (2016)
7. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
8. Hüllermeier, E., Fürnkranz, J., Cheng, W., Brinker, K.: Label ranking by learning pairwise preferences. Artif. Intell. **172**(16–17), 1897–1916 (2008)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
10. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
11. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
12. Kulis, B., Sustik, M.A., Dhillon, I.S.: Low-rank kernel learning with Bregman matrix divergences. J. Mach. Learn. Res. **10**, 341–376 (2009)
13. Liu, W., Ma, S., Tao, D., Liu, J., Liu, P.: Semi-supervised sparse metric learning using alternating linearization optimization. In: KDD, pp. 1139–1148 (2010)
14. van der Maaten, L.: Accelerating t-SNE using tree-based algorithms. J. Mach. Learn. Res. **15**(1), 3221–3245 (2014)
15. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. IJCAI (2018)
16. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: KDD, pp. 701–710 (2014)
17. Reagans, R., McEvily, B.: Network structure and knowledge transfer: the effects of cohesion and range. Adm. Sci. Q. **48**(2), 240–267 (2003)
18. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Disc. **23**(3), 447–478 (2011)
19. Tian, F., Gao, B., Cui, Q., Chen, E., Liu, T.Y.: Learning deep representations for graph clustering. In: AAAI (2014)
20. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al.: Constrained k-means clustering with background knowledge. In: ICML, pp. 577–584 (2001)
21. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: KDD, pp. 1225–1234 (2016)
22. Wang, F., Sun, J., Ebadollahi, S.: Integrating distance metrics learned from multiple experts and its application in patient similarity assessment. In: ICDM, pp. 59–70 (2011)
23. Wu, B., Zhang, Y., Hu, B.G., Ji, Q.: Constrained clustering and its application to face clustering in videos. In: CVPR, pp. 3507–3514 (2013)
24. Zhang, D., Yin, J., Zhu, X., Zhang, C.: Network representation learning: a survey. IEEE Trans. Big Data **6**, 3–28 (2020)
25. Zhang, H., Basu, S., Davidson, I.: A framework for deep constrained clustering - algorithms and advances. In: Proceedings of ECML/PKDD, pp. 57–72 (2019)
26. Zhou, Z.H.: A brief introduction to weakly supervised learning. Natl. Sci. Rev. **5**(1), 44–53 (2017)