Toward Unsupervised Graph Neural Network: Interactive Clustering and Embedding via Optimal Transport

Liang Yang, Junhua Gu
School of Artificial Intelligence Hebei University of Technology Tianjin, China

Chuan Wang, Xiaochun Cao
SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

Lu Zhai, Di Jin
College of Intelligence and Computing, Tianjin University, Tianjin, China

Yuanfang Guo
School of Computer Science and Engineering, Beihang University, Beijing, China

***Abstract*—Most of the existing Graph Neural Networks (GNNs) are deliberately designed for semi-supervised learning tasks, where supervision information (labelled node) is utilized to mitigate the oversmoothing problem of message passing. Unfortunately, the oversmoothing problem tends to be more severe in unsupervised tasks, since supervision information is not available. Since community structure/cluster is an essential characteristic of network, a natural approach to reduce the oversmoothing problem is to also constrain the node embeddings to maintain their own characteristics to prevent all the node embeddings from becoming too similar to be distinguished. In this paper, a novel Optimal Transport based Graph Neural Network (OT-GNN) is proposed to overcome the oversmoothing problem in unsupervised GNNs by imposing the equal-sized clustering constraints to the obtained node embeddings. To solve the combinatorial optimization problem, the constrained objective function of unsupervised GNN is relaxed to an Optimal Transport problem, and a fast version of the Sinkhorm-Knopp algorithm is adopted to handle large networks. Extensive experiments on node clustering and classification demonstrate the superior performance of our proposed OT-GNN.**

***Keywords*-graph neural network; network embedding; unsupervised learning; node clustering;**

I. Introduction

Most Graph Neural Networks (GNNs) [1], [2] are deliberately designed for (transductive and inductive) semi-supervised learning tasks, where supervision information (labelled nodes) plays the important role of enhancing the distinguishability. In fact, Laplacian smoothing (spatial perspective) [3] and low-pass filtering (spectral perspective) [4], which have been accepted to be the reason for the success of Graph Convolutional Network (GCN) [5], also significantly reduce the expressive power, i.e. data distinguishability, due to the over-smoothing problem [6]–[8]. Thus, the supervision information is utilized to mitigate this adverse effect. Unfortunately, supervision information cannot fundamentally solve this problem for the following two reasons. First and foremost, supervision information is not available in unsupervised tasks, such as network embedding, node clustering and link prediction. Most existing unsupervised GNNs either reconstruct the original information (adjacency matrix and attribute matrix) [9]–[11] or maximize mutual information [12], [13] to retain as much information as possible. Thus, the oversmoothing problem tends to be more severe for unsupervised GNNs than for semi-supervised ones. Second, although learning the message aggregation from labelled nodes can alleviate smoothing problem, it also induces serious overfitting problem [14], [15], which also seriously effects the performance.

In this paper, a novel Optimal Transport based Graph Neural Network (OT-GNN) is proposed to overcome the oversmoothing problem in unsupervised GNNs, which constrains all the node embeddings to be similar. Since community structure/cluster is an essential characteristic of network [16], a natural approach to reduce the oversmoothing problem is to also constrain the node embeddings to maintain their own characteristics to prevent all the node embeddings from becoming too similar to be distinguished. According to this intuition, OT-GNN elegantly trains the GNNs by imposing the clustering constraints to the obtained node embeddings. Specifically, the objective function of unsupervised GNNs is firstly derived from the cross-entropy loss of the semi-supervised node classification. This objective function can be optimized with respect to the model parameters and node labels. Unfortunately, it leads to a degenerate solution which assigns all the nodes into a single class due to the oversmoothing problem. Then, to prevent this degenerate solution, clustering constraints are added to ensure the nodes being uniformly classified into classes of equal size. To solve the formulated combinatorial optimization problem, the constrained objective function of unsupervised GNN is relaxed to an Optimal Transport problem, which can be then solved via linear programming in a polynomial time. To speedup this process on large networks, a fast version of the Sinkhorm-Knopp algorithm, which employs a regularization term to the loss function, is adopted, and an iterative algorithm is proposed with additional complexity proportional to the network size. Besides of being employed to train the GNNs, such as Graph Convolutional Network (GCN) [5] and Graph Attention Network (GAT) [17] for node embedding and clustering in an unsupervised manner, OT-GNN can also be utilized to regularize other unsupervised GNNs, such as Graph AutoEncoder (GAE) [9], for the link prediction task.

2374-8486/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDM50108.2020.00177

The main contributions of this paper are summarized as follows:

- To overcome the oversmoothing problem in unsupervised GNNs, we propose a novel Optimal Transport based Graph Neural Network (OT-GNN) with interactive node embedding and clustering.
- To efficiently train the unsupervised GNNs, we relax its constrained objective function to an Optimal Transport problem and solve it via a fast version of the Sinkhorm-Knopp algorithm.

II. Preliminaries

In this section, the notations are first given. Then, the basic concepts in Graph Neural Networks are provided.

A. Notations

A network can be represented by an attributed graph $G = (V, E, X)$. $V = \{v_i | i = 1, ..., N\}$ is a set of $|V| = N$ vertices, where v_i is associated with a feature $x_i \in \mathbb{R}^K$. $X \in \mathbb{R}^{K \times N}$ represents the collection of the features. Each column of X corresponds to a node. E stands for a set of edges. Each of the edges connects two vertices in V. The adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{N \times N}$ is obtained according to the network topology. If an edge connects the vertices v_i and v_j, $a_{ij} = 1$, and vice versa. If self-edges are allowed in the network, then $a_{nn} = 1$. Otherwise $a_{nn} = 0$. $d_n = \sum_j a_{nj}$ stands for the degree of v_n while $D = \text{diag}(d_1, d_2, ..., d_N)$ is the degree matrix of A. The graph Laplacian and its normalized form are defined as $L = D - A$ and $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, respectively.

For semi-supervised node classification task, a set of vertices $V_l \subset V$ are labelled. $y_i \in \{1, 2, 3..., F\}$ is utilized to represent the label of vertex $v_i \in V_l$, where F is the number of classes. A typical semi-supervised node classification algorithm classifies other nodes in $V - V_l$. For simplicity, the first $|V_l|$ nodes $\{v_i\}_{i=1}^{|V_l|}$ are assumed to be labelled, i.e., $\{y_1, y_2, ..., y_{|V_l|}\}$ are known.

B. Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are designed from either the spatial or spectral perspective. Spectral ones originate from spectral graph convolution, while spatial ones are designed along attribution propagation pipeline. By simplifying the time-consuming spectral graph convolution operation of existing spectral methods, Graph Convolutional Network (GCN) [5] defines the graph convolution operation as

$$T^{GCN} = \sigma(H^{GCN}) = \sigma(WX\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}), \quad (1)$$

where $\tilde{A} = A + I_N$ and $\tilde{D}_{nn} = \sum_j \tilde{A}_{nj} = d_n + 1$. W stands for the trainable weight matrix of a fully-connected layer. $\sigma(.)$ represents the nonlinear activation function, such as ReLU and softmax. $H^{GCN} \in \mathbb{R}^{K \times N}$ and T^{GCN} denotes the node representations (embeddings) before and after nonlinear activation function $\sigma(.)$, respectively. For clarity, H^{GCN} and T^{GCN} are called node feature and node representation, respectively. Eq. (1) can be formulated in a node-wise form as

$$\begin{aligned} t_i^{GCN} &= \sigma(h_i^{GCN}) \\ &= \sigma\Big(W \sum_{j \in N(i) \cup \{i\}} \frac{1}{\sqrt{(d_i+1)(d_j+1)}} x_j\Big), \end{aligned} \quad (2)$$

where h_i^{GCN} and t_i^{GCN}, the i^{th} columns of H^{GCN} and T^{GCN}, are the feature and representation of node v_i, respectively. $N(i)$ represents the neighbourhood of vertex v_i. Some recent work interprets GCN from smoothing and low-pass filtering. Li et al. [3] concludes the mechanism and success of GCN as performing a symmetric Laplacian smoothing operation before the actual predictions. The performance based on the smoothed attribute obviously outperforms that based on the original attributes. Wu et al. [4] reduce the unnecessary complexity and redundant computation of GCN to Simplified Graph Convolution (SGC) by successively removing nonlinearities and collapsing weight matrices between consecutive layers, and show that it corresponds to a fixed low-pass filter followed by a linear classifier. Although GCN and SGC significantly improve the performance, its main drawback is the fixed propagation weight $\frac{1}{\sqrt{(d_i+1)(d_j+1)}}$, which is completely determined by the degrees of the two connected nodes.

To overcome that drawback, Graph Attention Network (GAT) [17] attempts to learn the propagation weight by leveraging the self-attention mechanism as

$$\begin{aligned} \alpha_{ij} &= \text{softmax}\left(a(Wx_i, Wx_j)\right) \\ &= \frac{\exp\left(\text{LeakyReLU}(b^T[Wx_i || Wx_j])\right)}{\sum_{k \in N(i)} \exp\left(\text{LeakyReLU}(b^T[Wx_i || Wx_k])\right)}, \end{aligned} \quad (3)$$

where $a(.,.)$ stands for a neural network, $||$ denotes the concatenate operator and $b \in \mathbb{R}^{2F}$ is the learnable vector for propagation weights. Then the node representation in Eq. (2) can be enhanced to

$$t_i^{GAT} = \sigma(h_i^{GAT}) = \sigma\Big(W \sum_{j \in N(i)} \alpha_{ij} x_j\Big), \quad (4)$$

where both b and W are the trainable parameters. h_i and t_i are adopted to represent the node feature and embedding by ignoring the superscript, if we don't emphasize the methods used to obtain them.

1) Semi-supervised GCNs: To obtain the learnable parameters for (transductive and inductive) semi-supervised node classification task, GNNs can be trained by minimizing the cross-entropy between the predictions and given labels of the labelled vertices $v_i \in V_l$ as

$$\mathcal{L}_{sup} = -\sum_{v_i \in V_l} \sum_{y=1}^{F} \delta(y_i - y) \log(t_{iy}), \quad (5)$$

where t_{iy} is the y^{th} element of node embedding t_i. $\delta(.)$ is Dirac delta function and $\delta(y_i - y) = 1$ if and only if $y = y_i$, otherwise $\delta(y_i - y) = 0$. By setting the dimensionality of t_i to the number of classes F and adopting softmax as the nonlinearity function $\sigma(.)$, t_{iy} can be considered as the predicted probability of node v_i being classified to class y as

$$t_{iy} = p(y|h_i) = \text{softmax}_y(h_i) = \frac{\exp(h_{iy})}{\sum_k \exp(h_{ik})}. \quad (6)$$

III. Proposed Models

In this section, a novel unsupervised graph neural network (GNN), Optimal Transport based GNN (OT-GNN) is proposed. First, unsupervised GNN is derived from the semi-supervised one by adding clustering constraint to prevent oversmoothing, i.e., embedding collapsed. Then, the objective function of unsupervised GNN is interpreted from Optimal Transport perspective for efficient solution, and a fast version of Sinkhorn-Knopp algorithm is adopted to efficiently solve it.

A. Unsupervised Graph Neural Networks

In this subsection, we derive unsupervised graph neural network from the semi-supervised one in Section II-B1. By integrating the probability interpretation of t_{iy} (Eq. (6)) into the loss function of semi-supervised node classification (Eq. (5)), the objective function can be reformulated as

$$\mathcal{L}\left(p|\{y_1, y_2, ..., y_{|V_l|}\}\right) = -\sum_{v_i \in V_l} \log\left(p(y_i|h_i)\right), \quad (7)$$

where $\mathcal{L}\left(p|\{y_1, y_2, ..., y_{|V_l|}\}\right)$ denotes that node labels $\{y_1, y_2, ..., y_{|V_l|}\}$ are required to train the GNN $t_i = p(.|h_i)$ parameterized by W and b in Eq. (6). After training, parameters W and b in Eqs (2) and (4) are obtained. Thus the labels of unlabelled nodes $v_j \in V - V_l$ can be obtained from $p(y|h_j)$, and the objective function in Eq. (7) can be extended to

$$\begin{aligned} &\mathcal{L}\left(p, \{y_j\}_{j=|V_l|+1}^{|V|} | \{y_i\}_{i=1}^{|V_l|}\right) \\ &= -\sum_{v_i \in V_l} \log\left(p(y_i|h_i)\right) - \sum_{v_j \in V - V_l} \log\left(p(y_j|h_j)\right) \\ &= -\sum_{v_i \in V} \sum_{y=1}^{F} \delta(y_i - y) \log\left(p(y|h_i)\right), \quad (8) \end{aligned}$$

where the term corresponding to the unlabelled nodes, i.e. $\sum_{v_j \in V - V_l} \log\left(p(y_j|h_j)\right)$, equals to 0 and achieves its minimum, since y_j is obtained from $p(y|h_j)$ for unlabelled nodes. Therefore, semi-supervised node classification is achieved by interactively optimizing the objective function in Eq. (8) with respect to the model $p(y|h)$ parameterized by W and b and node labels $\{y_i\}_{i=1}^{|V|}$. This process works if and only if part of node labels $\{y_i\}_{i=1}^{|V_l|}$ are known and utilized to constrain the optimization.

Unfortunately, optimizing this objective function leads to a degenerate solution if all the nodes are unlabelled. Eq. (8) can be trivially minimized by training the model to assign all the nodes into a single class. To prevent this degenerate solution, constraints must be added to the objective function. To facilitate it, we first represent the labels as a posterior distribution $q(y|h_i)$ instead of a deterministic Dirac delta function $\delta(y_i - y)$, and reformulate Eq. (8) as

$$\mathcal{L}(p, q) = -\sum_{v_i \in V} \sum_{y=1}^{F} q(y|h_i) \log\left(p(y|h_i)\right), \quad (9)$$

where $q(y|h_i) \in \{0, 1\}$ and $\sum_y q(y|h_i) = 1$. Note that $q(y|h_i)$ is utilized to denote the posterior distribution of node v_i's label instead of indicating label y is learned from h_i as $p(y|h_i)$. Then, we aim to add some constraints to ensure nodes being classified into F classes. Since the distribution of cluster size is unknown, we simply assume that all nodes are uniformly classified into F clusters of equal size to prevent the degraded situation where most nodes are classified into one big cluster. Note that although the equal-sized clustering doesn't perfectly meet the ground-truth, it is an effective way to prevent all the node embeddings from being too similar. In experiments, the number of clusters, i.e., F , is set larger than the ground-truth, and the obtained embeddings are adopted for clustering instead of the assigned clusters. Thus, the objective function for unsupervised graph neural network can be written as

$$\begin{aligned} \arg\min_{q,p} \quad & -\sum_{v_i \in V} \sum_{y=1}^{F} q(y|h_i) \log\left(p(y|h_i)\right) \quad (10) \\ \text{s.t.} \quad & \forall y : q(y|h_i) \in \{0, 1\} \text{ and} \sum_{v_i \in V} q(y|h_i) = N/F, \end{aligned}$$

where the constraint $\sum_{v_i \in V} q(y|h_i) = N/F$ makes sure that there are N/F nodes are assigned to class y, thus all the nodes are equally partitioned. Unfortunately, this objective function is difficult to minimize, since it is a combinatorial optimization problem with respect to q.

B. Unsupervised GNNs as Optimal Transport

In this subsection, to address the difficulty in optimization, we interpret the objective function of unsupervised graph neural network, i.e., Eq. (10), with respective to the assignment q from the perspective of Optimal Transport. Let $P = [p_{yi}] \in \mathbb{R}_+^{K \times N}$ and $Q = [q_{yi}] \in \mathbb{R}_+^{K \times N}$ be the joint probability matrix estimated from the GNN and the joint probability of assignment, respectively, where

$$p_{yi} = p(y, h_i) = p(y|h_i)p(h_i) = p(y|h_i)\frac{1}{N}, \quad (11)$$

$$q_{yi} = q(y, h_i) = q(y|h_i)p(h_i) = q(y|h_i)\frac{1}{N}. \quad (12)$$

The loss function in Eq. (10) can be rewritten as

$$-\sum_{v_i \in V}\sum_{y=1}^{F} q(y|h_i)\log\left(p(y|h_i)\right) =< Q, -\log P > \quad (13)$$

where $\log$ is an element-wise operation. $< A, B >= \sum_i \sum_j a_{ij} b_{ij}$ is the Frobenius inner product of two matrices. To alleviate the difficulty of combinatorial optimization problem with respect to q, matrix Q is relaxed to belonging to transportation polytope [18] as

$$U(r, c) := \{Q \in \mathbb{R}_+^{F\times N} | Q\mathbf{1} = r,\ Q^T\mathbf{1} = c\}, \quad (14)$$

where $\mathbf{1}$ stands for the vector of all ones with appropriate dimensionality. r and c are the fixed vectors used to constrain the summarizations of Q along row and column, respectively. Since $Q = [q_{yi}]$ denotes the joint distribution of label y and node v_i, where each column corresponds to a vertex, $c \in \mathbb{R}^N$ is set as $\frac{1}{N}\mathbf{1}$. According to the equal-size cluster constraints $\sum_{v_i \in V} q(y|h_i) = N/F$ in Eq. (10) and $q_{yi} = q(y|h_i)\frac{1}{N}$ in Eq. (12), $r \in \mathbb{R}^K$ is set as $\frac{1}{F}\mathbf{1}$, i.e., $c = \frac{1}{N}\mathbf{1}, \quad r = \frac{1}{F}\mathbf{1}$. Thus, optimizing unsupervised GNNs in Eq. (10) with respective to the assignment Q can be relaxed to the following optimal transport problem [18]

$$\arg\min_{Q \in U(r,c)} < Q, -\log P >, \quad (15)$$

whose solution can be obtained via linear program problem in polynomial time.

C. Optimization

In this section, an algorithm, which iteratively updates q and p with another fixed, is proposed to minimize the objective function of unsupervised GNNs shown in Eq. (10). If GCN is adopted as the basic GNN, i.e., h_i is obtained from Eq. (2), the obtained unsupervised GNN is OT-GCN, while OT-GAT denotes that GAT is adopted as the basic GNN, i.e., h_i is obtained from Eq. (4).

1) Updating p with fixed q: If q is fixed, optimizing Eq. (10) with respective to p, which is parameterized by W and b, is equivalent to optimizing Eq. (5) or (7), where labels of all the nodes $\{y_i\}_{i=1}^{|V|}$ are assumed to have been obtained from q. It can be achieved via gradient descent as existing semi-supervised GNNs, such as GCN [5] and GAT [17].

2) Updating q with fixed p: If p is fixed, optimizing Eq. (10) with respective to q can be achieved via Optimal Transport. By connecting the unsupervised GNNs optimization in Eq. (10) with respective to the assignment q with the optimal transport problem in Eq. (15) in Section III-B, assignment can be obtained via linear programming in polynomial time. To further speedup this process on large networks, a fast version of the Sinkhorm-Knopp algorithm [19] is adopted. It employs a regularization term to the loss function of Eq. (15) as

$$\arg\min_{Q \in U(r,c)} < Q, -\log P > +\frac{1}{\lambda}\mathrm{KL}(Q||rc^T), \quad (16)$$

Table I
DATASETS.

Datasets	Nodes	Edges	Categories	Attributes
Texas	183	328	5	1,703
Cornell	195	304	5	1,703
Cora	2,708	5,429	7	1,433
Citeseer	3,312	4,732	6	3,703
Pubmed	19,729	44,338	3	500

where $\mathrm{KL}(.||.)$ stands for the Kullback-Leibler divergence. $rc^T \in \mathbb{R}^{K\times N}$, each element of which is $1/(NK)$, can be considered as the non-informative uniform prior distribution of Q. λ balances the convergence speed and the closeness of Eq. (16) to the optimal transport in Eq. (15). According to [19], optimizing Eq. (16) is equivalent to optimizing Eq. (15) with moderate value of λ. By introducing this regularization, the optimal Q can be analytically obtained as

$$Q = \mathrm{diag}(\gamma)P^{\lambda}\mathrm{diag}(\beta), \quad (17)$$

where the exponentiation P^{λ} is element-wise operation. γ and β are the two vectors used to ensure that the obtained Q is a probability matrix. $\mathrm{diag}(.)$ creates diagonal matrix from vector. According to [19], γ and β can be iteratively updated as

$$\gamma_y = [P^{\lambda}\beta]_y^{-1},\ \beta_i = [\gamma^T P^{\lambda}]_i^{-1}, \quad (18)$$

each of which consists of a matrix-vector multiplication with complexity $\mathcal{O}(NK)$. This complexity is linear with the size of the network. At beginning, β and γ are initialized as c and r, respectively. In experiments, just a few iterations, which require only $\mathcal{O}(NK)$ operations, are needed to obtain Q.

IV. EVALUATIONS

The experiments are conducted on five commonly utilized networks. Dataset statistics are summarized in Table I.

A. Baselines

To demonstrate the superiority of our proposed OT-GNN on representation learning, 7 state-of-the-art baselines are employed. These methods fall in two categories, network embedding methods and unsupervised graph neural networks. Network embedding methods include DeepWalk [20], node2vec [21], LINE [22] and GraRep [23]. The unsupervised GNNs includes (Variational) Graph AutoEncoder (VGAE) [9], Adversarially Regularized Graph Autoencoder (ARVGE) [10] and Deep Graph Infomax (DGI) [12].

Additionally, another 4 state-of-the-art methods, which directly obtain clustering results without embeddings, are compared with our proposed OT-GNN in node clustering task. These methods are all generative models. Degree Corrected SBM (DCSBM) [24] and Edge enhanced Motif-aware community detection (EdMot-SC) [25] are only based on network topology. Latent Dirichlet Allocation (LDA) [26]

Table II
COMPARISON ON NODE CLUSTERING IN TERMS OF AC AND NMI.

Metrics	Methods	Cornell	Texas	Cora	Cites.	Pub.
AC	DCSBM	37.95	48.09	38.48	26.57	53.64
	EdMot-SC	30.77	48.09	27.07	25.60	39.29
	LDA	44.62	56.28	37.19	31.34	46.30
	Block-LDA	46.15	54.10	25.52	24.35	49.01
	DeepWalk	36.05	46.72	45.61	36.21	64.84
	Node2Vec	33.85	47.54	56.30	40.76	65.56
	LINE	39.49	53.38	30.72	25.01	43.11
	GraRep	31.79	36.72	48.29	31.20	54.43
	VGAE	36.72	48.35	57.06	53.46	58.64
	ARVGE	38.21	41.48	64.08	43.50	58.76
	DGI	38.46	51.91	63.51	67.54	64.07
	OT-GCN	51.40	63.76	64.62	68.92	66.36
	OT-GAT	**52.79**	**64.67**	**66.70**	**69.54**	**67.32**
NMI	DCSBM	9.69	16.65	17.07	4.13	12.28
	EdMot-SC	9.67	18.79	9.58	11.26	0.21
	LDA	21.09	31.29	14.61	9.13	10.55
	Block-LDA	6.81	4.21	2.42	1.41	6.58
	DeepWalk	7.06	6.16	31.51	10.58	25.55
	Node2Vec	6.65	4.49	42.02	12.99	25.02
	LINE	9.27	18.16	10.13	5.62	7.17
	GraRep	8.80	12.43	35.46	9.61	17.76
	VGAE	7.77	8.52	42.92	27.93	17.83
	ARVGE	10.26	7.28	44.95	22.72	18.40
	DGI	12.52	13.98	49.76	42.74	26.64
	OT-GCN	31.66	23.17	50.89	43.74	27.62
	OT-GAT	**24.16**	**34.57**	**52.15**	**55.61**	**30.92**

Table III
COMPARISON ON NODE CLASSIFICATION IN TERMS OF AC (%).

Packages	Methods	Corn.	Texas	Cora	Cite.	Pub.
LibSVM	DeepWalk	38.97	49.18	82.57	52.52	78.79
	Node2Vec	35.90	50.27	79.98	61.63	80.30
	LINE	43.59	68.85	30.20	41.07	75.47
	GraRep	53.33	62.68	73.41	54.28	80.64
	VGAE	45.13	55.00	81.05	65.97	83.42
	ARVGE	42.56	56.28	80.42	65.10	80.64
	DGI	42.56	56.28	80.21	70.07	74.57
	OT-GCN	63.02	69.07	82.73	70.65	86.77
	OT-GAT	**65.91**	**69.98**	**83.82**	**72.62**	**87.63**
LINEAR	DeepWalk	38.46	48.09	82.04	48.42	78.36
	Node2Vec	37.95	50.27	80.79	52.44	80.08
	LINE	44.10	54.96	50.25	40.56	74.92
	GraRep	53.33	59.40	79.83	53.61	80.37
	VGAE	45.64	51.91	79.13	69.25	83.81
	ARVGE	41.54	59.02	81.24	66.71	80.59
	DGI	43.08	56.28	84.71	70.85	78.11
	OT-GCN	62.10	68.21	84.60	71.36	86.95
	OT-GAT	**63.62**	**69.71**	**85.52**	**72.55**	**88.56**

is a topic model based on node content. Block-LDA [27] is based on both topology and node attribute.

To ensure fairness, embedding dimension is uniformly set to 64 for all the methods on all the datasets. The parameters of the methods compared are set as what were used by their authors. All the results of the baseline methods are either from their original papers or produced by running the codes from the authors with their default settings.

B. Parameter Settings

In our proposed OT-GNN, Graph Convolutional Network (GCN) [5], Graph Attention Network (GAT) [17]. and Graph AutoEncoder (GAE) [9] are employed as basic graph neural networks and the resulted instances are named as OT-GCN, OT-GAT and OT-GAE, respectively. The number of clusters, i.e., F, varies from 10 to 20. Similar to DGI, one-layer GCN with dimension of output layer as 64 is adopted in OT-GCN, while one-layer GAT with the number of heads and the dimension of each head as 4 and 16, respectively, is adopted in OT-GAT.

C. Node Clustering

The results are shown in Table II. It can be observed that the two instances of OT-GNN, i.e., OT-GCN and OT-GAT outperform all other unsupervised GNNs on all the networks, especially on four webpage networks. OT-GAT achieves the best performance on all 7 networks in terms of NMI and achieves the best performance on 6 network in term of accuracy (AC). OT-GAT is slightly lower than recently proposed TLSC [28], which is a generative model combing topology and node content, in term of accuracy. This may attributes to its assumption of inconsistency between topology and node attribute, which makes its superiority on large cluster more remarkable. Factually, the superiority in terms of NMI indicates our proposed OT-GNN can obtain better performance on difficult small cluster, since NMI is a fair metric, which is more sensitive on the small cluster.

D. Node Classification

For node classification, both LibSVM and LibLINEAR are employed to classify these nodes according to the obtained embedding. On citation network, i.e., Cora, Citeseer and Pubmed, we adopt the experimental settings in [29], where 20 nodes per class, 500 nodes and 1,000 nodes are employed for training, validation and performance evaluation, respectively. For the four medium webpage networks, including Cornell and Texas, we adopt 20% labelled nodes for training, 10% labelled nodes for validation, and the other nodes for testing.

Both OT-GCN and OT-GAT, which are two instances of our proposed OT-GNN, outperform the state-of-the-art baselines on all the 7 networks. These gains are more significant on four webpages networks, i.e., Cornell, Texas, Washington and Wisconsin, since the community structures on them are more clear than network homophily property, which is the theory basis of GNNs. Besides, the performances of OT-GAT are better than those of OT-GCN, since OT-GAT takes expressive GAT [17], which jointly learns mapping function with parameter W and propagation weights with parameter b, as basic component. Note that, to the best of our knowledge, OT-GAT is the first successful unsupervised GNN, which takes GAT as basic component. The failure of adoption GAT in previous unsupervised GNN may attribute to that it is more likely to lead to overfitting from the learnable propagation weights in GAT than from the fixed propagation weights in GCN. Therefore, our proposed OT-GNN makes it possible to adopt more powerful basic GNNs without oversmoothing.

V. Conclusions

To alleviate the oversmoothing issue in unsupervised graph neural networks (GNNs), a novel Optimal Transport based unsupervised GNN (OT-GNN) is proposed. It constrains the node embeddings to keep their community/cluster structures to prevent all the node embeddings from becoming too similar to be distinguished. By imposing the obtained node embeddings to be classified into clusters of equal size, OT-GCN performs interactive node embeddings and clustering. The constrained objective function of the unsupervised GNN is relaxed to an optimal transport problem, and a fast version of the Sinkhorm-Knopp algorithm is adopted to handle large networks. Extensive experiments on node clustering and classification demonstrate the effectiveness of our proposed OT-GNN in preventing from oversmoothing.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 61972442, Grant 61802282, Grant 61802391, Grant 61772361, Grant 61971016 and Grant U1936210, in part by the Key Research and Development Project of Hebei Province of China under Grant 20350802D, in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Beijing Education Committee Cooperation Beijing Natural Science Foundation under Grant KZ201910005007, in part by the Hebei Province Innovation Capacity Enhancement Project under Grant 199676146H, and in part by the Fundamental Research Funds for Central Universities.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *TNNLS*, 2020.

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019.

[3] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018, pp. 3538–3545.

[4] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019, pp. 6861–6871.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[6] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *ICML*, 2018, pp. 5449–5458.

[7] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *ICLR*, 2019.

[8] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *ICLR*, 2020.

[9] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[10] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *IJCAI*, 2018, pp. 2609–2615.

[11] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *WSDM*, 2019, pp. 393–401.

[12] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019.

[13] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *WWW*, 2020, pp. 259–270.

[14] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Improving graph attention networks with large margin-based constraints," *NerIPS Workshop*, vol. abs/1910.11945, 2019.

[15] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.

[16] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002.

[17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[18] G. Peyré and M. Cuturi, "Computational optimal transport," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[19] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *NIPS*, 2013, pp. 2292–2300.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.

[21] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.

[22] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[23] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.

[24] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *PRE*, vol. 83, p. 016107, 2011.

[25] P. Li, L. Huang, C. Wang, and J. Lai, "Edmot: An edge enhancement approach for motif-aware community detection," in *SIGKDD*, 2019, pp. 479–487.

[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[27] R. Balasubramanyan and W. W. Cohen, "Block-lda: Jointly modeling entity-annotated text and entity-entity links," in *SDM*, 2011, pp. 450–461.

[28] D. Jin, K. Wang, G. Zhang, P. Jiao, D. He, F. Fogelman-Soulie, and X. Huang, "Detecting communities with multiplex semantics by distinguishing background, general and specialized topics," 2019, pp. 1–1.

[29] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016, pp. 40–48.