Attributed Graph Learning with 2-D Graph Convolution

Anonymous authors
Paper under double-blind review

Abstract

Graph convolutional neural networks have demonstrated promising performance in attributed graph learning, thanks to the use of graph convolution that effectively combines graph structures and node features for learning node representations. However, one intrinsic limitation of the commonly adopted 1-D graph convolution is that it only exploits graph connectivity for feature smoothing, which may lead to inferior performance on sparse and noisy real-world attributed networks. To address this problem, we propose to explore relational information among node attributes to complement node relations for representation learning. In particular, we propose to use 2-D graph convolution to jointly model the two kinds of relations and develop a computationally efficient dimensionwise separable 2-D graph convolution (DSGC). Theoretically, we show that DSGC can reduce intra-class variance of node features on both the node dimension and the attribute dimension to facilitate learning. Empirically, we demonstrate that by incorporating attribute relations, DSGC achieves significant performance gain over state-of-the-art methods on node classification and clustering on several real-world attributed networks.

1 Introduction

In an attributed graph, each node is associated with a feature vector, and nodes are connected by edges encoding their relations. Commonly seen attributed graphs include citation networks where each node is a document represented by a bag-of-words feature vector and edges are citation links, web graphs where each webpage is also represented as a vector of words and edges are hyperlinks, social networks where each user is represented by a user profile vector and edges indicate user friendship, and protein-protein interaction networks where each protein is represented by a list of protein signatures and edges encode interactions between proteins. Learning on attributed graphs including node classification and clustering finds many important applications in real-world networks.

Since the connectivity patterns and node contents of an attributed graph usually contain different information, it often requires joint modelling both aspects of information to achieve good learning performance. A major class of methods (Yang et al., 2015; Pan et al., 2016; Huang et al., 2017) is devoted to learning efficient node representations of an attributed graph via nonnegative matrix factorization or random walk statistics and then perform downstream learning tasks with the learned representations. A number of semi-supervised classification methods (Belkin et al., 2006; Weston et al., 2008; Yang et al., 2016) classify nodes in an attributed graph by training a supervised classifier on node features with some kind of graph regularizer. Recently, a series of works based on graph convolutional neural networks including ChebyNet (Defferrard et al., 2016), graph convolutional networks (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017) and graph attention networks (GAT) (Velickovic et al., 2018) have been shown to achieve state-of-the-art performance in node classification and clustering (Kipf & Welling, 2016; Wang et al., 2017) on attributed graphs.

The key component of these models is one-dimensional (1-D) graph convolution, a function that naturally combines graph structures and node contents by aggregating a node's features with its neighbours'. As shown in (Li et al., 2019), the graph convolutional operator used in GCN and many follow-up works acts as a low-pass graph filter that smooths a node's features with its neighbours'. Under the assumption that nearby nodes tend to be in the same class, it can produce similar feature representations for nodes in the same class, thereby making them easier to be classified or clustered. While this works well for attributed graphs with clear cluster structures, real-world networks could be highly noisy and sparse. For example, in a web graph such as Wikipedia, a hyperlink between two webpages does not necessarily indicate that they belong to the same category, so mixing their features could be harmful for learning. Furthermore, it has been shown that many real-world networks are scale-free (Albert & Barabási, 2002), which means there exist many low-degree nodes. Since these nodes may have very few or even no links to other nodes, it would be difficult and even impossible for them to learn similar feature representations as other same-class nodes, which is an intrinsic limitation of the 1-D graph convolution that is commonly adopted in existing models.

To address these limitations, we propose to explore relational information on a different dimension – the relations between feature attributes, in addition to node relations. Likewise, a relation between two feature attributes should reflect some kind of similarity between them. The assumption is that attributes that tend to indicate same classes should have strong relations. For example, in a citation network, node attributes are words, and documents of AI category usually contain words such as “learning”, “robotics”, “machine”, “neural”, etc. These indicative words for AI category should have much stronger relations among themselves than with other non-indicative words. These informative relations can be used to construct an attribute affinity graph to smooth node features in a similar way as the node relations do, only in a different dimension. Importantly, attribute relations can complement node relations in node representation learning. For instance, consider a document that has no links to others and hence it is impossible to do feature smoothing with node relations. But with attribute relations, it can still learn similar feature representations as other same-class nodes.

In this paper, we make the following contributions.

- **Methodology**: We propose to use 2-D graph convolution to jointly model node relations and attribute relations for learning node representations of attributed graphs. Further, we develop a computationally efficient dimensionwise separable 2-D graph convolution (DSGC), which is equivalent to performing 1-D graph convolution alternately on the node dimension and the attribute dimension respectively.
- **Theoretical Insight**: We show that regular 1-D graph convolution on the node dimension can reduce intra-class variance of node features, which explains the success of many existing methods. Further, we show that with a properly constructed attribute affinity graph, graph convolution on the attribute dimension can also reduce intra-class variance of node features. Jointly, our analysis provides a theoretical justification of DSGC.
- **Empirical Study**: We implement DSGC for node classification and clustering on attributed graphs, and compare it with state-of-the-art methods on a citation network, a web graph, and an email network. The results demonstrate the superiority of DSGC over regular 1-D graph convolution on spare and noisy real-world attributed networks. We also show that DSGC can be plugged into existing models to substantially improve their performance.

2 Related Works

Structural Graph Learning. Methods for structural graph learning only utilize graph structures (node relations). A common approach is to learn smooth low-dimensional embeddings of nodes using Markov random walks (Szummer & Jaakkola, 2002; Perozzi et al., 2014; Grover & Leskovec, 2016), Laplacian eigenmaps (Belkin & Niyogi, 2004), spectral kernels (Chapelle et al., 2003; Zhang & Ando, 2006), autoencoders (Wang et al., 2016; Cao et al., 2016; Ye et al., 2018) and generative adversarial nets (Dai et al., 2018; Wang et al., 2018). Another direction is graph-based semi-supervised classification, which includes methods based on low-density graph partition (Blum & Chawla, 2001; Blum et al., 2004; Joachims, 2003) and the popular label propagation method and its variants (Zhu et al., 2003; Zhou et al., 2004; Bengio et al., 2006; Hein & Maier, 2007; Wu et al., 2012).

Attributed Graph Learning. Methods for attributed graph learning take into account both graph structures and node features. A major class of methods learns node representations (Yang et al., 2015; Pan et al., 2016; Huang et al., 2017) or clustering nodes (Xia et al., 2014; Zhou et al., 2010; Li et al., 2018) by applying nonnegative matrix factorization, random walk statistics, or Laplacian eigenmaps on both graph structures and node features. Some node clustering methods use a Bayesian model (Xu et al., 2012; Bojchevski & Günnemann, 2018) or design a distance measure that trades off structural and feature information (Zhou et al., 2009; Cheng et al., 2011). Statistical relational learning methods model node relations and features with probabilistic graphical models, e.g., relational Markov networks (Taskar et al., 2002). Another category of related work is the graph-based semi-supervised node classification methods that exploit both graph structures and node features. For example, iterative classification algorithm (Sen et al., 2008) iteratively classifies an unlabeled node using its neighbours' labels and features. Manifold regularization (Belkin et al., 2006), deep semi-supervised embedding (Weston et al., 2008), and Planetoid (Yang et al., 2016) classify node features by training a supervised classifier with a Laplacian or embedding-based regularizer.

Graph Neural Networks. Another line of research on attributed graph learning is based on graph neural networks (Scarselli et al., 2009; Li et al., 2016). Inspired by the success of convolutional neural networks (CNN) on Euclidean data, recent works (Bruna et al., 2014; Henaff et al., 2015; Duvenaud et al., 2015; Atwood & Towsley, 2016) proposed to use 1-D graph convolution for attributed graph learning. To avoid the expensive eigen-decomposition, ChebyNet (Defferrard et al., 2016) uses a polynomial filter represented by k-th order polynomials of graph Laplacian via Chebyshev expansion. Graph convolutional networks (GCN) (Kipf & Welling, 2017) further simplifies ChebyNet by designing an efficient layer-wise propagation rule via a first-order approximation of spectral graph convolution. GCN achieved outstanding results in semi-supervised classification and inspired many follow-up works including MoNet (Monti et al., 2017), GraphSAGE (Hamilton et al., 2017), graph attention networks (Velickovic et al., 2018), gated attention networks (Zhang et al., 2018a), FastGCN (Chen et al., 2018b), dual graph convolutional neural network (Zhuang & Ma, 2018), stochastic GCN (Chen et al., 2018a), attributed network representation learning (Zhang et al., 2018c), LanczosNet (Liao et al., 2019), deep graph infomax (Velickovic et al., 2019), graph Markov neural networks (Qu et al., 2019), DisenGCN (Ma et al., 2019a), MixHop (Abu-El-Haija et al., 2019), etc. A recent interesting research direction is on learning graph structures for graph neural networks, e.g., Bayesian GCN (Zhang et al., 2018b) and LDS (Franceschi et al., 2019). Remarkably, LDS can automatically learn a graph for data samples even when it is not available.

Some attributed graph clustering methods based on 1-D graph convolution also showed promising performance, including graph autoencoder and graph variational autoencoder (Kipf & Welling, 2016), marginalized graph autoencoder (Wang et al., 2017), adversarially regularized graph autoencoder and adversarially regularized variational graph autoencoder (Pan et al., 2018), and adaptive graph convolution (Zhang et al., 2019). More comprehensive reviews of graph neural networks can be found in (Cai et al., 2018; Zhang et al., 2018d; Zhou et al., 2018).

3 2-D Graph Convolution

In this section, we present 2-D graph convolution for attributed graph learning. A comprehensive introduction of multi-dimensional graph Fourier transform can be found in (Kurokawa et al., 2017). Different from Kurokawa et al. (2017), here we propose a localized 2-D graph convolution to circumvent the computationally intensive graph Fourier transform. Furthermore, we propose an even simpler dimensionwise separable 2-D graph convolution to efficiently model both node relations and attribute relations along the two dimensions of the feature matrix of an attributed graph.

3.1 2-D Graph Fourier Transform and Spectral Graph Convolution

A 2-D *graph signal* is a function defined on the Cartesian product of the vertex sets of two graphs. Formally, given two weighted undirected graph $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$, denote the vertex sets by $\mathcal{V}^{(1)}$ and $\mathcal{V}^{(2)}$, the edge set by $\mathcal{E}^{(1)}$ and $\mathcal{E}^{(2)}$, and the weighted adjacency matrix by $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$. A 2-D graph signal x on $(\mathcal{G}^{(1)}, \mathcal{G}^{(2)})$ is a real-valued function $f : \mathcal{V}^{(1)} \times \mathcal{V}^{(2)} \rightarrow \mathbb{R}$, which can be conveniently represented in matrix form:

$$\boldsymbol{X} = (x_{ij}) \in \mathbb{R}^{n\times m}, \quad x_{ij} = f(\nu_i^{(1)}, \nu_j^{(2)}), \tag{1}$$

where $n = |\mathcal{V}^{(1)}|$ and $m = |\mathcal{V}^{(2)}|$. In this paper, $\mathcal{G}^{(1)}$ represents the given node graph; $\mathcal{G}^{(2)}$ represents the constructed attribute affinity graph; and $\boldsymbol{X}$ is the node feature matrix, which is a 2-D signal on the two graphs. We call the adjacency matrices $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$ node affinity matrix and attribute affinity matrix respectively.

Define the graph Laplacian of $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ as $\boldsymbol{L}_l = \boldsymbol{D}^{(1)} - \boldsymbol{A}^{(1)}$ and $\boldsymbol{L}_r = \boldsymbol{D}^{(2)} - \boldsymbol{A}^{(2)}$ respectively. Denote by λ_i and μ_j the eigenvalues of $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$, and $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m]$

the corresponding eigenbasis respectively, then the $n \times m$ outer products $\boldsymbol{u}_i\boldsymbol{v}_j^\top$ form a basis for the linear space $\mathbb{R}^{n\times m}$. It is known as 2-D graph Fourier basis – an analogy of the Fourier basis in classical harmonic analysis in graph domain. The corresponding eigenvalue pair (λ_i, μ_j) is known as the frequency of basis matrix $\boldsymbol{u}_i\boldsymbol{v}_j^\top$. Then, a 2-D graph signal $\boldsymbol{X}$ can be decomposed as:

$$\boldsymbol{X} = \sum_{ij} s_{ij}(\boldsymbol{u}_i\boldsymbol{v}_j^\top) = \boldsymbol{USV}^\top,\ \boldsymbol{S} = (s_{ij}) \in \mathbb{R}^{n\times m}. \tag{2}$$

Then, we can define 2-D *graph Fourier transform* as $\boldsymbol{S} = \boldsymbol{U}^\top\boldsymbol{XV}$, where $\boldsymbol{S}$ is called the spectrum of signal $\boldsymbol{X}$ or Fourier coefficients.

Based on 2-D graph Fourier transform, we can now manipulate 2-D graph signals in the spectral (frequency) domain and define 2-D spectral graph convolution. By the convolution theorem, the convolution of two signals equals to point-wise multiplication of their spectrum in the spectral domain. 2-D *spectral graph convolution* is a function conv : $\mathcal{R}^{n\times m} \to \mathcal{R}^{n\times m}$ that takes signal $\boldsymbol{X}$ as input and outputs a new signal $\boldsymbol{Z}$:

$$\boldsymbol{Z} = \sum_{ij} p(\lambda_i, \mu_j)s_{ij}(\boldsymbol{u}_i\boldsymbol{v}_j^\top) = \boldsymbol{U}(\boldsymbol{S} \circ \boldsymbol{P})\boldsymbol{V}^\top, \tag{3}$$

where $p(\lambda, \mu) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the frequency response of the convolution; $\circ$ is Hadamard (element-wise) product; and $\boldsymbol{P} \in \mathbb{R}^{n\times m}$ with $P_{ij} = p(\lambda_i, \mu_j)$ is the frequency response in matrix form.

3.2 Fast Localized 2-D Spatial Graph Convolution

Although Eq. (3) well defines 2-D graph convolution, it is often impractical to perform convolution in the spectral domain, due to the high cost of computing the eigenbasis $\boldsymbol{U}, \boldsymbol{V}$ needed for Fourier transform. Similar to (Defferrard et al., 2016) on 1-D graph convolution, here we propose 2-D spatial graph convolution to avoid intensive computation. Without loss of generality, we restrict the frequency response $p(\cdot,\cdot)$ to be a polynomial of two variables λ, μ with parameters $\boldsymbol{\Theta} \in \mathbb{R}^{n\times m}$, i.e., $p(\lambda, \mu) = \sum_{i,j} \theta_{ij}\lambda^i\mu^j$. Then, the 2-D spectral graph convolution in Eq. (3) becomes

$$\boldsymbol{Z} = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} \theta_{ij}\boldsymbol{L}_l^i\boldsymbol{X}\boldsymbol{L}_r^j. \tag{4}$$

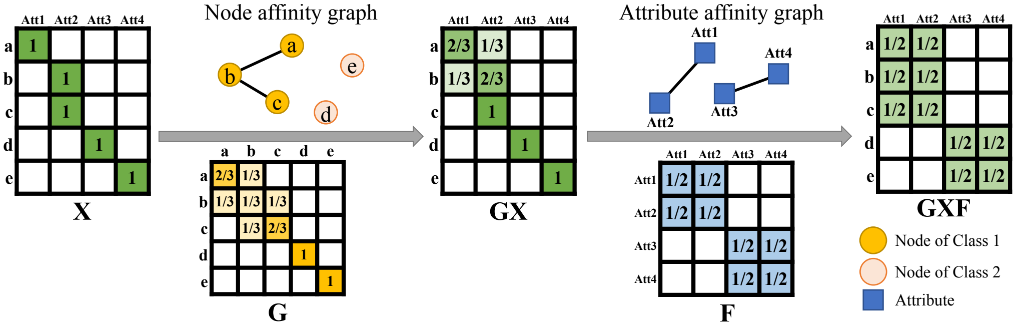
Eq. (4) is called 2-D *spatial graph convolution*, as it manipulates the signal $\boldsymbol{X}$ in the spatial domain. Parameter $\boldsymbol{\Theta}$ is called the kernel of the convolution. Here, the spatial convolutional filter is localized. Denote by k_1 and k_2 the largest exponent of λ and μ in the polynomial p respectively, then $i > k_1$ and $j > k_2$ imply $\theta_{ij} = 0$. The convoluted signal z_{ij} of vertex pair $(\nu_i^{(1)}, \nu_j^{(2)})$ only depends on the neighbourhood of $\nu_i^{(1)}$ within k_1 hops and the neighbourhood of $\nu_j^{(2)}$ within k_2 hops, so the filter is said to be k_1-localized on $\mathcal{G}^{(1)}$ and k_2-localized on $\mathcal{G}^{(2)}$, and the size of the kernel $\boldsymbol{\Theta}$ is $k_1 \times k_2$.

Dimensionwise Separable 2-D Graph Convolution (DSGC) Although the above spatial graph convolution avoids the computationally expensive Fourier transform, its general form with kernel size $k_1 \times k_2$ still involves at least $k_1 \times k_2$ matrix multiplications. Inspired by the depthwise separable convolution proposed in (Howard et al., 2017), we streamline spatial graph convolution by restricting the rank of $\boldsymbol{\Theta}$ to be one. Consequently, $\boldsymbol{\Theta}$ is decomposed as an outer product of two vectors $\boldsymbol{\theta}^{(1)} \in \mathbb{R}^n$ and $\boldsymbol{\theta}^{(2)} \in \mathbb{R}^m$. The frequency response p can be decomposed as a product of two single variable polynomials, i.e., $p(\lambda, \mu) = p_1(\lambda)p_2(\mu)$, where $p_1(\lambda) = \sum_i \theta_i^{(1)}\lambda^i$ and $p_2(\mu) = \sum_j \theta_j^{(2)}\mu^j$. Finally, the 2-D spatial graph convolution in Eq. (4) becomes

$$\boldsymbol{Z} = \boldsymbol{GXF}, \quad \text{where } \boldsymbol{G} = p_1(\boldsymbol{L}_l) \text{ and } \boldsymbol{F} = p_2(\boldsymbol{L}_r). \tag{5}$$

We call Eq. (5) dimensionwise separable graph convolution (DSGC). The fastest way to compute it only requires $k_1 + k_2$ matrix multiplications, much less than the $k_1 \times k_2$ matrix multiplications needed by a general 2-D spatial graph convolution.

We call $\boldsymbol{GX}$ node graph convolution and $\boldsymbol{G}$ the node graph convolutional filter. Similarly, we call $\boldsymbol{XF}$ attribute graph convolution and $\boldsymbol{F}$ the attribute graph convolutional filter. Notably, $\boldsymbol{G}$ and $\boldsymbol{F}$ can complement each other to learn better node representations. As illustrated in Fig. 1, the node affinity



X

	Att1	Att2	Att3	Att4
a	1			
b		1		
c		1		
d			1	
e				1

G

	a	b	c	d	e
a	2/3	1/3			
b	1/3	1/3	1/3		
c		1/3	2/3		
d				1	
e					1

GX

	Att1	Att2	Att3	Att4
a	2/3	1/3		
b	1/3	2/3		
c		1		
d			1	
e				1

F

	Att1	Att2	Att3	Att4
Att1	1/2	1/2		
Att2	1/2	1/2		
Att3			1/2	1/2
Att4			1/2	1/2

GXF

	Att1	Att2	Att3	Att4
a	1/2	1/2		
b	1/2	1/2		
c	1/2	1/2		
d			1/2	1/2
e			1/2	1/2

Figure 1: Illustration of DSGC.

graph only captures node relations of Class 1, and applying node graph convolution on the feature matrix $\boldsymbol{X}$ smooths the node features of Class 1, but it does not affect those of Class 2. Fortunately, the attribute affinity graph encodes complementary relational information on another dimension, and further applying attribute graph convolution makes the node features of Class 2 similar and those of Class 1 even more similar, thereby making them much easier to be classified or clustered.

4 Intra-class Variance Reduction by DSGC

Given a data distribution, the lowest possible error rate an classifier can achieve is the Bayes error rate (Fukunaga, 2013), which is caused by the intrinsic overlap between different classes and cannot be avoided. In this section, we show that DSGC with proper graph convolutional filters can reduce intra-class variance of the data distribution while keeping class centers roughly unchanged, hence reducing the overlap between classes and improving learning performance.

Intra-class Variance and Inter-class Variance. Suppose samples $\boldsymbol{x}_i$ and their labels y_i are observations of a random vector $\mathbb{X} = [\mathbb{X}_1, \cdots, \mathbb{X}_m]^\top$ and a random variable $\mathbb{Y}$ respectively. We define the variance of random vector $\mathbb{X}$ to be the sum of the variance of each dimension $\mathbb{X}_j$, i.e., the trace of the covariance matrix of $\mathbb{X}$. According to law of total variance (Grinstead & Snell, 2012), the variance of $\mathbb{X}$ can be divided into intra-class variance and inter-class variance:

$$\operatorname{Var}(\mathbb{X}) = \underbrace{\operatorname{E}[\operatorname{Var}(\mathbb{X}|\mathbb{Y})]}_{\text{Intra-class Variance}} + \underbrace{\operatorname{Var}(\operatorname{E}[\mathbb{X}|\mathbb{Y}])}_{\text{Inter-class Variance}}, \tag{6}$$

where the conditional variance $\operatorname{Var}(\mathbb{X}|\mathbb{Y}=k)$ is the variance of class k and the conditional expectation $\operatorname{E}[\mathbb{X}|\mathbb{Y}=k]$ is the k-th class center. Intra-class variance (IntraVar) measures the average divergence within each class, while inter-class variance (InterVar) measures the divergence among class centers. We are interested in the IntraVar/InterVar ratio. Here, we assume that each class of data $\Pr(\mathbb{X}|\mathbb{Y}=k)$ follows a unimodal distribution (e.g. Gaussian, chi-square, Laplace), a class of most common distributions in the real world, and with roughly convex contours. Under this assumption, a low IntraVar/InterVar ratio generally indicates low classification error, since the class overlap is reduced. This is also verified by our experiments on real attributed networks in section 7.

4.1 Intra-class Variance Reduction by Node Graph Convolution

As variance of sample mean is always less than variance of individual samples, averaging samples of the same class can always reduce intra-class variance. Actually this is how node graph convolution works. For any node i, node graph convolution $\boldsymbol{GX}$ produces a new feature vector $\boldsymbol{z}_i = \sum_j G_{ij}\boldsymbol{x}_j$. When $\boldsymbol{G}$ is a stochastic matrix, the output feature vector $\boldsymbol{z}_i$ is a weighted average of the neighbours of $\boldsymbol{x}_i$. Denote by $\mathbb{Z}$ a random vector of $\boldsymbol{z}_i$. Intuitively, as long as each node i has enough same-class neighbours, $\mathbb{Z}$ will have a smaller IntraVar/InterVar ratio than $\mathbb{X}$. Formally, assume that the graph is generated by a stochastic block model, where nodes from the same class are connected with probability p and those from different classes are connected with probability q. Then, with the stochastic graph filter $\boldsymbol{G} = \boldsymbol{D}^{-1}\boldsymbol{A}^{(1)}$, we have the following theorem.

Theorem 1. *When q is sufficiently small, the* IntraVar/InterVar *ratio of $\mathbb{Z}$ is less than or equal to that of $\mathbb{X}$, i.e.,*

$$\frac{\mathrm{E}\left[\mathrm{Var}\left(\mathbb{Z}|\mathbb{Y}\right)\right]}{\mathrm{Var}\left(\mathrm{E}\left[\mathbb{Z}|\mathbb{Y}\right]\right)} \le \frac{\mathrm{E}\left[\mathrm{Var}\left(\mathbb{X}|\mathbb{Y}\right)\right]}{\mathrm{Var}\left(\mathrm{E}\left[\mathbb{X}|\mathbb{Y}\right]\right)}. \tag{7}$$

The proof is given in the Appendices. Under the assumption that nodes in the same class are most likely to be connected, node graph convolution $\boldsymbol{G}$ can reduce intra-class variance while keeping inter-class variance roughly unchanged, thereby decreasing the IntraVar/InterVar ratio.

4.2 Intra-class Variance Reduction by Attribute Graph Convolution

In the following, we show that a proper attribute graph convolutional filter $\boldsymbol{F}$ can also reduce the IntraVar/InterVar ratio. We use the convention that the random vector $\mathbb{X}$ is a column vector, and hence the attribute graph convolution $\boldsymbol{XF}$ results in a new random vector $\boldsymbol{F}^\top\mathbb{X}$. We also assume that the node features are mean-centered, i.e. $\mathrm{E}\left[\mathbb{X}\right] = \boldsymbol{0}$. Denote by $\mathrm{Cov}^{(k)} = \mathrm{Cov}\left(\mathbb{X}|\mathbb{Y}=k\right)$ the covariance matrix of $\mathbb{X}$ w.r.t. each class, and denote by $\pi_k = \Pr(\mathbb{Y}=k)$ the portion of each class. Then, after attribute graph convolution, the intra-class variance of data becomes

$$\mathrm{E}\left[\mathrm{Var}\left(\boldsymbol{F}^\top\mathbb{X}|\mathbb{Y}\right)\right] = \sum_k \pi_k \sum_{ij} \mathrm{Cov}_{ij}^{(k)} (\boldsymbol{F}\boldsymbol{F}^\top)_{ij} \tag{8}$$

Theorem 2. *If the attribute graph convolutional filter $\boldsymbol{F}$ is a doubly stochastic matrix, then the output of attribute graph convolution has an intra-class variance less than or equal to that of $\mathbb{X}$, i.e.,*

$$\sum_i F_{ij} = \sum_j F_{ij} = 1 \textit{ and } F_{ij} \ge 0, \forall\, i,j \quad \Rightarrow \quad \mathrm{E}\left[\mathrm{Var}\left(\boldsymbol{F}^\top\mathbb{X}|\mathbb{Y}\right)\right] \le \mathrm{E}\left[\mathrm{Var}\left(\mathbb{X}|\mathbb{Y}\right)\right].$$

The proofs of Eq. (8) and Theorem 2 are given in the Appendices. Eq. (8) suggests that to reduce intra-class variance, the attribute affinity graph should connect attributes $\mathbb{X}_i$ and $\mathbb{X}_j$ with small or even negative covariance $\mathrm{Cov}_{ij}^{(k)}$. For example, in Figure 1, the attributes Att3 and Att4 are both indicative of class 2, but negatively correlated w.r.t. class 2, so connecting them in the attribute affinity graph can greatly reduce the intra-class variance of class 2.

To achieve a low IntraVar/InterVar ratio, in addition to reducing intra-class variance, we also need to keep the class centers apart after convolution, which then depends on the quality of the attribute affinity graph. A good attribute affinity graph should connect attributes that share similar expectations conditioned on $\mathbb{Y}$. Formally, each attribute $\mathbb{X}_j$ has K conditional expectations w.r.t. $\mathbb{Y}$, which are denoted as a vector $\boldsymbol{e}_j = \left(\mathrm{E}\left[\mathbb{X}_j|\mathbb{Y}=1\right], \cdots, \mathrm{E}\left[\mathbb{X}_j|\mathbb{Y}=K\right]\right) \in \mathbb{R}^K$. We have the following.

Theorem 3. *If $\forall F_{ij} \neq 0$, $\left\|\boldsymbol{e}_i - \boldsymbol{e}_j\right\|_2 \le \varepsilon$, then the distance between $\boldsymbol{e}_j$ and $\widehat{\boldsymbol{e}}_j = \sum_i F_{ij}\boldsymbol{e}_i$ is also less than or equal to ε, i.e.,*

$$\left\|\boldsymbol{e}_i - \boldsymbol{e}_j\right\|_2 \le \varepsilon,\ \forall F_{ij} \neq 0 \quad \Rightarrow \quad \left\|\boldsymbol{e}_j - \widehat{\boldsymbol{e}}_j\right\|_2 \le \varepsilon,$$

and ε can be arbitrarily small with a proper $\boldsymbol{F}$.

By Theorem 3, since the conditional expectations of each attribute may change little after attribute graph convolution, one can infer that the class centers will also change little, and so does the inter-class variance. Combining Theorems 2 & 3, it suggests that a proper attribute affinity graph should connect attributes that have similar class means but are less positively correlated, so as to achieve a low IntraVar/InterVar ratio and improve performance. Again, it can be seen in Figure 1 that the attributes Att3 and Att4 have exactly the same class means, but are negatively correlated w.r.t. class 2, and connecting them in the attribute affinity graph reduces the intra-class variance of class 2 to 0.

5 Attributed Graph Learning with DSGC

5.1 Unsupervised Node Representation Learning

Given an attributed graph with node feature matrix $\boldsymbol{X}$, we can learn the node representations $\boldsymbol{Z}$ in an unsupervised manner by applying DSGC on $\boldsymbol{X}$, i.e.,

$$\boldsymbol{Z} = \boldsymbol{GXF}, \tag{9}$$

and then perform various downstream learning tasks with $\boldsymbol{Z}$.

Node Classification. With the learned node representations $\boldsymbol{Z}$, we can simply train a classifier such as multilayer perceptron with the labeled nodes, and then apply the trained classifier on the unlabeled nodes to predict their labels.

Node Clustering. With the learned node representations $\boldsymbol{Z}$, we propose to cluster nodes as follows. We first apply a linear kernel on $\boldsymbol{Z}$ to learn pairwise proximity between nodes, i.e., $\boldsymbol{K} = \boldsymbol{Z}\boldsymbol{Z}^\top$, and then perform spectral clustering (Perona & Freeman, 1998; Von Luxburg, 2007) on $\boldsymbol{K}$.

5.2 End-to-End Semi-Supervised Learning

For semi-supervised learning on attributed graphs, we can also use DSGC to replace the 1-D graph convolution used in existing end-to-end semi-supervised learning models including GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018) and GraphSAGE (Hamilton et al., 2017). For example, to incorporate DSGC into the vanilla GCN, we can modify the first layer propagation of GCN as:

$$\boldsymbol{H}^{(1)} = \sigma(\boldsymbol{G}\boldsymbol{X}\boldsymbol{F}\boldsymbol{W}^{(1)}), \tag{10}$$

where $\boldsymbol{H}^{(1)}$ is the hidden units in the first layer, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{m\times l}$ is the trainable parameters of GCN, and σ is a nonlinear function such as ReLU. Both $\boldsymbol{G}$ and $\boldsymbol{F}$ are fixed filters, where $\boldsymbol{G}$ is the node graph convolutional filter of GCN, and $\boldsymbol{F}$ is the proposed attribute graph convolutional filter.

Importantly, Eq. (10) can be considered as feeding a filtered feature matrix $\boldsymbol{X}\boldsymbol{F}$ instead of the raw feature matrix $\boldsymbol{X}$ to GCN. By our above analysis, a proper attribute graph filter $\boldsymbol{F}$ can reduce intra-class variance, which makes $\boldsymbol{X}\boldsymbol{F}$ much easier to classify and guarantees to help train a better model. Further, it can be shown that the model trained by Eq. (10) is essentially different from that of GCN almost surely. For GCN, the model is freely chosen from the parameter space $\boldsymbol{W}^{(1)}$, while the model trained by Eq. (10) is restricted in a subspace $\boldsymbol{F}\boldsymbol{W}^{(1)}$. Since $\boldsymbol{F}$ is low-pass (section 6.2), it is also low-rank, and $\boldsymbol{F}\boldsymbol{W}^{(1)}$ is a small subspace of $\mathbb{R}^{m\times l}$ projected by $\boldsymbol{F}$. Model parameters in this subspace are generally better in terms of the generalization performance (test accuracy), due to the variance reduction property of $\boldsymbol{F}$. However, the model learned by Eq. (10) can hardly be learned by GCN, since the subspace $\boldsymbol{F}\boldsymbol{W}^{(1)}$ has measure zero, which is an extremely small subset of $\mathbb{R}^{m\times l}$.

6 Implementation of DSGC

6.1 Node Graph Convolutional Filters

The node affinity graph ($\boldsymbol{A}^{(1)}$) is given as part of the dataset. There are various graph convolutional filters available (Li et al., 2019), e.g., the one used in GCN (Kipf & Welling, 2017) is a symmetrically normalized node affinity matrix. In our experiments, we use a row normalized node affinity matrix (consistent to our analysis in section 4.1) of order 2 (following GCN) as the filter for node graph convolution:

$$\boldsymbol{G} = (\boldsymbol{D}_1^{-1}\boldsymbol{A}^{(1)})^2, \tag{11}$$

where $\boldsymbol{D}_1$ is the degree matrix of $\boldsymbol{A}^{(1)}$. We observe in our experiments that the performance of the symmetrically normalized node affinity matrix (GCN filter) is similar.

6.2 Attribute Graph Convolutional Filters

A key issue in implementing DSGC is to construct the attribute affinity graph ($\boldsymbol{A}^{(2)}$). Possible ways of constructing $\boldsymbol{A}^{(2)}$ include extracting entity relation information from existing knowledge bases, building a similarity graph from features, or identifying correlations by domain knowledge. In the following, we describe two methods for text data, which have been proven useful by our experiments.

Positive point-wise mutual information (PPMI). Positive PMI (Church & Hanks, 1990) is a common tool for measuring the association between two words in natural language processing. Given a pair of words w_i and w_j, the edge weight is defined as the PPMI between w_i and w_j:

$$a_{ij}^{(2)} = \text{PPMI}(w_i, w_j) = \left[\log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}\right]_+, \tag{12}$$

where $p(w_i, w_j)$ and $p(w_i)$ are learned by sliding a window over a large corpus of text. PMI reflects word collocation, as it assumes that if two words co-occur more than expected under independence, there must be some kind of semantic relation between them, which is often true in practice. With the constructed attribute affinity graph $\boldsymbol{A}^{(2)}$ by PPMI, in our experiments, we use a symmetrically normalized affinity matrix as the filter:

$$\boldsymbol{F} = \boldsymbol{D}_2^{-\frac{1}{2}} \boldsymbol{A}^{(2)} \boldsymbol{D}_2^{-\frac{1}{2}}, \tag{13}$$

where $\boldsymbol{D}_2$ is the degree matrix of $\boldsymbol{A}^{(2)}$.

Word embedding based k-NN graphs. Word embedding is a collection of techniques that map vocabularies to vectors in a Euclidean space. Embeddings of words are pre-trained vectors learned from corpus with algorithms such as GloVe (Pennington et al., 2014). Since word embeddings capture semantic relations between words (Bakarov, 2018), they can be used for constructing an attribute affinity graph. With the embedding vectors, we can construct a k-NN graph with some proximity metric such as the Euclidean distance. With the constructed attribute affinity graph $\boldsymbol{A}^{(2)}$, in our experiments, we use the following one-step lazy random walk filter (Li et al., 2019):

$$\boldsymbol{F} = (\boldsymbol{I} + \boldsymbol{D}_2^{-\frac{1}{2}} \boldsymbol{A}^{(2)} \boldsymbol{D}_2^{-\frac{1}{2}})/2, \tag{14}$$

where $\boldsymbol{D}_2$ is the degree matrix of $\boldsymbol{A}^{(2)}$. Note that the filters in Eq. (13) and (14) are slightly different, since the diagonal weight of the affinity matrix constructed by word embedding is much smaller than that of the PPMI affinity matrix, and we want to assign sufficient weight to each attribute itself. Also, we use filters of order 1 since the PPMI affinity matrix is very dense. According to the analysis in Li et al. (2019), Eq. (13) and (14) are both low-pass filters.

7 Empirical Study

To validate the effectiveness of DSGC, we conduct extensive experiments for semi-supervised node classification and node clustering on three large real-world attributed networks including 20 Newsgroups (20 NG) (Lang, 1995), Wikispeedia (Wiki) (West et al., 2009; West & Leskovec, 2012) and Large Cora (L-Cora) (McCallumzy et al., 1999; Li et al., 2019). [1] Due to space limitation, details of datasets, experimental setup and computational time are provided in the Appendices.

Variance Reduction. First of all, to verify our analysis in section 4, we demonstrate the variance reduction effect of both node graph convolution and attribute graph convolution. As shown in Figure 2, 1-D graph convolution with either $\boldsymbol{G}$ or $\boldsymbol{F}$ already greatly reduces the IntraVar/InterVar ratio, and together they further significantly reduce the ratio.

7.1 Node Classification

Baselines. We test the proposed node classification method by DSGC with or without using the node graph convolutional filter ($\boldsymbol{G}$) and the attribute graph convolutional filter ($\boldsymbol{F}$) in five cases. We use a two-layer multilayer perceptron as the classifier for DSGC. PPMI and Emb denote constructing the attribute affinity graph with positive point-wise mutual information and word embedding respectively. We compare DSGC with the following baselines: label propagation (LP) (Wu et al., 2012), multi-layer perceptron (MLP), GCN (Kipf & Welling, 2017), generalized label propagation (GLP) (Li et al., 2019), GraphSAGE (Hamilton et al., 2017), graph attention networks (GAT) (Velickovic et al., 2018), and deep graph infomax (DGI) (Velickovic et al., 2019). We also try to incorporate DSGC into GCN, GAT, GraphSAGE, and LDS (Franceschi et al., 2019) (see the experimental setup of LDS in Appendix A.4) as described in section 5.2 to improve their performance.

Performance. We test semi-supervised node classification under two scenarios – 20 labels per class and 5 labels per class. For each task, we report the mean classification accuracies and standard deviations over 10 runs, as summarized in Table 1, where the top 2 accuracies are highlighted in bold. In Table 2, we report the results of GAT, GCN, GraphSAGE, and LDS after incorporating DSGC. The following observations can be made.

[1] Note that we did not use the "Cora", "Citeseer" and "PubMed" datasets as in (Kipf & Welling, 2017; Yang et al., 2016; Sen et al., 2008), since the attribute (word) lists are not provided in these datasets.

Table 1: Classification accuracy.

Datasets			20 NG		L-Cora		Wiki	
Methods	$\boldsymbol{G}$	$\boldsymbol{F}$	20 labels	5 labels	20 labels	5 labels	20 labels	5 labels
LP	✓	✗	16.39 ± 0.20	8.62 ± 0.20	55.77 ± 0.97	38.97 ± 3.15	9.53 ± 0.05	10.54 ± 0.19
MLP	✗	✗	65.77 ± 0.22	36.10 ± 1.11	51.05 ± 0.71	33.56 ± 2.43	60.86 ± 0.69	29.95 ± 1.04
GLP	✓	✗	76.21 ± 0.18	47.86 ± 1.63	67.58 ± 1.06	51.04 ± 1.61	33.42 ± 1.44	15.38 ± 1.37
GCN	✓	✗	76.14 ± 0.24	47.70 ± 1.64	66.69 ± 0.98	48.62 ± 1.81	48.65 ± 0.65	38.30 ± 1.48
GAT	✓	✗	75.16 ± 0.25	49.14 ± 1.55	66.49 ± 1.01	49.27 ± 2.25	48.88 ± 0.68	36.90 ± 1.75
DGI	✓	✗	73.34 ± 0.27	**66.57** ± 0.63	61.39 ± 0.50	**54.77** ± 1.24	49.70 ± 1.63	43.64 ± 1.89
GraphSAGE	✓	✗	65.73 ± 0.17	42.48 ± 0.77	57.28 ± 0.71	46.79 ± 1.91	65.52 ± 0.62	48.81 ± 0.76
DSGC ($\boldsymbol{GX}$)	✓	✗	76.27 ± 0.20	47.92 ± 1.57	67.16 ± 1.04	51.93 ± 1.46	49.78 ± 0.50	43.79 ± 1.48
DSGC ($\boldsymbol{XF}$)	✗	Emb	68.08 ± 0.22	43.41 ± 0.99	55.34 ± 0.66	38.13 ± 2.20	**68.43** ± 0.37	**54.25** ± 0.99
	✗	PPMI	76.10 ± 0.21	61.45 ± 0.74	58.54 ± 0.79	44.58 ± 2.00	**69.53** ± 0.36	**58.44** ± 1.48
DSGC ($\boldsymbol{GXF}$)	✓	Emb	**77.38** ± 0.14	55.24 ± 1.21	**68.38** ± 0.85	52.87 ± 1.56	58.66 ± 0.65	45.71 ± 1.64
	✓	PPMI	**81.91** ± 0.20	**70.35** ± 0.72	**67.60** ± 0.82	**54.07** ± 1.13	58.35 ± 0.52	46.72 ± 1.66

[⋆] ✓ and ✗ indicate using/not using $\boldsymbol{G}$ or $\boldsymbol{F}$.

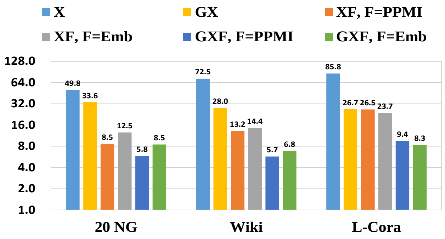


Figure 2: IntraVar/InterVar ratios.

Table 2: Improve baselines with DSGC.

Methods	$\boldsymbol{F}$	20 NG	L-Cora	Wiki
GAT	✗	75.16 ± 0.25	66.49 ± 1.01	48.88 ± 0.68
	PPMI	80.52 ± 0.16	67.00 ± 0.65	55.51 ± 0.38
GCN	✗	76.14 ± 0.24	66.69 ± 0.98	48.65 ± 0.65
	PPMI	81.80 ± 0.22	65.83 ± 0.87	60.22 ± 0.71
LDS	✗	75.74 ± 0.68	62.51 ± 0.70	62.89 ± 0.61
	PPMI	81.98 ± 0.46	62.94 ± 0.32	66.97 ± 0.63
GraphSAGE	✗	65.73 ± 0.17	57.28 ± 0.71	65.52 ± 0.62
	PPMI	76.27 ± 0.33	60.23 ± 1.81	67.26 ± 0.52

[⋆] ✗ indicates not using $\boldsymbol{F}$.

1) Node graph convolution does not always help. On 20 NG and L-Cora, methods based on node graph convolution such as DSGC ($\boldsymbol{GX}$), GLP, GCN and GAT all outperform MLP significantly. However, on Wiki, node graph convolution harms the performance, which is due to the highly noisy hyperlink graph with intra-class edge ratio 38.0%, much lower than that of 20 NG (96.8%) and L-Cora (76.5%) (see Table 5 in the Appendices). This shows the limitation of node graph convolution.

2) Attribute graph convolution works. On all the three datasets, DSGC ($\boldsymbol{XF}$) outperforms MLP by a large margin. This shows that attribute graph convolution can learn useful node representations even when there are no connections between nodes. Remarkably, on Wiki, where the hyperlink graph is of bad quality, DSGC ($\boldsymbol{XF}$) with either PPMI or Emb outperforms all the baselines by a large margin. DSGC ($\boldsymbol{XF}$) with PPMI improves upon the best baseline by 4% and 10% in absolute accuracy for tasks with 20 labels and 5 labels per class respectively.

3) 2-D graph convolution is effective.

• For datasets with good node graphs such as 20 NG and L-Cora, DSGC ($\boldsymbol{GXF}$) performs much better than either DSGC ($\boldsymbol{GX}$) or DSGC ($\boldsymbol{XF}$). DSGC ($\boldsymbol{GXF}$) with PPMI achieves the best performance. On 20 NG, it improves upon the best baseline by more than 5% for tasks with 20 labels per class, and about 4% for tasks with 5 labels per class, both in absolute accuracy. On L-Cora, it is comparable with the best baseline. On both datasets, for tasks with 20 labels per class, DSGC ($\boldsymbol{GXF}$) with Emb also outperforms all the baselines; for tasks with 5 labels per class, it outperforms all the baselines except DGI. This shows that node graph convolution and attribute graph convolution can complement each other and lead to significant performance gain.

• For datasets with bad node graphs such as Wiki, DSGC ($\boldsymbol{GXF}$) significantly improves upon DSGC ($\boldsymbol{GX}$) and outperforms all the baselines except GraphSAGE, which uses extra labeled data for validation while DSGC uses none. Nevertheless, since node graph convolution degrades performance on such a noisy graph, DSGC ($\boldsymbol{XF}$) performs better than DSGC ($\boldsymbol{GXF}$).

Table 3: Clustering performance.

Datasets			20 NG		L-Cora		Wiki	
Methods	$\boldsymbol{G}$	$\boldsymbol{F}$	Acc(%)	NMI(%)	Acc(%)	NMI(%)	Acc(%)	NMI(%)
Spectral	✗	✗	25.29 ± 1.01	28.18 ± 0.74	28.22 ± 1.01	11.61 ± 0.04	29.25 ± 0.00	21.83 ± 0.00
GAE	✓	✗	38.92 ± 1.39	44.58 ± 0.40	34.45 ± 0.76	22.38 ± 0.18	33.78 ± 0.32	22.88 ± 0.20
VGAE	✓	✗	25.04 ± 0.81	25.72 ± 0.77	29.45 ± 1.25	17.53 ± 0.15	33.83 ± 0.45	21.46 ± 0.19
MGAE	✓	✗	47.83 ± 2.33	**56.14** ± 1.00	35.87 ± 0.97	30.57 ± 0.98	32.73 ± 1.16	27.95 ± 2.29
ARGE	✓	✗	42.04 ± 0.50	44.13 ± 0.91	36.07 ± 0.05	27.74 ± 0.01	26.49 ± 0.10	17.17 ± 0.05
ARVGE	✓	✗	21.10 ± 0.61	21.79 ± 0.49	26.45 ± 0.03	12.94 ± 0.01	33.82 ± 0.13	21.42 ± 0.11
AGC	✓	✗	38.83 ± 0.84	47.08 ± 1.57	**41.76** ± 0.01	**33.65** ± 0.01	32.74 ± 0.01	24.90 ± 0.01
DSGC ($\boldsymbol{GX}$)	✓	✗	38.42 ± 0.66	46.28 ± 0.93	38.26 ± 0.02	30.66 ± 0.02	31.43 ± 0.09	24.16 ± 0.18
DSGC ($\boldsymbol{XF}$)	✗	Emb	28.99 ± 0.06	33.22 ± 0.10	30.80 ± 0.56	17.46 ± 0.21	**35.45** ± 0.91	**33.44** ± 0.66
	✗	PPMI	**48.36** ± 2.40	53.27 ± 2.17	36.46 ± 0.06	22.53 ± 0.03	**38.10** ± 0.01	**36.07** ± 0.02
DSGC ($\boldsymbol{GXF}$)	✓	Emb	43.40 ± 0.66	50.97 ± 0.58	40.75 ± 0.02	**33.05** ± 0.04	30.50 ± 0.01	25.48 ± 0.03
	✓	PPMI	**52.25** ± 1.97	**61.34** ± 1.07	**41.24** ± 0.04	30.92 ± 0.01	31.37 ± 0.08	26.06 ± 0.20

⋆ ✓ and ✗ indicate using/not using $\boldsymbol{G}$ or $\boldsymbol{F}$.

- As shown in Table 2, for node classification with 20 labels per class on all three datasets, after incorporating DSGC, the performances of GCN, GAT, LDS, and GraphSAGE are substantially improved in most cases. This further verifies the effectiveness of 2-D graph convolution.

7.2 Node Clustering

Baselines. We test the proposed node clustering method by DSGC (section 5.1) with or without using $\boldsymbol{G}$ and $\boldsymbol{F}$ in five cases. We compare DSGC with strong baselines for attributed graph clustering including GAE and VGAE (Kipf & Welling, 2016), MGAE (Wang et al., 2017), ARGE and ARVGE (Pan et al., 2018), and AGC (Zhang et al., 2019). We also compare with spectral clustering (Spectral) that operates on a similarity matrix constructed by applying linear kernel on the node representations.

Performance. To evaluate the clustering performance, we adopt two widely-used metrics (Aggarwal & Reddy, 2014): clustering accuracy (Acc) and normalized mutual information (NMI). The results are shown in Table 3, where the top 2 results are highlighted in bold. The following observations can be made. **1)** Attribute graph convolution works very well. On 20 NG, DSGC ($\boldsymbol{XF}$) with PPMI already outperforms most baselines by a large margin. On Wiki, DSGC ($\boldsymbol{XF}$) with PPMI or Emb significantly outperforms all the baselines. **2)** Similar to the classification tasks, 2-D graph convolution is effective. On 20 NG, DSGC ($\boldsymbol{GXF}$) with PPMI can further improve upon the very strong performance of DSGC ($\boldsymbol{XF}$) and performs the best; On L-Cora, DSGC ($\boldsymbol{GXF}$) with PPMI or Emb improves upon either DSGC ($\boldsymbol{GX}$) or DSGC ($\boldsymbol{XF}$) and outperforms most baselines significantly. On Wiki, DSGC ($\boldsymbol{XF}$) performs better than DSGC ($\boldsymbol{GXF}$), due to the low-quality hyperlink graph as explained above.

For both classification and clustering, we observe that in most cases DSGC with PPMI can achieve better performance than with Emb. This shows the effectiveness of PPMI in capturing meaningful word relations based on information theory and statistics (Church & Hanks, 1989), whereas Emb only relies on a distance metric for measuring word similarity.

8 Conclusion

In this paper, we have proposed to model attributed graphs with 2-D graph convolution. We have demonstrated theoretically and empirically that by exploiting attribute relations in addition to node relations, a simple and efficient dimensionwise separable 2-D graph convolution (DSGC) can learn better node representations than existing methods based on regular 1-D graph convolution on noisy and sparse real-word networks. We believe 2-D graph convolution is a promising tool for attributed graph learning, and there is much left to be explored. In future work, we plan to further investigate the construction of attribute affinity graphs for different types of attributed networks, apply DSGC to solve various practical problems, and design new efficient filters for 2-D graph convolution.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, pp. 21–29, 2019.

Charu C Aggarwal and Chandan K Reddy. *Data Clustering: Algorithms and Applications*. CRC Press, Boca Raton, 2014.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *NeurIPS*, pp. 1993–2001, 2016.

Amir Bakarov. A survey of word embeddings evaluation methods. *CoRR*, abs/1801.09536, 2018.

M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56(1):209–239, 2004.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(1):2399–2434, 2006.

Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. *Semi-supervised Learning*, pp. 193–216, 2006.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pp. 19–26, 2001.

A. Blum, J. Lafferty, M.R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *ICML*, pp. 13, 2004.

Aleksandar Bojchevski and Stephan Günnemann. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *AAAI*, pp. 2738–2745, 2018.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *ICLR*, 2014.

HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE TKDE*, 30(9):1616–1637, 2018.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, pp. 1145–1152, 2016.

O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. In *NeurIPS*, pp. 601–608, 2003.

Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *ICML*, pp. 941–949, 2018a.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018b.

Hong Cheng, Yang Zhou, and Jeffrey Xu Yu. Clustering large attributed graphs: A balance between structural and attribute similarities. *ACM TKDD*, 5(2):12:1–12:33, 2011.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information and lexicography. In *27th Annual Meeting of the Association for Computational Linguistics, 26-29 June 1989, University of British Columbia, Vancouver, BC, Canada, Proceedings.*, pp. 76–83, 1989.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. Adversarial network embedding. In *AAAI*, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, pp. 3844–3852, 2016.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, pp. 2224–2232, 2015.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *ICML*, pp. 1972–1982, 2019.

Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

Charles M Grinstead and James Laurie Snell. *Introduction to probability*. American Mathematical Soc., 2012.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD*, pp. 855–864. ACM, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pp. 1024–1034, 2017.

Matthias Hein and Markus Maier. Manifold denoising. In *NeurIPS*, pp. 561–568, 2007.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In *SIAM ICDM*, pp. 633–641, 2017.

T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pp. 290–297, 2003.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional network-s. In *ICLR*, 2017.

Takashi Kurokawa, Taihei Oki, and Hiromichi Nagao. Multi-dimensional graph fourier transform. *arXiv preprint arXiv:1712.07811*, 2017.

Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, pp. 331–339, 1995.

Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. Label efficient semi-supervised learning via graph filtering. In *CVPR*, pp. 9582–9591, 2019.

Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. Community detection in attributed graphs: an embedding approach. In *AAAI*, pp. 338–345, 2018.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *CoRR*, abs/1901.01484, 2019.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *ICML*, pp. 4212–4221, 2019a.

Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. A flexible generative framework for graph-based semi-supervised learning. *CoRR*, abs/1905.10769, 2019b.

Andrew McCallumzy, Kamal Nigamy, Jason Renniey, and Kristie Seymorey. Building domain-specific search engines with machine learning techniques. In *Proceedings of the AAAI Spring Symposium on Intelligent Agents in Cyberspace*, pp. 28–39. Citeseer, 1999.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pp. 5425–5434, 2017.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *IJCAI*, pp. 1895–1901, 2016.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, pp. 2609–2615, 2018.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, pp. 1532–1543, 2014.

Pietro Perona and William Freeman. A factorization approach to grouping. In *ECCV*, pp. 655–670, 1998.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *ACM SIGKDD*, pp. 701–710, 2014.

Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: graph markov neural networks. In *ICML*, pp. 5241–5250, 2019.

Claudio Saccá, Michelangelo Diligenti, and Marco Gori. Collective classification using semantic based regularization. In *IEEE ICMLA*, volume 1, pp. 283–286, 2013.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *NeurIP-S*, pp. 945–952, 2002.

Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *UAI*, pp. 485–492, 2002.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *CIKM*, pp. 889–898, 2017.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *ACM SIGKDD*, pp. 1225–1234, 2016.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*, 2018.

Robert West and Jure Leskovec. Human wayfinding in information networks. In *WWW*, pp. 619–628, 2012.

Robert West, Joelle Pineau, and Doina Precup. Wikispeedia: An online game for inferring semantic distances between concepts. In *IJCAI*, pp. 1598–1603, 2009.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *ICML*, pp. 1168–1175, 2008.

Xiaoming Wu, Zhenguo Li, Anthony M. So, John Wright, and Shih-fu Chang. Learning with Partially Absorbing Random Walks. In *NeurIPS*, pp. 3077–3085, 2012.

Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *AAAI*, pp. 2149–2155, 2014.

Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A model-based approach to attributed graph clustering. In *ACM SIGMOD*, pp. 505–516, 2012.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, pp. 2111–2117, 2015.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pp. 40–48, 2016.

Fanghua Ye, Chuan Chen, and Zibin Zheng. Deep autoencoder-like nonnegative matrix factorization for community detection. In *CIKM*, pp. 1393–1402, 2018.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *UAI*, pp. 339–349, 2018a.

T. Zhang and R.K. Ando. Analysis of spectral kernel design based semi-supervised learning. In *NeurIPS*, pp. 1601–1608, 2006.

Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. In *IJCAI*, pp. 4327–4333, 2019.

Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. *CoRR*, abs/1811.11103, 2018b.

Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. ANRL: attributed network representation learning via deep neural networks. In *IJCAI*, pp. 3155–3161, 2018c.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *CoRR*, abs/1812.04202, 2018d.

Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NeurIPS*, pp. 321–328, 2004.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729, 2009.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, pp. 689–698, 2010.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pp. 912–919, 2003.

Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *WWW*, pp. 499–508, 2018.

Appendices

Appendix A provides details about the experiments in section 7 and presents additional experimental results. Appendix B, C and D provide proofs of the theorems in section 4.

Appendix A Experiment Details and Supplementary Experiments

A.1 Dataset Details

Table 4: Classes of each dataset.

Datasets	Classes			
20 NG	talk.politics.guns	sci.crypt	rec.autos	comp.graphics
	talk.politics.mideast	sci.electronics	rec.motorcycles	comp.os.ms-windows.misc
	talk.politics.misc	sci.med	rec.sport.baseball	comp.sys.ibm.pc.hardware
	talk.religion.misc	sci.space	rec.sport.hockey	comp.sys.mac.hardware
	soc.religion.christian	alt.atheism	misc.forsale	comp.windows.x
Wiki	Everyday Life	Religion	Science	Design and Technology
	Countries	People	History	Language and literature
	Citizenship			
L-Cora	Artificial Intelligence	Networking	Encryption and Compression	
	Data Structures Algorithms and Theory		Hardware and Architecture	
	Operating Systems	Programming	Human Computer Interaction	
	Information Retrieval	Databases		

20 Newsgroups (20 NG) Lang (1995) is an email discussion group, where each node is an email and there are 18846 emails in total. Each email is represented by an 11697-dimension tf-idf feature vector. Two emails are connected by an edge if they replies the same one. These emails are categorized into 20 classes as listed in Table 4.

Wikispeedia (Wiki) West et al. (2009); West & Leskovec (2012) is a webpage network in which the nodes are 3767 Wikipedia webpages, and the edges are web hyperlinks. Each webpage is described by a 18316-dimension tf-idf vector. We remove several tiny classes from the dataset, so the webpages distribute more evenly across the remaining 9 categories, which are also listed in Table 4.

Large Cora (L-Cora) McCallumzy et al. (1999) is a citation network in which the nodes are computer science research papers represented by 3780 dimension of tf-idf values. Two papers are connected by an undirected edge if and only if one cites the other. These citation links form a node graph. The nodes in the dataset are originally categorized into a topic hierarchical tree with 73 leaves. After removing the papers that belong to no topic and the ones that have no authors or title, a subset of 11881 papers is obtained Saccá et al. (2013). These papers are then classified into 10 highest-level topics in the topic hierarchy, as listed in Table 4. We name this dataset “Large Cora” to distinguish it from the “Cora” dataset with 2708 papers used in Kipf & Welling (2017); Yang et al. (2016); Sen et al. (2008). Note that we did not test on this ”Cora” and the “Citeseer” and “PubMed” datasets as in Kipf & Welling (2017); Yang et al. (2016); Sen et al. (2008), because the attributes (words) are not provided in these datasets.

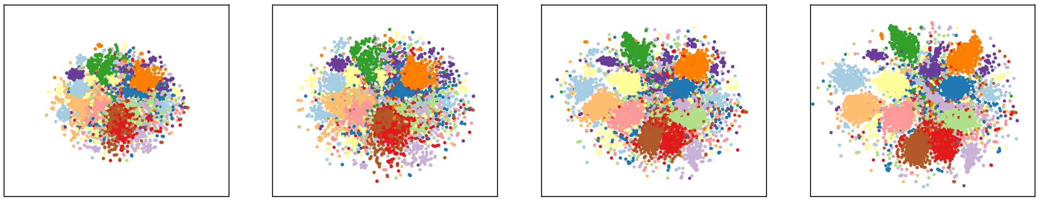
The statistics of the datasets are summarized in Table 5, where the last row shows the intra-class edge ratio of the node graph of each dataset, which can reflect the quality of the graph.

A.2 Visualization

In Figure 3, we visualize the results of performing graph convolution on the node features of 20 NG dataset by t-SNE. It can be seen that graph convolution can successfully reduce the overlap among classes, and 2-D graph convolution is more effective than 1-D.

Table 5: Dataset statistics.

Dataset	20 NG	Wiki	L-Cora
Vertices	18846	3767	11881
Edges	147034	129597	64898
Classes	20	9	10
Features	11697	18316	3780
Connected Components	8504	303	833
Intra-class edge ratio	96.8%	38.0%	76.5%



(a) $\boldsymbol{X}$ (b) $\boldsymbol{GX}$ (c) $\boldsymbol{XF}$ (d) $\boldsymbol{GXF}$

Figure 3: t-SNE visualization of "20 NG". (a) Raw features; (b) Results of node graph convolution; (c) Results of affinity graph convolution; (d) Results of 2-D graph convolution (DSGC).

A.3 Training Time

The training time is summarized in Figure 4. We can see that DSGC is several times faster than GCN, GAT, DGI and GraphSAGE. This is because DSGC only performs graph convolution once before training, while others need to do graph convolution in every training step.

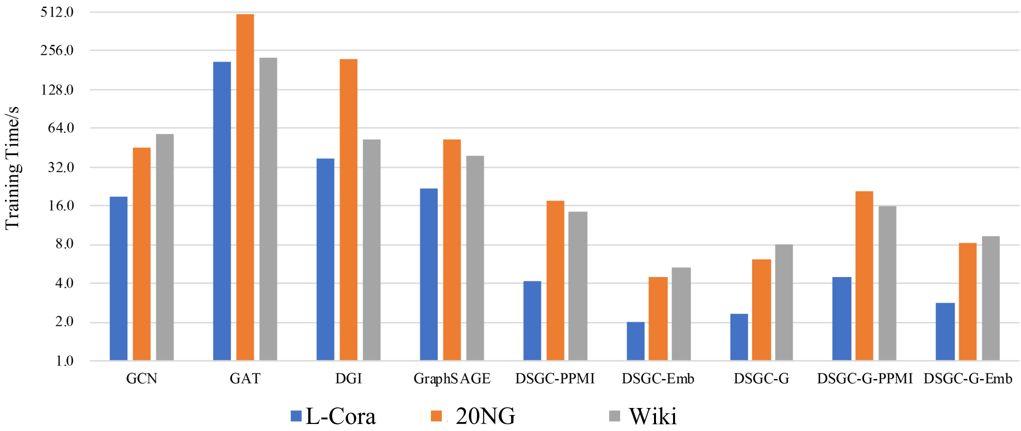


Figure 4: Training time of DSGC and baselines.

A.4 Parameter Settings for Node Classification

For DSGC, we test positive point-wise mutual information (PPMI) and word embedding based k-NN graphs (Emb) for constructing the attribute affinity graph. For constructing the PPMI graph, we set the context window size to 20 and use the inverse distance as co-occurrence weight. For constructing the Emb graph, we use GloVe Pennington et al. (2014) to learn word embeddings and set the number of nearest neighbours $k = 20$. The classifier of DSGC is a two-layer multi-layer perceptron (MLP) with 64 hidden units. The MLP is trained with Adam Optimizer for 200 epochs with 0.015 learning rate. For the sake of fair comparison to all baselines and to make sure our method can work well in practice, we do not use extra labeled data for validation on our methods and the baselines, unless otherwise stated. Instead, we always select the model with the lowest train loss in the 200 epochs. In each run, the dataset is randomly split into a labeled set and an unlabeled set. The classification results are averaged over 10 runs.

Hyperparameters of all models, including our methods and baselines, are tuned to its best performance by grid search. DSGC follows the network setting and hyperparameters of GCN, which are a 2-layer structure with 64 hidden units and 0.015 learning rate. We set the balancing parameter α in LP as 100. We largely follow Velickovic et al. (2018) to set network structures and hyperparameters of GAT, which are 2 layers, 8 heads, 8 neurons of each head, 0.005 learning rate, 400 epochs, and 0.3 dropout. Our setting of DGI is the same as Velickovic et al. (2019), except that the number of hidden units is decreased to 128, so it can be fitted into our GPU memory. Our setting of GraphSAGE is the same as Hamilton et al. (2017) (using 30% data as validation set), except that the number of samples in layer 2 is reduced to 5, also due to limited GPU memory. Parameters of GLP are same as Li et al. (2019). For LDS Franceschi et al. (2019), since it cannot scale to the size of 20 NG (out of GPU memory), we follow the authors to select a 10-category subset of 20 NG. For each dataset, both the training set and validation set for LDS contain 20 labels per class.

All models are tested on a platform with Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz and single NVIDIA(R) GeForce(R) GTX 1080 Ti.

A.5 Parameter Settings for Node Clustering

For our method DSGC, the attribute affinity graphs PPMI and Emb are constructed in the same way as in node classification. For other baselines, we follow the parameter settings described in the original papers. In particular, for GAE and VGAE (Kipf & Welling, 2016), we construct encoders with a 32-neuron hidden layer and a 16-neuron embedding layer, and train the encoders for 200 iterations using the Adam algorithm with a learning rate of 0.01. For MGAE (Wang et al., 2017), the corruption level p is 0.4, the number of layers is 3, and the parameter λ is 10^{-5}. For ARGE and ARVGE (Pan et al., 2018), we construct encoders with a 32-neuron hidden layer and a 16-neuron embedding layer. The discriminators are built by two hidden layers with 16 neurons and 64 neurons respectively. We train all the autoencoder-related models for 200 iterations and optimize them using the Adam algorithm. The learning rate and the discriminator learning rate are both 0.001. For AGC (Zhang et al., 2019), the maximum iteration number is 60. For fair comparison, these baselines also adopt spectral clustering as DSGC to get the clustering results. We repeat each method for 10 times, and show the mean clustering results and the standard deviations.

Appendix B Proof of Theorem 1 in Section 4.1

We suppose that nodes from the same class are connected with probability p, and nodes from different classes are connected with probability q, i.e., the adjacency matrix $\boldsymbol{A}^{(1)}$ of node graph obeys

$$\Pr(a_{ij}=1)=\left\{\begin{array}{ll} p, & \text{if } y_i=y_j \\ q, & \text{if } y_i\neq y_j \end{array}\right. ,\quad \Pr(a_{ij}=0)=\left\{\begin{array}{ll} 1-p, & \text{if } y_i=y_j \\ 1-q, & \text{if } y_i\neq y_j \end{array}\right. \tag{15}$$

and $\boldsymbol{G}=\boldsymbol{D}^{-1}\boldsymbol{A}^{(1)}$ is a stochastic matrix. We also assumes that classes are balanced, i.e., $\Pr(\mathbb{Y}=k)=1/K$ for all k, then we have following theorem.

Theorem 1. *When q is sufficiently small, $\mathbb{Z}$ has an* IntraVar/InterVar *ratio less than or equal to that of $\mathbb{X}$, i.e.,*

$$\frac{\mathrm{E}\left[\mathrm{Var}\left(\mathbb{Z}|\mathbb{Y}\right)\right]}{\mathrm{Var}\left(\mathrm{E}\left[\mathbb{Z}|\mathbb{Y}\right]\right)}\leq\frac{\mathrm{E}\left[\mathrm{Var}\left(\mathbb{X}|\mathbb{Y}\right)\right]}{\mathrm{Var}\left(\mathrm{E}\left[\mathbb{X}|\mathbb{Y}\right]\right)}. \tag{16}$$

Proof. The proof consists of two parts. In the first part, we prove that inter-class variance is unchanged after node graph convolution, when q approximates 0, i.e.,

$$\lim_{q\to 0}\mathrm{Var}\left(\mathrm{E}\left[\mathbb{Z}|\mathbb{Y}\right]\right)=\mathrm{Var}\left(\mathrm{E}\left[\mathbb{X}|\mathbb{Y}\right]\right). \tag{17}$$

In the second part, we prove that intra-class variance becomes smaller after node graph convolution, i.e.,

$$\mathrm{E}\left[\mathrm{Var}\left(\mathbb{Z}|\mathbb{Y}\right)\right]\leq\mathrm{E}\left[\mathrm{Var}\left(\mathbb{X}|\mathbb{Y}\right)\right], \tag{18}$$

when G is a stochastic matrix.

Part 1. Inter-class variance is unchanged. Since

$$\boldsymbol{z}_i=\sum_j G_{ij}\boldsymbol{x}_j,$$

we have

$$\begin{aligned}
\mathrm{E}\left[\boldsymbol{z}_i|y_i=k\right] &= \sum_j \mathrm{E}\left[G_{ij}\right]\mathrm{E}\left[\boldsymbol{x}_j\right] \qquad \text{\# by linearity of expectation}\\
&= \sum_{j,y_j=k}\mathrm{E}\left[G_{ij}\right]\mathrm{E}\left[\boldsymbol{x}_j\right]+\sum_{j,y_j\neq k}\mathrm{E}\left[G_{ij}\right]\mathrm{E}\left[\boldsymbol{x}_j\right]\\
&= \frac{\sum_{j,y_j=k}\mathrm{E}\left[a_{ij}\right]\mathrm{E}\left[\boldsymbol{x}_j\right]+\sum_{j,y_j\neq k}\mathrm{E}\left[a_{ij}\right]\mathrm{E}\left[\boldsymbol{x}_j\right]}{\sum_j\mathrm{E}\left[a_{ij}\right]}\\
&= \frac{p\sum_{j,y_j=k}\mathrm{E}\left[\mathbb{X}|\mathbb{Y}=k\right]+q\sum_{j,y_j\neq k}\mathrm{E}\left[\boldsymbol{x}_j\right]}{\frac{N}{K}(p-q)+Nq}\\
&= \frac{\frac{N}{K}p\mathrm{E}\left[\mathbb{X}|\mathbb{Y}=k\right]+q\sum_j\mathrm{E}\left[\boldsymbol{x}_j\right]-q\sum_{j,y_j=k}\mathrm{E}\left[\boldsymbol{x}_j\right]}{\frac{N}{K}(p-q)+Nq}\\
&= \frac{\frac{N}{K}(p-q)\mathrm{E}\left[\mathbb{X}|\mathbb{Y}=k\right]+Nq\mathrm{E}\left[\mathbb{X}\right]}{\frac{N}{K}(p-q)+Nq}\\
&= \frac{(p-q)\mathrm{E}\left[\mathbb{X}|\mathbb{Y}=k\right]+Kq\mathrm{E}\left[\mathbb{X}\right]}{(p-q)+Kq}
\end{aligned}$$

When q approximates 0, $\mathrm{E}\left[\boldsymbol{z}_i|y_i=k\right]$ will approximate $\mathrm{E}\left[\mathbb{X}|\mathbb{Y}=k\right]$, so

$$\begin{aligned}
\mathrm{E}\left[\mathbb{Z}|\mathbb{Y}=k\right] &= \sum_{i,y_i=k}\Pr(\mathbb{Z}=\boldsymbol{z}_i|y_i=k)\mathrm{E}\left[\boldsymbol{z}_i|y_i=k\right]\\
&= \sum_{i,y_i=k}\Pr(\mathbb{Z}=\boldsymbol{z}_i|y_i=k)\mathrm{E}\left[\mathbb{X}|\mathbb{Y}=k\right]\\
&= \mathrm{E}\left[\mathbb{X}|\mathbb{Y}=k\right].
\end{aligned}$$

Hence,

$$\mathrm{Var}\left(\mathrm{E}\left[\mathbb{Z}|\mathbb{Y}\right]\right)=\mathrm{Var}\left(\mathrm{E}\left[\mathbb{X}|\mathbb{Y}\right]\right). \tag{19}$$

Part 2. Intra-class variance becomes smaller. Denote by $\text{Cov}\left(\cdot,\cdot\right)$ the covariance of two random variables. We have

$$\begin{aligned}
\text{Var}\left(\sum_j G_{ij}\boldsymbol{x}_j\right) &= \sum_j G_{ij}^2 \text{Var}\left(\boldsymbol{x}_j\right) + \sum_{j,l} G_{ij}G_{il}\text{Cov}\left(\boldsymbol{x}_j,\boldsymbol{x}_l\right) && \text{\# property of variance} \\
&\leq \sum_{j,l} G_{ij}G_{il}\sqrt{\text{Var}\left(\boldsymbol{x}_j\right)}\sqrt{\text{Var}\left(\boldsymbol{x}_l\right)} && \text{\# property of covariance} \\
&= \left(\sum_j G_{ij}\sqrt{\text{Var}\left(\boldsymbol{x}_j\right)}\right)^2 .
\end{aligned}$$

Since

$$\begin{aligned}
\text{Var}\left(\boldsymbol{z}_i | y_i = k\right) &= \text{Var}\left(\sum_j G_{ij}\boldsymbol{x}_j \middle| y_j = k\right) \\
&\leq \left(\sum_j G_{ij}\sqrt{\text{Var}\left(\boldsymbol{x}_j | y_j = k\right)}\right)^2 && \text{\# by the above inequality} \\
&= \left(\sum_j G_{ij}\sqrt{\text{Var}\left(\mathbb{X} | \mathbb{Y} = k\right)}\right)^2 \\
&= \left(\sqrt{\text{Var}\left(\mathbb{X} | \mathbb{Y} = k\right)}\right)^2 && \text{\# since } \sum_j G_{ij} = 1 \\
&= \text{Var}\left(\mathbb{X} | \mathbb{Y} = k\right),
\end{aligned}$$

we have

$$\text{Var}\left(\mathbb{Z} | \mathbb{Y} = k\right) = \sum_{i, y_i = k} \Pr(\mathbb{Z} = \boldsymbol{z}_i | y_i = k)\text{Var}\left(\boldsymbol{z}_i | y_i = k\right) \leq \text{Var}\left(\mathbb{X} | \mathbb{Y} = k\right). \tag{20}$$

Then we have

$$\begin{aligned}
\text{E}\left[\text{Var}\left(\mathbb{Z} | \mathbb{Y}\right)\right] &= \sum_k \Pr(\mathbb{Y} = k)\text{Var}\left(\mathbb{Z} | \mathbb{Y} = k\right) \\
&\leq \sum_k \Pr(\mathbb{Y} = k)\text{Var}\left(\mathbb{X} | \mathbb{Y} = k\right) && \text{\# by the above inequality} \\
&= \text{E}\left[\text{Var}\left(\mathbb{X} | \mathbb{Y}\right)\right]. && (21)
\end{aligned}$$

Combining Eq. (19) and Eq. (21), we prove that when q is sufficiently small,

$$\frac{\text{E}\left[\text{Var}\left(\mathbb{Z} | \mathbb{Y}\right)\right]}{\text{Var}\left(\text{E}\left[\mathbb{Z} | \mathbb{Y}\right]\right)} \leq \frac{\text{E}\left[\text{Var}\left(\mathbb{X} | \mathbb{Y}\right)\right]}{\text{Var}\left(\text{E}\left[\mathbb{X} | \mathbb{Y}\right]\right)}. \tag{22}$$

□

APPENDIX C PROOF OF THEOREM 2 IN SECTION 4.2

Theorem 2. *If the attribute graph convolutional filter $\boldsymbol{F}$ is a doubly stochastic matrix, then the output of attribute graph convolution has an intra-class variance less than or equal to that of $\mathbb{X}$, i.e.,*

$$\sum_i F_{ij} = \sum_j F_{ij} = 1 \text{ and } F_{ij} \geq 0, \forall\, i, j \quad \Rightarrow \quad \mathrm{E}\left[\mathrm{Var}\left(\boldsymbol{F}^\top \mathbb{X}|\mathbb{Y}\right)\right] \leq \mathrm{E}\left[\mathrm{Var}\left(\mathbb{X}|\mathbb{Y}\right)\right].$$

Proof. We first prove a lemma that variance of each class will not increase after attribute graph convolution, i.e., $\mathrm{Var}\left(F^\top \mathbb{X}|\mathbb{Y}=k\right) \leq \mathrm{Var}\left(\mathbb{X}|\mathbb{Y}=k\right)$. Denote by $\mathrm{Cov}\left(\cdot\right)$ the covariance matrix of a random vector. Based on our definition of variance in section 4, we have

$$\begin{aligned}
\mathrm{Var}\left(\boldsymbol{F}^\top \mathbb{X}|\mathbb{Y}=k\right) &= \mathrm{Tr}\left(\mathrm{Cov}\left(\boldsymbol{F}^\top \mathbb{X}|\mathbb{Y}=k\right)\right) \\
&= \mathrm{Tr}\left(\boldsymbol{F}^\top \mathrm{Cov}\left(\mathbb{X}|\mathbb{Y}=k\right)\boldsymbol{F}\right) && \text{\# property of covariance} \\
&= \mathrm{Tr}\left(\mathrm{Cov}\left(\mathbb{X}|\mathbb{Y}=k\right)\boldsymbol{F}\boldsymbol{F}^\top\right) && \text{\# cyclic property of trace} \\
&= \sum_{ij} \mathrm{Cov}\left(\mathbb{X}_i, \mathbb{X}_j|\mathbb{Y}=k\right)(\boldsymbol{F}\boldsymbol{F}^\top)_{ij} && \text{\# property of trace} \\
&= \sum_{ij} \mathrm{Cov}_{ij}^{(k)}(\boldsymbol{F}\boldsymbol{F}^\top)_{ij} \\
&\leq \sum_{ij} \sqrt{\mathrm{Var}\left(\mathbb{X}_i|\mathbb{Y}=k\right)}\sqrt{\mathrm{Var}\left(\mathbb{X}_j|\mathbb{Y}=k\right)}\,(\boldsymbol{F}\boldsymbol{F}^\top)_{ij} && \text{\# property of covariance} \\
&= \sum_{ij} \sigma_i \sigma_j (\boldsymbol{F}\boldsymbol{F}^\top)_{ij} && \text{\# } \boldsymbol{\sigma} \in \mathbb{R}^m, \sigma_i \triangleq \sqrt{\mathrm{Var}\left(\mathbb{X}_i|\mathbb{Y}=k\right)} \\
&= \boldsymbol{\sigma}^\top \boldsymbol{F}\boldsymbol{F}^\top \boldsymbol{\sigma} \\
&\leq \|\boldsymbol{\sigma}\|_2^2 && \text{\# eigenvalues of } \boldsymbol{F} \text{ is no more than 1} \\
&= \sum_i \mathrm{Var}\left(\mathbb{X}_i|\mathbb{Y}=k\right) \\
&= \mathrm{Var}\left(\mathbb{X}|\mathbb{Y}=k\right).
\end{aligned}$$

Next, we prove the theorem with the above lemma. Denote by $\pi_k = \Pr(\mathbb{Y}=k)$ the portion of each class, then we have

$$\begin{aligned}
\mathrm{E}\left[\mathrm{Var}\left(\boldsymbol{F}^\top \mathbb{X}|\mathbb{Y}\right)\right] &= \sum_k \pi_k \mathrm{Var}\left(\boldsymbol{F}^\top \mathbb{X}|\mathbb{Y}=k\right) \\
&= \sum_k \pi_k \sum_{ij} \mathrm{Cov}_{ij}^{(k)}(\boldsymbol{F}\boldsymbol{F}^\top)_{ij} \\
&\leq \sum_k \pi_k \mathrm{Var}\left(\mathbb{X}|\mathbb{Y}=k\right) \\
&= \mathrm{E}\left[\mathrm{Var}\left(\mathbb{X}|\mathbb{Y}\right)\right]
\end{aligned}$$

□

Construction of a Doubly Stochastic Filter $\boldsymbol{F}$ Given an attribute affinity matrix $\boldsymbol{A}^{(2)}$, one could easily construct a doubly stochastic filter $\boldsymbol{F}$ by the following steps: 1) compute graph Laplacian $\boldsymbol{L}^{(2)} = \boldsymbol{D}^{(2)} - \boldsymbol{A}^{(2)}$; 2) compute doubly stochastic graph matrix $\boldsymbol{\Omega} = (\boldsymbol{I} + \boldsymbol{L}^{(2)})^{-1}$; 3) take $\boldsymbol{\Omega}$ as a new attribute affinity matrix, choose a polynomial p with nonnegative coefficients that sum to 1, and let $\boldsymbol{F} = p(\boldsymbol{\Omega})$. It can be easily seen that $\boldsymbol{F}$ is doubly stochastic.

Appendix D Proof of Theorem 3 in Section 4.2

Theorem 3. *If $\forall F_{ij} \neq 0$, $\|\boldsymbol{e}_i - \boldsymbol{e}_j\|_2 \leq \varepsilon$, then the distance between $\mathbf{e}_j$ and $\widehat{\boldsymbol{e}}_j = \sum_i F_{ij}\boldsymbol{e}_i$ is also less than or equal to ε, i.e.,*

$$\|\boldsymbol{e}_i - \boldsymbol{e}_j\|_2 \leq \varepsilon,\ \forall F_{ij} \neq 0 \quad \Rightarrow \quad \|\boldsymbol{e}_j - \widehat{\boldsymbol{e}}_j\|_2 \leq \varepsilon,$$

and ε can be arbitrarily small with a proper $\boldsymbol{F}$.

Proof.

$$\begin{aligned}
\|\boldsymbol{e}_j - \widehat{\boldsymbol{e}}_j\|_2 &= \left\|\boldsymbol{e}_j - \sum_i F_{ij}\boldsymbol{e}_i\right\|_2 \\
&= \left\|\sum_i F_{ij}(\boldsymbol{e}_j - \boldsymbol{e}_i)\right\|_2 && \text{\# since } \sum_i F_{ij} = 1 \\
&\leq \sum_i F_{ij}\,\|\boldsymbol{e}_j - \boldsymbol{e}_i\|_2 && \text{\# Cauchy-Schwarz inequality} \\
&\leq \sum_i F_{ij}\varepsilon = \varepsilon
\end{aligned}$$

Next, we prove that there exists such an F that ε is 0. This is equivalent to finding a doubly stochastic F satisfying $\sum_i F_{ij}\boldsymbol{e}_i = \boldsymbol{e}_j$ for all j. Since $F = I$ is a trivial solution, it is solvable. Denote by m the number of attributes, and denote by K the number of classes. This linear system consists of $m(K+2)$ equations and m^2 variables. In most real-world attributed networks, the number of attributes is far greater than the number of classes, so the number of variables in this linear system is greater than the number of equations. Given that it is solvable, it must have infinite number of solutions other than I. Thus, ε can be arbitrarily small with a proper $\boldsymbol{F}$. □