Beyond Graph Convolutional Network: An Interpretable Regularizer-Centered Optimization Framework

Shiping Wang[1,2], **Zhihao Wu**[1,2], **Yuhong Chen**[1,2], **Yong Chen**[3*]
[1]College of Computer and Data Science, Fuzhou University, China
[2]Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, China
[3]School of Computer Science, Beijing University of Posts and Telecommunications, China
shipingwangphd@163.com, zhihaowu1999@gmail.com, yhchen2320@163.com, alphawolf.chen@gmail.com.

Abstract

Graph convolutional networks (GCNs) have been attracting widespread attentions due to their encouraging performance and powerful generalizations. However, few work provide a general view to interpret various GCNs and guide GCNs' designs. In this paper, by revisiting the original GCN, we induce an interpretable regularizer-centerd optimization framework, in which by building appropriate regularizers we can interpret most GCNs, such as APPNP, JKNet, DAGNN, and GNN-LF/HF. Further, under the proposed framework, we devise a dual-regularizer graph convolutional network (dubbed tsGCN[1]) to capture topological and semantic structures from graph data. Since the derived learning rule for tsGCN contains an inverse of a large matrix and thus is time-consuming, we leverage the Woodbury matrix identity and low-rank approximation tricks to successfully decrease the high computational complexity of computing infinite-order graph convolutions. Extensive experiments on eight public datasets demonstrate that tsGCN achieves superior performance against quite a few state-of-the-art competitors w.r.t. classification tasks.

Introduction

Owing to the powerful ability to aggregate neighborhood information, Graph Convolutional Network (GCN) has been successfully applied to diverse domains, such as computer vision (Chen et al. 2019; Nie et al. 2020; Wang et al. 2022), recommender systems (Xu et al. 2019; Chen et al. 2022), privacy preserving (Hu et al. 2022), and traffic forecasting (Yu, Yin, and Zhu 2018; Chen et al. 2020b). Rooted in a series of theoretical foundations, GCN extends convolution operations to the non-Euclidean spaces and effectively propagates label signals, and therefore its variants have been extensively employed for a variety of graph-related tasks, including classification (Zhang et al. 2019; Yang et al. 2022; Chen et al. 2023; Wu et al. 2022), clustering (Fan et al. 2020; Zhu and Koniusz 2021) and link prediction (Chen et al. 2020a; Halliwell 2022). In a nutshell, GCN generates embeddings with the well-established graph convolutional layers gathering semantics from neighbors according to the network topology, which are revealed to be the most critical component.

Although GCN has behaved well in many machine learning tasks, lots of studies have pointed out its certain drawbacks and made efforts for further improvements. Bo et al. (Bo et al. 2021) indicated that the propagation mechanism could be considered as a special form of low-pass filter, and presented a GCN with an adaptive frequency. Zhang et al. (Zhang et al. 2021) argued that most GCN-based methods ignored the global information and proposed SHNE, which leveraged the structure and feature similarity to capture latent semantics. Wang et al. (Wang et al. 2020) revealed that the original GCN aggregated information from node neighbors inadequately, and then developed a multi-channel GCN by utilizing feature-based semantic graph. In spite of the performance boosts of these GCN variants, they didn't establish a generalized framework, i.e., these approaches understood and enhanced GCN from certain and non-generalizable perspectives, thereby they are exceedingly difficult to be further developed, and with limited interpretability.

Consequently, it is expected to construct a unified framework for various GCNs with better interpretability; however, it is a pity that this kind of work is still in shortage. Zhao and Akoglu (Zhao and Akoglu 2020) linked GCN and Graph-regularized PCA (GPCA), and then proposed a multi-layer network by stacking the GPCA layers. Zhu et al. (Zhu et al. 2021) attempted to interpret existing GCN-based methods with a unified optimization framework, under which they devised an adjustable graph filter for a new GCN variant. Yang et al. (Yang et al. 2021) designed a family of graph convolutional layers inspired by the updating rules of two typical iterative algorithms. Although these efforts have contributed to better understanding of GCNs, they only explained GCNs in partial aspects, promoting the expectation of a more comprehensive analysis of GCNs.

To tackle the aforementioned issues, this paper induces an interpretable regularizer-centered optimization framework, which provides a novel perspective to digest various GCNs, i.e., this framework captures the common essential properties of existing state-of-the-art GCN variants and could defines them just by devising different regularizers. Moreover, in light of the analyses on current representative GCNs, we find that most of the existing approaches only consider capturing the topological regularization, while the feature-based

[*]Corresponding author
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
[1]Its code is available at https://github.com/ZhihaoWu99/tsGCN and the supplementary material is uploaded to https://arxiv.org/abs/2301.04318.

semantic structure is underutilized, and hence this motivates us to design a dual-regularizer graph convolutional network (called tsGCN) within the regularizer-centered optimization framework for the fullest explorations of both structures and semantics from graph data. Due to the high computational complexity of infinite-order graph convolutions, the unified framework provides a straightforward way employing truncated polynomials to approximate the graph Laplacian, similar to the truncated Chebyshev polynomials by vanilla GCN, restricting the message passing of a single graph convolution to the first-order neighborhood. And we also design an efficient way to perform the infinite-order graph convolution.

The main contributions of this paper can be summarized as the following three aspects:

- Propose a regularizer-centered constrained optimization framework, which interprets various existing GCNs with specific regularizers.
- Establish a new dual-regularizer graph convolutional network (tsGCN), which exploits topological and semantic structures of the given data; and develop an efficient algorithm to reduce the computational complexity of solving infinite-order graph convolutions.
- Conduct a series of experiments to show that tsGCN performs much better than many SOTA GCNs, and also consumes much less time than the newly GNN-HF/LF.

Related Work

Graph Convolutional Networks

The original GCN was first introduced by Kipf and Welling (Kipf and Welling 2017), who generalized the convolution operations from the Euclidean domain to the non-Euclidean domain. SGC (Wu et al. 2019) assumed that the nonlinear transform of GCN was not that significant, and then devised a simplified GCN by removing the nonlinear activation functions and collapsing the weight matrices. PPNP (Klicpera, Bojchevski, and Günnemann 2019) employed the relationship between PageRank and GCN for the improvement on the propagation mechanism of GCN, and an iterative version called APPNP was further proposed to reduce the high computational complexity. Attempting to adaptively learn the influence radii for each node and task, JKNet (Xu et al. 2018) combined various aggregations at the last layer and was able to learn representations of different orders for graph substructures. GNN-LF and GNN-HF (Zhu et al. 2021) considered the low-pass and the high-pass filter as the convolution kernels to improve GCN's expressive power, respectively. AdaGCN (Sun, Zhu, and Lin 2021) leveraged Adaboost strategy for the enhancement of GCN, allowing information to be shared between layers. To sum up, a main characteristic of these methods is exploring GCN from the perspectives of redesigning information aggregation strategies or modifying graph convolutions, and few work try to construct a unified framework to interpret various GCNs and reveal the underlying common principles.

Further Insights on GCNs

Quite a few studies have been devoted to explore the mechanisms of GCN for deeper insights. Li, Han, and Wu (Li, Han, and Wu 2018) indicated that the convolutional operation of GCN was a special form of Laplacian smoothing, attributed to which GCN suffered from the so-called over-smoothing problem. Specifically, the performance of GCN will decrease as the number of layers increases, which has been validated by many other studies. However, Liu, Gao, and Ji (Liu, Gao, and Ji 2020) held a different opinion that the entanglement of two steps in GCN damaged the performance of the deep GCN, where the two steps were explained as propagation and transformation. Based on this view, they decoupled the two operations and further presented a deeper GCN. Zhu et al. (Zhu et al. 2021) also decomposed the convolution operation of GCN into two separate stages, called aggregation and transformation, and focused on the aggregation process, formulating an optimization objective to interpret it. Yang et al. (Yang et al. 2019) explored network topology refinement, leveraging a topology optimization process for the explanation. Oono and Suzuki (Oono and Suzuki 2020) analyzed the forward propagation of GCN and interpreted it with a specific dynamical system, linking GCN to the underlying topological structures. Overall, these studies have contributed to the interpretability of GCNs, and also let researchers better understand GCNs. In this paper, we build a unified optimization framework from a novel view of graph regularizers to interpret and understand GCNs.

Mathematical Notations

For the convenience of formal descriptions, derivations, and analyses, necessary notations are narrated as below. A graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ marks the vertex set with $|\mathcal{V}| = N$ (N is the total number of nodes in graph $\mathcal{G}$), $\mathcal{E}$ marks the edge set, and $\mathbf{A} = [\mathbf{A}_{ij}]_{N\times N}$ marks an affinity matrix of which $\mathbf{A}_{ij}$ measures the similarity between the i-th and the j-th node. In addition, $\mathbf{D} = [\mathbf{D}_{ij}]_{N\times N}$ represents the degree matrix of $\mathcal{G}$ with $\mathbf{D}_{ii} = \sum_{j=1}^{N} \mathbf{A}_{ij}$, and then the normalized symmetrical graph Laplacian of $\mathcal{G}$ is computed as $\widetilde{\mathbf{L}} = \mathbf{I} - \widetilde{\mathbf{A}}$ with $\widetilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$.

Revisiting Graph Convolutional Network

For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, the SVD of its graph Laplacian is $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U} \in \mathbb{R}^{N\times N}$ is comprised of orthonormal eigenvectors and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_N)$ is a diagonal matrix with λ_i denoting the i-th eigenvalue and $\lambda_i \geq \lambda_j$ ($i = 1, \cdots, N$). Essentially, this decomposition induces a Fourier transform on the graph domain, where eigenvectors correspond to Fourier components and eigenvalues represent frequencies of the graph. For an input signal $\mathbf{x} \in \mathbb{R}^N$ defined on the graph $\mathcal{G}$, the corresponding graph Fourier transform of $\mathbf{x}$ is $\widehat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$, and its inverse transform is derived as $\mathbf{x} = \mathbf{U}\widehat{\mathbf{x}}$. Consequently, the graph convolution between the signal $\mathbf{x}$ and the filter $\mathbf{g} \in \mathbb{R}^N$ is

$$\mathbf{g} * \mathbf{x} = \mathbf{U}(\widehat{\mathbf{g}} \odot \widehat{\mathbf{x}}) = \mathbf{U}((\mathbf{U}^\top\mathbf{g}) \odot (\mathbf{U}^\top\mathbf{x})), \tag{1}$$

where $\odot$ is the Hadamard product between two vectors. Particularly, denoting $\mathbf{g}_{\mathbf{\Theta}} = \text{diag}(\mathbf{\Theta}) := \mathbf{U}^\top\mathbf{g}$ parameterized by $\mathbf{\Theta} \in \mathbb{R}^N$, the graph convolution between $\mathbf{x}$ and $\mathbf{g}$ can be rewritten as

$$\mathbf{g} * \mathbf{x} = \mathbf{U}(\widehat{\mathbf{g}} \odot \widehat{\mathbf{x}}) = \mathbf{U}\mathbf{g}_{\mathbf{\Theta}}\mathbf{U}^\top\mathbf{x}, \tag{2}$$

Methods	Propagation Rules	Regularizer $\mathcal{L}(\mathbf{H}^{(l)};\mathcal{G})$	Projective Set
GCN	$\mathbf{H}^{(l)} = \sigma\left(\widehat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$	$\mathrm{Tr}\left({\mathbf{H}^{(l)}}^\top\widetilde{\mathbf{L}}\mathbf{H}^{(l)}\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}_+, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}\end{cases}$
SGC	$\mathbf{H}^{(l)} = \sigma\left(\widehat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$	$\mathrm{Tr}\left({\mathbf{H}^{(l)}}^\top\widetilde{\mathbf{L}}\mathbf{H}^{(l)}\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}\end{cases}$
APPNP	$\mathbf{H}^{(l)} = \sigma\left((1-\alpha)\widehat{\mathbf{A}}\mathbf{H}^{(l-1)} + \alpha\mathbf{H}^{(0)}\right)$	$\mathrm{Tr}\left(\frac{1}{1-\alpha}{\mathbf{H}^{(l)}}^\top\widehat{\mathbf{A}}^{-1}(\mathbf{H}^{(l)} - 2\alpha\mathbf{H}^{(0)})\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}\end{cases}$
JKNet	$\mathbf{H}^{(l)} = \sigma\left(\sum_{k=1}^K \alpha_k\widehat{\mathbf{A}}^k\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$	$\mathrm{Tr}\left({\mathbf{H}^{(l)}}^\top\widehat{\mathbf{A}}^{-1}(\mathbf{I}+\beta\widetilde{\mathbf{L}})\mathbf{H}^{(l)}\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}\end{cases}$
DAGNN	$\mathbf{H}^{(l)} = \sigma\left(\sum_{k=0}^K \alpha_k\widehat{\mathbf{A}}^k\mathbf{H}^{(0)}\right)$	$\mathrm{Tr}\left({\mathbf{H}^{(l)}}^\top(\mathbf{I}+\beta\widetilde{\mathbf{L}})\mathbf{H}^{(l)}\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}\end{cases}$
GNN-HF	$\mathbf{H}^{(l)} = \sigma\left((\mathbf{I}+\alpha\widehat{\mathbf{L}})^{-1}(\mathbf{I}+\beta\widehat{\mathbf{L}})\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$	$\mathrm{Tr}\left({\mathbf{H}^{(l)}}^\top(\mathbf{I}+\beta\widehat{\mathbf{L}})^{-1}(\mathbf{I}+\alpha\widehat{\mathbf{L}})\mathbf{H}^{(l)}\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}_+, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}.\end{cases}$
GNN-LF	$\mathbf{H}^{(l)} =$ $\sigma\left((\mathbf{I}+\alpha\widehat{\mathbf{A}})^{-1}(\mathbf{I}+\beta\widehat{\mathbf{A}})\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$	$\mathrm{Tr}\left({\mathbf{H}^{(l)}}^\top(\mathbf{I}+\beta\widehat{\mathbf{A}})^{-1}(\mathbf{I}+\alpha\widehat{\mathbf{A}})\mathbf{H}^{(l)}\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}_+, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}\end{cases}$
tsGCN	$\mathbf{H}^{(l)} = \sigma\left((\mathbf{I}+\alpha\widetilde{\mathbf{L}}_{\mathcal{G}}+\beta\widetilde{\mathbf{L}}_{\mathcal{X}})^{-1}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$	$\mathrm{Tr}\left({\mathbf{H}^{(l)}}^\top(\mathbf{I}+\alpha\widetilde{\mathbf{L}}_{\mathcal{G}}+\beta\widetilde{\mathbf{L}}_{\mathcal{X}})\mathbf{H}^{(l)}\right)$	$\begin{cases}\mathcal{S}^{(l)} = \mathcal{S}_+, l \in [L{-}1], \\ \mathcal{S}^{(L)} = \mathcal{S}_{simplex}\end{cases}$

Table 1: Different regularizers can derive different GCN variants under the regularizer-centered optimization framework.

where $\mathbf{\Theta}$ is regarded as the filter coefficients to be optimized. Especially, $\mathbf{\Theta}$ is assumed to be the polynomials of the spectrums of the graph Laplacian (Hammond, Vandergheynst, and Gribonval 2011), i.e.,

$$\mathbf{\Theta} = \mathbf{\Theta}(\mathbf{\Lambda}) = \sum_{i=1}^{K}\mathbf{\Theta}_i\mathbf{\Lambda}^i, \tag{3}$$

where K is the order of Chebyshev polynomials. By fixing $K = 2$, the graph convolutional network (GCN) (Kipf and Welling 2017) takes an effective form

$$\mathbf{g} * \mathbf{x} = \theta(\mathbf{I} + \mathbf{L})\mathbf{x}, \tag{4}$$

where $\mathbf{\Theta} = [\theta]$ is a parameter to be optimized. When extending single channel signal $\mathbf{x}$ and filter θ to multi-channel $\mathbf{H}^{(l)} \in \mathbb{R}^{N\times d_l}$ and $\mathbf{\Theta}^{(l)} \in \mathbb{R}^{d_l\times f_l}$, the GCN is converted to

$$\mathbf{H}^{(l)} = \sigma(\widehat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}), \tag{5}$$

where $\widehat{\mathbf{A}}$ is a normalized version of $\mathbf{I} + \widetilde{\mathbf{A}}$, $\sigma(\cdot)$ is an activation function, and $\mathbf{H}^{(l)} \in \mathbb{R}^{N\times d_l}$ is the output of the l-th layer with $\mathbf{H}^{(0)} = \mathbf{X}$ being the input feature matrix.

An Interpretable Regularizer-centered Optimization Framework for GCNs

Given the input $\mathbf{H}^{(l-1)}$ of the (l)-th layer, GCN can compute the output $\mathbf{H}^{(l)}$ by minimizing

$$\mathcal{L} = -\mathrm{Tr}({\mathbf{H}^{(l)}}^\top\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}) + \frac{1}{2}\mathrm{Tr}({\mathbf{H}^{(l)}}^\top\widetilde{\mathbf{L}}\mathbf{H}^{(l)}) \tag{6}$$
$$\text{s.t. } \mathbf{H}^{(l)} \geq \mathbf{0},$$

where $\frac{1}{2}\mathrm{Tr}({\mathbf{H}^{(l)}}^\top\widetilde{\mathbf{L}}\mathbf{H}^{(l)}) = \frac{1}{4}\sum_{j=1}^N\sum_{i=1}^N \mathbf{A}_{ij}||\frac{\mathbf{h}_i^{(l)}}{\sqrt{\mathbf{D}_{ii}}} - \frac{\mathbf{h}_j^{(l)}}{\sqrt{\mathbf{D}_{jj}}}||_2^2$ with $\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)};\cdots;\mathbf{h}_N^{(l)}]$; it is a normalized regularizer to preserve the pairwise similarity of any two nodes in the given graph. Besides, the $-\mathrm{Tr}({\mathbf{H}^{(l)}}^\top\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)})$ is actually a fitting loss term bewteen $\mathbf{H}^{(l)}$ and $\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}$, i.e., $||\mathbf{H}^{(l)} - \mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}||_F^2$ with $\mathbf{H}^{(l-1)}$ and $\mathbf{\Theta}^{(l)}$ fixed when optimizing $\mathbf{H}^{(l)}$. Note that the square term $||\mathbf{H}^{(l)}||_F^2$ is a L2-regularized smoother, which can be ignored or absorbed in the second graph regularizer $\mathrm{Tr}({\mathbf{H}^{(l)}}^\top\widetilde{\mathbf{L}}\mathbf{H}^{(l)})$.

Taking derivative of $\mathcal{L}$ with respect to $\mathbf{H}^{(l)}$ and setting it to zero, we obtain $\mathbf{H}^{(l_+)}$ as

$$\mathbf{H}^{(l_+)} = (\mathbf{I} - \widetilde{\mathbf{A}})^{-1}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}; \tag{7}$$

and then there yields

$$\mathbf{H}^{(l)} = \sigma\left(\mathbf{H}^{(l_+)}\right), \tag{8}$$

when the nonnegative constratints $\mathbf{H}^{(l)} \geq \mathbf{0}$ are further considered. Notice that $\sigma(\cdot)$ is the ReLU$(\cdot)$ activation function. Here, if the matix inverse $(\mathbf{I} - \tilde{\mathbf{A}})^{-1} = \sum_{i=0}^{\infty}\tilde{\mathbf{A}}^i$ is approximated by the first-order expansion, i.e., $(\mathbf{I} - \tilde{\mathbf{A}})^{-1} \approx \mathbf{I} + \tilde{\mathbf{A}}$, then Eq. (8) will lead to the updating rule (5) of GCN.

Usually, the activation functions in GCN are ReLU$(\cdot)$ and Softmax$(\cdot)$, which could be converted to different projection optimizations. Concretely, the $\mathrm{ReLU}(\cdot)$ activation function is equivalent to project a point $\mathbf{x}$ onto the non-negative plane $\mathcal{S}_+ = \{\mathbf{s} \in \mathbb{R}^d | \mathbf{s} \geq \mathbf{0}\}$, i.e.,

$$\mathrm{ReLU}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathcal{S}_+} -\mathbf{x}^\top\mathbf{y} + \frac{1}{2}||\mathbf{y}||_2^2. \tag{9}$$

By the way, we denote $\mathcal{S} = \{\mathbf{s} \in \mathbb{R}^d\}$, which corresponds to an identity activation function. In terms of the Softmax$(\cdot)$ activation function, it can be regarded as projecting $\mathbf{x}$ onto the set $\mathcal{S}_{simplex} = \{\mathbf{s} \in \mathbb{R}^d | \mathbf{1}^\top\mathbf{s} = 1, \mathbf{s} \geq \mathbf{0}\}$, i.e.,

$$\mathrm{Softmax}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathcal{S}_{simplex}} -\mathbf{x}^\top\mathbf{y} + \mathbf{y}^\top\log(\mathbf{y}), \tag{10}$$

where $\mathbf{y}^\top\log(\mathbf{y}) = \sum_{i=1}^d \mathbf{y}_i\log(\mathbf{y}_i)$ is the negative entropy of $\mathbf{y}$ (Amos 2019). In fact, with respect to other activation functions, they can also be equivalent to project a point onto some feasible set with some metric.

Datasets	#Node	#Edge	#Class	#Feature
Cora	2,708	5,429	7	1,433
Citeseer	3,327	4,732	6	3,703
Pubmed	19,717	44,338	3	500
ACM	3,025	13,128	3	1,870
BlogCatalog	5,196	171,743	6	8,189
CoraFull	19,793	65,311	70	8,710
Flickr	7,575	239,738	9	12,047
UAI	3,067	28,311	19	4,973

Table 2: Dataset statistics.

Up to present, we have actually utilized a constrained optimization problem to interpret GCN, including information propagations (i.e., Eq. (7)) and the nonlinear activation functions (i.e., ReLU($\cdot$) and Softmax($\cdot$)).

The above analyses can not only explain the vanilla GCN, but also stimulate a regularizer-centered optimization framework that can further unify various GCNs. By extending the optimization (6), a more general framework is written as

$$\mathcal{L} = -\mathrm{Tr}(\mathbf{H}^{(l)\top}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}) + \frac{1}{2}\mathcal{L}(\mathbf{H}^{(l)};\mathcal{G}) \tag{11}$$

$$\text{s.t. } \mathbf{H}^{(l)} \in \{\mathcal{S}_+ \ or\ \mathcal{S}\}, l \in [L-1], \mathbf{H}^{(L)} \in \mathcal{S}_{simplex}.$$

Under this framework, different regularizers could derive different GCNs, for example,

- If $\mathcal{L}(\mathbf{H}^{(l)};\mathcal{G}) = \mathrm{Tr}\left(\mathbf{H}^{(l)\top}(\mathbf{I}+\mu\widehat{\mathbf{L}})^{-1}(\mathbf{I}+\lambda\widehat{\mathbf{L}})\mathbf{H}^{(l)}\right)$ with $\lambda = \beta + \frac{1}{\alpha} - 1$, $\mu = \beta$, and $\widehat{\mathbf{L}} = \mathbf{I} - \widehat{\mathbf{A}}$, then it induces the updating rule $\mathbf{H}^{(l)} = \sigma\left((\mathbf{I}+\alpha\widehat{\mathbf{A}})^{-1}(\mathbf{I}+\beta\widehat{\mathbf{A}})\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$, which corresponds to GNN-HF (Zhu et al. 2021).
- If $\mathcal{L}(\mathbf{H}^{(l)};\mathcal{G}) = \mathrm{Tr}\left(\mathbf{H}^{(l)\top}(\mathbf{I}+\mu\widehat{\mathbf{A}})^{-1}(\mathbf{I}+\lambda\widehat{\mathbf{A}})\mathbf{H}^{(l)}\right)$ with $\lambda = \frac{-\alpha\beta+2\alpha-1}{\alpha\beta-\alpha+1}$ and $\mu = \frac{1}{\beta} - 1$, then it gives rise to the updating rule $\mathbf{H}^{(l)} = \sigma\left((\mathbf{I}+\alpha\widehat{\mathbf{A}})^{-1}(\mathbf{I}+\beta\widehat{\mathbf{A}})\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right)$, which corresponds to GNN-LF (Zhu et al. 2021).

For more cases, their results are summarized in Table 1, and the derivation details can refer to those of the original GCN (from Eq. (7) to Eq. (10)) and the supplementary.

Remarks. The work (Zhu et al. 2021) is most similar to our work with the same research idea: they both want to propose a unified framework to interpret the current GCNs and guide the design of new GCN variants; however, they are realized in different ways. To be specific, (1) Zhu et al. (Zhu et al. 2021) develop an optimization framework to explain different GCNs' propagation processes; whereas we propose a constrained optimization framework not only to interpret various GCNs' propagation processes, but also explain the nonlinear activation layers; (2) (Zhu et al. 2021) unifies various GCNs via devising various fitting items; while our work derives different GCNs through designing different regularizers. To sum up, our work interprets the whole (not partial) GCNs with regularizer-centered constrained optimizations.

Algorithm 1: Topological and Semantic Regularized GCN

```
Require: Graph data $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, labels $\mathbf{y}$, number of
    layers $L$, and hyperparameters $\{\alpha, \beta, r\}$.
Ensure: Predicted label set $\{y_i^*\}_{i=n+1}^N$.
1: Initialize model parameters $\{\mathbf{H}^{(l)}, \mathbf{\Theta}^{(l)}\}_{l=1}^L$;
2: Compute the joint graph Laplacian $\alpha\widetilde{\mathbf{L}}_\mathcal{G} + \beta\widetilde{\mathbf{L}}_\mathcal{X}$ and its
   low-rank factorization $\mathbf{W}\mathbf{V}^\top$;
3: Substitute the matrix inverse $(\mathbf{I} + \alpha\widetilde{\mathbf{L}}_\mathcal{G} + \beta\widetilde{\mathbf{L}}_\mathcal{X})^{-1}$ with
   $\mathbf{I} - \mathbf{W}(\mathbf{I} + \mathbf{V}^\top\mathbf{W})^{-1}\mathbf{V}^\top$;
4: while not convergent do
5:     Calculate hidden layers $\{\mathbf{H}^{(l)}\}_{l=1}^L$ by Eq. (14);
6:     Update weights: $\mathbf{\Theta}^{(l+1)} \leftarrow \mathbf{\Theta}^{(l)} - \eta\frac{\partial\mathcal{L}}{\partial\mathbf{\Theta}^{(l)}}$;
7: end while
8: return The predicted labels: $y_i^* = \arg\max_j \mathbf{H}_{ij}^{(L)}$.
```

tsGCN: Topological and Semantic Regularized Graph Convolutional Network

One finding from most existing GCNs is that they often ignored feature-based semantic structures, which can weaken the representation learning abilities of graph networks, then we focus on two regularizers, i.e.,

$$\mathcal{L}_1(\mathbf{H}^{(l)};\mathcal{G}) = \frac{1}{2}\mathrm{Tr}\left(\{\mathbf{H}^{(l)}\}^\top(\frac{1}{2}\mathbf{I} + \alpha\widetilde{\mathbf{L}}_\mathcal{G})\mathbf{H}^{(l)}\right), \tag{12}$$

$$\mathcal{L}_2(\mathbf{H}^{(l)};\mathcal{X}) = \frac{1}{2}\mathrm{Tr}\left(\{\mathbf{H}^{(l)}\}^\top(\frac{1}{2}\mathbf{I} + \beta\widetilde{\mathbf{L}}_\mathcal{X})\mathbf{H}^{(l)}\right), \tag{13}$$

where $\widetilde{\mathbf{L}}_\mathcal{G}$ is a graph Laplacian from the given adjacency matrix (e.g., $\widetilde{\mathbf{L}}_\mathcal{G} = \widetilde{\mathbf{L}}$), and $\widetilde{\mathbf{L}}_\mathcal{X}$ is a graph Laplacian calculated from the pairwise similarity of any two graph nodes. Hence, we devise a dual-regularizer, i.e., $\mathcal{L}(\mathbf{H}^{(l)}) = \mathcal{L}_1(\mathbf{H}^{(l)};\mathcal{G}) + \mathcal{L}_2(\mathbf{H}^{(l)};\mathcal{X})$, and if it is under the optimization framework (11), then there yields the following updating rule

$$\mathbf{H}^{(l)} = \sigma\left((\mathbf{I} + \alpha\widetilde{\mathbf{L}}_\mathcal{G} + \beta\widetilde{\mathbf{L}}_\mathcal{X})^{-1}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}\right). \tag{14}$$

Since this method seeks to preserve both the topological and semantic structures for more accurate presentations, we call it tsGCN (i.e., **T**opological and **S**emantic regularized **GCN**).

Notably, the computational complexity of $(\mathbf{I} + \alpha\widetilde{\mathbf{L}}_\mathcal{G} + \beta\widetilde{\mathbf{L}}_\mathcal{X})^{-1}$ is $\mathcal{O}(N^3)$, which tends to be unaffordable in practical applications. To this end, a low-rank approximation is operated, i.e., $\alpha\widetilde{\mathbf{L}}_\mathcal{G} + \beta\widetilde{\mathbf{L}}_\mathcal{X} \approx \mathbf{W}\mathbf{V}^\top$, where $\mathbf{W}, \mathbf{V} \in \mathbb{R}^{N\times r}$ with $r \ll N$. This leads to the Woodbury matrix identity:

$$(\mathbf{I} + \mathbf{W}\mathbf{V}^\top)^{-1} = \mathbf{I} - \mathbf{W}(\mathbf{I} + \mathbf{V}^\top\mathbf{W})^{-1}\mathbf{V}^\top, \tag{15}$$

of which the computational complexity costs $\mathcal{O}(N^2)$.

Given that the optimal $\mathbf{M}^*$ of the following problem

$$\min_{\mathbf{M}\in\mathbb{R}^{N\times N}:\ \mathrm{rank}(\mathbf{M})=r} ||\mathbf{M} - (\alpha\widetilde{\mathbf{L}}_\mathcal{G} + \beta\widetilde{\mathbf{L}}_\mathcal{X})||_\mathrm{F}^2 \tag{16}$$

is attained at the r-truncated singular value decomposition of $\alpha\widetilde{\mathbf{L}}_\mathcal{G} + \beta\widetilde{\mathbf{L}}_\mathcal{X}$, i.e., $\mathbf{M}^* = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$, where $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$ is a diagonal matrix containing the r largest singular values. An

Metrics	Methods / Datasets	Cora	Citeseer	Pubmed	ACM	BlogCatalog	CoraFull	Flickr	UAI
ACC	Chebyshev	76.2 (0.7)	69.3 (0.4)	74.0 (0.8)	82.8 (1.4)	68.3 (1.6)	57.2 (1.1)	38.5 (1.6)	49.7 (0.4)
	GraphSAGE	76.7 (0.6)	64.4 (0.9)	75.5 (0.2)	—	57.8 (0.7)	59.9 (0.7)	32.7 (1.0)	41.7 (1.4)
	GAT	79.1 (0.8)	68.3 (0.5)	78.4 (0.3)	84.6 (0.5)	67.1 (1.7)	62.4 (0.4)	40.4 (0.9)	49.7 (3.0)
	GCN	80.6 (1.4)	69.1 (1.5)	77.6 (1.3)	88.8 (0.5)	84.2 (0.6)	62.8 (0.4)	51.0 (1.2)	58.5 (1.1)
	SGC	79.3 (1.0)	66.4 (1.7)	76.8 (2.0)	80.8 (2.7)	81.3 (0.2)	62.9 (2.2)	51.0 (0.1)	56.5 (3.5)
	APPNP	78.0 (0.1)	65.8 (0.2)	78.0 (0.0)	88.2 (0.0)	87.7 (0.3)	63.1 (0.5)	57.5 (0.2)	62.3 (1.2)
	JKNet	**83.1 (0.1)**	72.3 (0.1)	80.1 (0.2)	82.3 (0.6)	75.7 (0.1)	62.6 (0.0)	54.0 (0.3)	45.6 (0.5)
	DAGNN	81.9 (0.7)	70.0 (1.1)	80.6 (0.7)	87.4 (0.9)	84.6 (1.9)	65.6 (0.3)	54.6 (5.9)	46.7 (12.4)
	GNN-LF	81.1 (0.5)	72.3 (0.9)	80.0 (0.4)	90.8 (0.5)	86.7 (0.6)	63.5 (0.9)	56.6 (0.6)	36.6 (19.8)
	GNN-HF	80.7 (0.2)	68.8 (1.3)	77.7 (0.2)	91.2 (0.5)	84.5 (0.4)	63.0 (0.7)	60.7 (0.4)	54.8 (1.4)
	tsGCN (inv)	80.3 (0.3)	**73.3 (0.4)**	78.4 (0.3)	85.1 (1.6)	87.8 (6.3)	67.0 (0.9)	53.3 (12.6)	64.2 (1.8)
	tsGCN	82.0 (0.3)	73.1 (0.4)	**82.4 (0.1)**	**92.8 (0.3)**	**92.3 (0.5)**	**67.9 (0.9)**	**79.1 (3.0)**	**67.9 (0.6)**
F1	Chebyshev	76.3 (0.7)	65.4 (0.8)	73.9 (0.7)	82.5 (1.4)	64.3 (1.6)	40.0 (0.5)	38.4 (1.5)	39.1 (0.2)
	GraphSAGE	76.7 (0.5)	60.7 (0.5)	74.7 (0.2)	—	54.7 (0.6)	51.9 (0.6)	31.0 (1.1)	35.3 (1.0)
	GAT	77.1 (0.7)	64.6 (0.5)	78.2 (0.2)	84.8 (0.5)	66.3 (1.9)	46.4 (0.4)	38.1 (1.1)	40.8 (1.3)
	GCN	79.4 (1.4)	65.2 (2.4)	77.2 (1.4)	88.9 (0.5)	82.4 (0.5)	52.8 (0.8)	50.0 (1.7)	45.0 (1.1)
	SGC	77.7 (0.9)	61.5 (1.7)	76.5 (2.3)	81.1 (2.6)	80.7 (0.3)	53.2 (2.1)	44.2 (0.2)	46.7 (1.7)
	APPNP	77.6 (0.1)	63.2 (0.2)	77.7 (0.0)	88.3 (0.0)	85.7 (0.3)	48.2 (0.7)	56.9 (0.2)	48.6 (1.6)
	JKNet	**82.3 (0.3)**	67.8 (0.1)	79.3 (0.3)	82.2 (0.6)	75.0 (0.1)	51.3 (0.1)	51.1 (0.5)	31.7 (1.5)
	DAGNN	80.0 (0.7)	65.7 (0.7)	80.7 (0.7)	87.5 (0.9)	83.8 (2.4)	53.0 (0.9)	55.5 (6.7)	39.3 (11.2)
	GNN-LF	79.1 (0.7)	66.7 (0.4)	80.2 (0.5)	90.9 (0.5)	85.9 (0.6)	50.5 (1.9)	54.3 (1.0)	29.7 (15.1)
	GNN-HF	78.6 (0.3)	64.3 (1.7)	78.1 (0.2)	91.3 (0.5)	83.8 (0.4)	49.0 (1.1)	58.6 (0.6)	44.9 (0.8)
	tsGCN (inv)	78.5 (0.3)	**69.6 (0.4)**	78.7 (0.3)	85.1 (1.5)	85.2 (7.1)	57.2 (1.1)	52.9 (15.8)	48.5 (0.8)
	tsGCN	80.5 (0.5)	69.0 (0.3)	**82.4 (0.1)**	**92.8 (0.4)**	**90.1 (0.6)**	**58.7 (0.7)**	**79.3 (2.9)**	**50.1 (0.1)**

Table 3: Accuracies and F1-scores (mean% and standard deviation%) of all methods, where the best results are in bold and the second-best are underlined. Note that GraphSAGE fails to work on the ACM dataset, and thus its results are marked with "—".

optimal $\{\mathbf{W}^*, \mathbf{V}^*\}$ to $\alpha\widetilde{\mathbf{L}}_{\mathcal{G}} + \beta\widetilde{\mathbf{L}}_{\mathcal{X}} \approx \mathbf{W}\mathbf{V}^\top$ can be given by an analytic form of $\mathbf{W}^* = \mathbf{V}^* = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$.

To obtain the optimum $\{\mathbf{W}^*, \mathbf{V}^*\}$, the iterative algorithm (Sun and Xu 2019) with $\mathcal{O}(N^2)$ can be leveraged as

$$\mathbf{Z}^{(t+1)} \leftarrow (\alpha\widetilde{\mathbf{L}}_{\mathcal{G}} + \beta\widetilde{\mathbf{L}}_{\mathcal{X}})\mathbf{U}^{(t)}, \tag{17}$$

$$\{\mathbf{U}^{(t+1)}, \mathbf{R}^{(t+1)}\} \leftarrow \text{QR}(\mathbf{Z}^{(t+1)}), \tag{18}$$

where QR$(\cdot)$ denotes the QR-decomposition. Note that this algorithm can converge to the r largest eigenvalues $\mathbf{R}^{(t+1)}$ and its corresponding eigenvectors $\mathbf{Z}^{(t+1)}$ when t is large enough, that is $\mathbf{W}^* = \mathbf{V}^* = \mathbf{U}^{(t+1)}[\mathbf{R}^{(t+1)}]^{\frac{1}{2}}$.

Gathering all analyses mentioned above, the procedure for tsGCN is summarized in Algorithm 1.

Experiment

Datasets

Cora, Citeseer and Pubmed are citation networks, and CoraFull is a larger version of Cora; ACM is a paper network, and BlogCatalog and Flickr are social networks; UAI has been utilized for community detection. The detailed statistics of the above eight public datasets are concluded in Table 2.

Compared Methods

Two types of methods are employed here for comparisons. Chebyshev (Defferrard, Bresson, and Vandergheynst 2016), GraphSAGE (Hamilton, Ying, and Leskovec 2017) and GAT (Velickovic et al. 2018) are classical graph neural networks. GCN, SGC (Wu et al. 2019), APPNP (Klicpera, Bojchevski, and Günnemann 2019), JKNet (Xu et al. 2018), DAGNN (Liu, Gao, and Ji 2020), GNN-LF and GNN-HF (Zhu et al. 2021) are selected as state-of-the-art GCN variants.

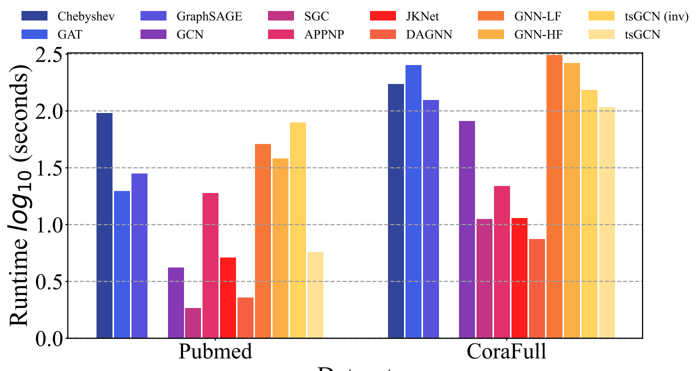
Parameter Setups

For all experiments, we randomly split samples into a small set of 20 labeled samples per class for training, a set of 500 samples for validating and a set of $1,000$ samples for testing. In terms of the ten baseline methods, all their configurations are set as the default in their original papers. With respect to tsGCN, the learning rate, weight decay and the size of hidden units are set to 1×10^{-2}, 5×10^{-4} and 32, respectively. The hyperparameters α and β are selected in $\{0.1, 0.2, \ldots, 1.0\}$ for different datasets, and r is chosen in $\{\lfloor\frac{N}{2^{11}}\rfloor, \lfloor\frac{N}{2^{10}}\rfloor, \ldots, \lfloor\frac{N}{2^{3}}\rfloor\}$, where N is the number of nodes.

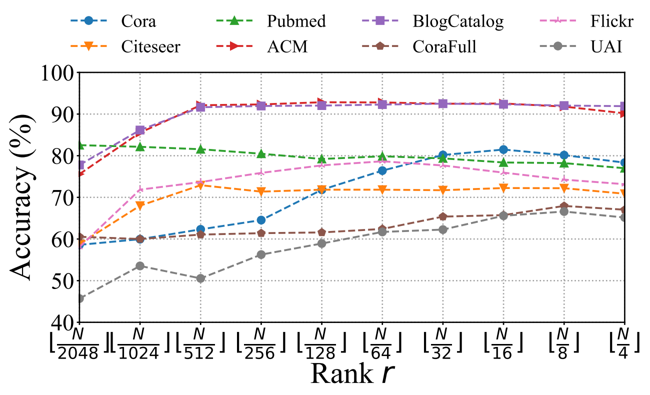
Semi-supervised Classification

Performance Comparisons. The semi-supervised classification task is conducted on selected datasets, whose results are recorded in Table 3. Specifically, we compare our tsGCN with the ten baseline methods in terms of both accuracy and F1-score, marking the best and second-best results on each dataset. Note that tsGCN (inv) denotes the variant that directly calculates the matrix inverse in Eq. (14). From Table 3, we have the following observations:

- tsGCN achieves the best performance on most datasets, and is only slightly inferior to the JKNet method on the smallest Cora dataset.
- tsGCN yields higher scores than JKNet and APPNP, especially on Pubmed, CoraFull, BlogCatalog, and Flickr,

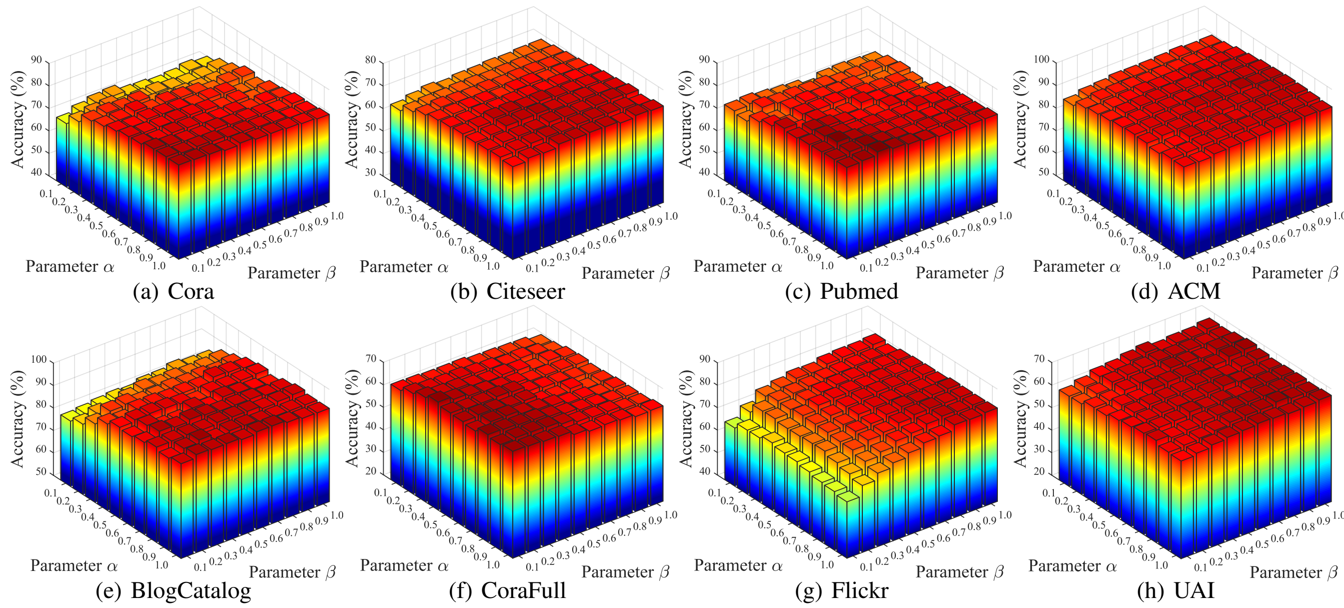


(a) Runtime



(b) Classification Accuracies

Figure 1: (a) All methods' runtime on two large datasets. (b) The classification accuracies of tsGCN w.r.t. (α, β) on all datasets.



(a) Cora (b) Citeseer (c) Pubmed (d) ACM

(e) BlogCatalog (f) CoraFull (g) Flickr (h) UAI

Figure 2: The classification accuracies of tsGCN w.r.t. hyperparameters α and β on all datasets.

where the first two are relatively large datasets and the latter two have dense edges. tsGCN even outperforms the second-best approach GNN-HF by about 20% on Flickr.

It is worth mentioning that tsGCN utilizes global information by the infinite-order graph convolution, and JKNet and APPNP also develop different techniques for the same goal. Hence, the advantage of tsGCN over APPNP and JKNet implies that the infinite-order graph convolution implemented by low-rank approximations not only requires less computational complexity, but also effectively captures high-order neighborhood information and filters significant noises.

Runtime Comparisons. This section collects the training time (i.e., runtime) of all methods on two large datasets, i.e., Pubmed and CoraFull, as exhibited in Fig. 1(a): the first three columns correspond to classical GNNs, while the rest are GCNs. Fig. 1(a) shows that tsGCN takes much less runtime than Chebyshev, GAT, and GraphSAGE but performs moderately well among the state-of-the-art GCNs. Specifically, tsGCN is (1) inferior to SGC, JKNet, and DAGNN; (2) well-matched with the original GCN; (3) but advantageous over the recently proposed GNN-LF and GNN-HF.

Parameter Sensitivity Analysis

Fig. 1(b) curves the accuracies of tsGCN w.r.t. various ranks by fixing other parameters α and β. Considering that different datasets hold different distributions, their optimal ranks to the optimization (16) are also different. For example, in regard to the curves on BlogCatalog and ACM, their accuracies first go up to a high value and then keep steady, which indicates that when rank $r = \lfloor N/512 \rfloor$, the low-rank approximation is effective and efficient enough. When it comes to the curve on Pubmed, the trend of its performance monotonically decreases as rank r becomes bigger, which implies

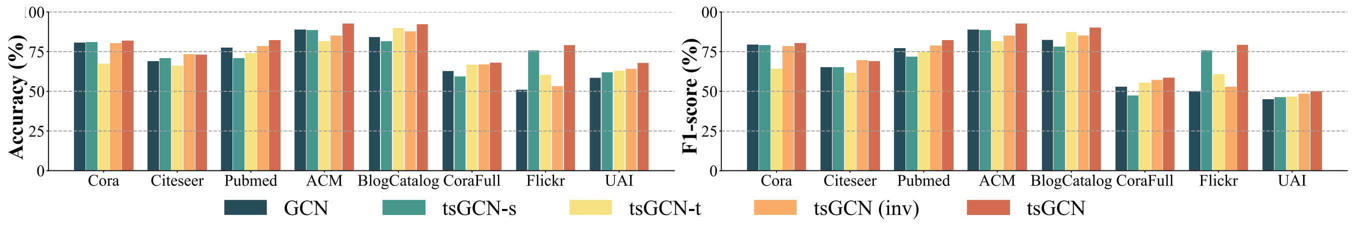


Figure 3: Accuracies and F1-scores of tsGCN and its variants on all datasets. Note that tsGCN-s and tsGCN-t are with only semantic and topological regularizer, respectively.

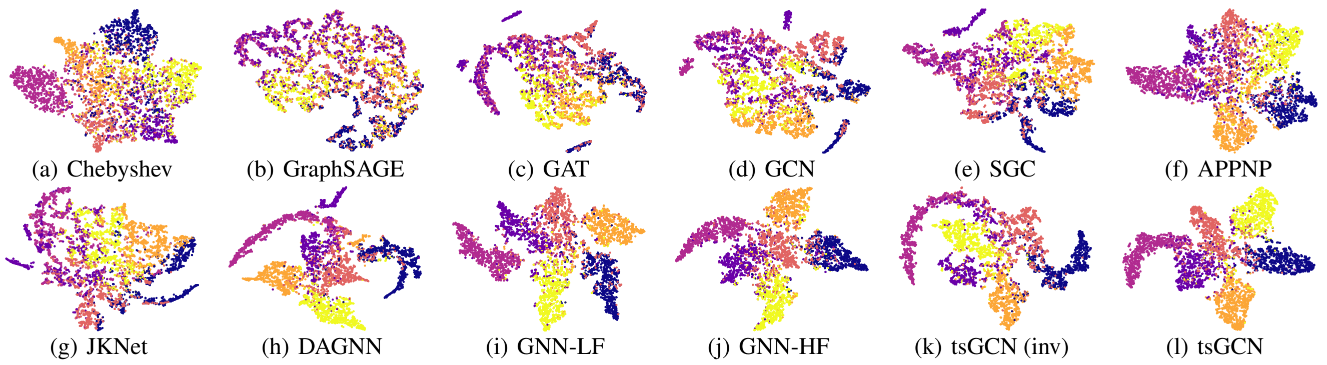


Figure 4: Different methods' t-SNE visualizations on BlogCatalog, where each color corresponds to one class.

that a very low-rank (i.e., $r = \lfloor N/2048 \rfloor$) approximation is sufficient enough to preserve abundant information for good results. However, with respect to the other curves such as on Flickr and Cora, the y-axis' scores generally rise to a peak first and then fall continuously as the rank r increases. For these cases, the optimal ranks differ at their peaks.

Fig. 2 plots the accuracies of tsGCN w.r.t. (α, β) by fixing the optimal ranks. On Cora, Pubmed, BlogCatalog, and CoraFull, tsGCN performs well with large α and small β; while, on Citeseer, ACM, Flickr, and UAI, tsGCN generates high accuracy when these two parameters are both large.

Ablation Study

The results of GCN, tsGCN-s, tsGCN-t, tsGCN (inv), and tsGCN are plotted in Fig. 3, telling us:

- The performance is unsatisfactory when the two regularizers are adopted alone, while tsGCN can always effectively fuse the two to better capture underlying structures.
- tsGCN (inv) is even worse than single-regularizer model on some datasets, indicating that the infinite-order graph convolutions implemented by the matrix inverse can pull-in instability to the model.
- Compared to GCN, tsGCN (inv) performs worse whereas tsGCN shows substantial improvements on all datasets, indicating that the low-rank approximation enhances the robustness of infinite-order graph convolutions.

Data Visualization

Fig. 4 exhibits the graph representations learned by different methods on BlogCatalog. As can be seen clearly, the results of the three classical graph neural networks, i.e., Chebyshev, GraphSAGE and GAT, are unsatisfactory; while for the other competitors, there are:

- Both tsGCN (inv) and tsGCN are better than other GCNs, which indicates that the dual-regularizer can extract more accurate inter-relationships from the topological and semantic structures.
- Comparing the embeddings learned by tsGCN and tsGCN (inv), classes are more clearly recognized and the within-clusters are more compact with tsGCN, which testifies the effectiveness of low-rank approximation.

In a nutshell, the embeddings of the proposed model show the best inter-class separation and intra-class aggregation.

Conclusion

By revisiting GCN, this paper puts forward an interpretable regularizer-centered optimization framework, in which the connections between existing GCNs and diverse regularizers are revealed. It is worth mentioning that this framework provides a new perspective to interpret existing work and guide new GCNs just by designing new graph regularizers. Impressed by the significant effectiveness of the feature based semantic graph, we further combine it with nodes' topological structures, and develop a novel dual-regularizer graph convolutional network, called tsGCN. Since the analytical updating rule of tsGCN contains a time-consuming matrix inverse, we devise an efficient algorithm with low-rank approximation tricks. Experiments on node classification tasks demonstrate that tsGCN performs much better than quite a few state-of-the-art competitors, and also exhibit that tsGCN runs much faster than the very recently proposed GCN variants, e.g., GNN-HF and GNN-LF.

Acknowledgments

This work is in part supported by National Natural Science Foundation of China (Grant No. U21A20472, 62276065, and 62006005), and Natural Science Foundation of Fujian Province (Grant No. 2020J01130193).

References

Amos, B. 2019. *Differentiable Optimization-Based Modeling for Machine Learning*. Ph.D. thesis, Carnegie Mellon University.

Bo, D.; Wang, X.; Shi, C.; and Shen, H. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. In *AAAI*, 3950–3957.

Chen, H.; Yin, H.; Sun, X.; Chen, T.; Gabrys, B.; and Musial, K. 2020a. Multi-level Graph Convolutional Networks for Cross-platform Anchor Link Prediction. In *KDD*, 1503–1511.

Chen, W.; Chen, L.; Xie, Y.; Cao, W.; Gao, Y.; and Feng, X. 2020b. Multi-Range Attentive Bicomponent Graph Convolutional Network for Traffic Forecasting. In *AAAI*, 3529–3536.

Chen, Y.; Huang, L.; Wang, C.; and Lai, J. 2022. Hybrid-Order Gated Graph Neural Network for Session-Based Recommendation. *IEEE Transactions on Industrial Informatics*, 18(3): 1458–1467.

Chen, Z.; Fu, L.; Yao, J.; Guo, W.; Plant, C.; and Wang, S. 2023. Learnable graph convolutional network and feature fusion for multi-view learning. *Information Fusion*, 95: 109–119.

Chen, Z.; Wei, X.; Wang, P.; and Guo, Y. 2019. Multi-Label Image Recognition With Graph Convolutional Networks. In *CVPR*, 5177–5186.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 1–9.

Fan, S.; Wang, X.; Shi, C.; Lu, E.; Lin, K.; and Wang, B. 2020. One2Multi Graph Autoencoder for Multi-view Graph Clustering. In *WWW*, 3070–3076.

Halliwell, N. 2022. Evaluating Explanations of Relational Graph Convolutional Network Link Predictions on Knowledge Graphs. In *AAAI*, 12880–12881.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 1024–1034.

Hammond, D. K.; Vandergheynst, P.; and Gribonval, R. 2011. Wavelets on Graphs via Spectral Graph Theory. *Applied and Computational Harmonic Analysis*, 30: 129–150.

Hu, H.; Cheng, L.; Vap, J. P.; and Borowczak, M. 2022. Learning Privacy-Preserving Graph Convolutional Network with Partially Observed Sensitive Attributes. In *WWW*, 3552–3561.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised Classification with Graph Convolutional Networks. In *ICLR*, 1–13.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*, 1–15.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*, 3538–3545.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards Deeper Graph Neural Networks. In *KDD*, 338–348.

Nie, W.; Zhao, Y.; Liu, A.; Gao, Z.; and Su, Y. 2020. Multi-graph Convolutional Network for Unsupervised 3D Shape Retrieval. In *MM*, 3395–3403.

Oono, K.; and Suzuki, T. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *ICLR*, 1–8.

Sun, J.; and Xu, Z. 2019. Neural Diffusion Distance for Image Segmentation. In *NeurIPS*, 1441–1451.

Sun, K.; Zhu, Z.; and Lin, Z. 2021. AdaGCN: Adaboosting Graph Convolutional Networks into Deep Models. In *ICLR*, 1–15.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*, 1–12.

Wang, X.; Zhu, M.; Bo, D.; Cui, P.; Shi, C.; and Pei, J. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *KDD*, 1243–1253.

Wang, Y.; Cao, M.; Fan, Z.; and Peng, S. 2022. Learning to Detect 3D Facial Landmarks via Heatmap Regression with Graph Convolutional Network. In *AAAI*, 2595–2603.

Wu, F.; Jr., A. H. S.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. In *ICML*, 6861–6871.

Wu, Z.; Shu, L.; Xu, Z.; Chang, Y.; Chen, C.; and Zheng, Z. 2022. Robust Tensor Graph Convolutional Networks via T-SVD Based Graph Augmentation. In *KDD*, 2090–2099.

Xu, F.; Lian, J.; Han, Z.; Li, Y.; Xu, Y.; and Xie, X. 2019. Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-Commerce Recommendation. In *CIKM*, 529–538.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; ichi Kawarabayashi, K.; and Jegelka, S. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*, 5449–5458.

Yang, L.; Kang, Z.; Cao, X.; Jin, D.; Yang, B.; and Guo, Y. 2019. Topology Optimization based Graph Convolutional Network. In *AAAI*, 4054–4061.

Yang, M.; Shen, Y.; Li, R.; Qi, H.; Zhang, Q.; and Yin, B. 2022. A New Perspective on the Effects of Spectrum in Graph Neural Networks. In *ICML*, 25261–25279.

Yang, Y.; Liu, T.; Wang, Y.; Zhou, J.; Gan, Q.; Wei, Z.; Zhang, Z.; Huang, Z.; and Wipf, D. 2021. Graph Neural Networks Inspired by Classical Iterative Algorithms. In *ICML*, 11773–11783.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *IJCAI*, 3634–3640.

Zhang, Y.; Pal, S.; Coates, M.; and Üstebay, D. 2019. Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification. In *AAAI*, 5829–5836.

Zhang, Z.; Chen, C.; Chang, Y.; Hu, W.; Xing, X.; Zhou, Y.; and Zheng, Z. 2021. SHNE: Semantics and Homophily Preserving Network Embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 1–12.

Zhao, L.; and Akoglu, L. 2020. Connecting Graph Convolutional Networks and Graph-Regularized PCA. *arXiv preprint arXiv:2006.12294*.

Zhu, H.; and Koniusz, P. 2021. Simple Spectral Graph Convolution. In *ICLR*, 1–11.

Zhu, M.; Wang, X.; Shi, C.; Ji, H.; and Cui, P. 2021. Interpreting and Unifying Graph Neural Networks with An Optimization Framework. In *WWW*, 1215–1226.