Why Do Attributes Propagate in Graph Convolutional Neural Networks?

Liang Yang[1,2,3], **Chuan Wang**[2*], **Junhua Gu**[1,3], **Xiaochun Cao**[2], **Bingxin Niu**[1,3]

[1]School of Artificial Intelligence, Hebei University of Technology, Tianjin, China
[2]State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
[3]Hebei Province Key Laboratory of Big Data Calculation, Hebei University of Technology, Tianjin, China
yangliang@vip.qq.com, {wangchuan, caoxiaochun}@iie.ac.cn, {jhgu_hebut, niubingxin666}@163.com

Abstract

Many efforts have been paid to enhance Graph Convolutional Network from the perspective of propagation under the philosophy that "Propagation is the essence of the GCNNs". Unfortunately, its adverse effect is over-smoothing, which makes the performance dramatically drop. To prevent the over-smoothing, many variants are presented. However, the perspective of propagation can't provide an intuitive and unified interpretation to their effect on prevent over-smoothing. In this paper, we aim at providing a novel explanation to the question of *"Why do attributes propagate in GCNNs?"*. which not only gives the essence of the oversmoothing, but also illustrates why the GCN extensions, including multi-scale GCN and GCN with initial residual, can improve the performance. To this end, an intuitive Graph Representation Learning (GRL) framework is presented. GRL simply constrains the node representation similar with the original attribute, and encourages the connected nodes possess similar representations (pairwise constraint). Based on the proposed GRL, exiting GCN and its extensions can be proved as different numerical optimization algorithms, such as gradient descent, of our proposed GRL framework. Inspired by the superiority of conjugate gradient descent compared to common gradient descent, a novel Graph Conjugate Convolutional (GCC) network is presented to approximate the solution to GRL with fast convergence. Specifically, GCC adopts the *obtained* information of the last layer, which can be represented as the difference between the input and output of the last layer, as the input to the next layer. Extensive experiments demonstrate the superior performance of GCC.

Introduction

Graph Neural Networks (GNNs) (Wu et al. 2020; Xu et al. 2019) have become a hot topic in deep learning for their potentials in modeling irregular data. GNNs have been widely used and achieved state-of-the-art performance in many fields, such as computer vision, natural language processing (Yang et al. 2020), traffic forecasting, chemistry and medical analysis, etc. Existing GNNs fall into two categories, spectral methods (Defferrard, Bresson, and Vandergheynst 2016) and spatial ones (Hamilton, Ying, and Leskovec 2017; Gilmer et al. 2017; Yang et al. 2019b,a; Jin et al. 2019, 2020, 2021).

*Corresponding author.

Graph Convolutional Network (GCN) (Kipf and Welling 2017), which is a simple, well-behaved and insightful GNN, bridges above two perspectives by proving that the propagation can be motivated from a first-order approximation of spectral graph convolutions. Recently progress also demonstrates the equivalent of spatial and spectral ones (Balcilar et al. 2020). Many efforts have been paid to enhance GCN from the perspective of propagation (Gilmer et al. 2017), such as learnable propagation weights in Graph Attention Network (GAT) (Velickovic et al. 2018), Gated Attention Network (GaAN) (Zhang et al. 2018) and Probabilistic GCN (Yang et al. 2020), structural neighbourhood in Geom-GCN (Pei et al. 2020) and multi-scale (multi-hop) combination in N-GCN (Abu-El-Haija et al. 2019a), MixHop (Abu-El-Haija et al. 2019b), LanczosNet (Liao et al. 2019) and Krylov GCN (Luan et al. 2019). The common philosophy of them is: *"Propagation is the essence of the GCNNs"*. And, the success of GCNs attributes to the Laplacian smoothing induced by the propagation (Li, Han, and Wu 2018).

Unfortunately, the most serious issue of GNNs is the over-smoothing, which makes the performance dramatically drop, caused by the multiple propagations via stacking multiple graph convolution layers. Recently, (Oono and Suzuki 2020) shows the the exponential loss of expressive power of GNNs by generalizing the forward propagation of a GCN as a specific dynamical system. To prevent over-smoothing, two kinds of methods are proposed. On one hand, methods in the first category constrain the propagation. Disentangled GCN (Ma et al. 2019) makes each attribute only be propagated on part of the edges. DropEdge (Rong et al. 2020) randomly removes a certain number of edges from the input graph at each training epoch to reduce the adverse effect of message passing. On the other hand, methods in the second category constrain the propagation result with the original attributes. PageRank-GCN (Klicpera, Bojchevski, and Günnemann 2019) integrates personalized PageRank to GCN to combine the original attribute. JKNet (Xu et al. 2018) employs dense connections for multi-hop message passing, while DeepGCN (Li et al. 2019) and (GCNII) (Chen et al. 2020) incorporates residual layers into GCNs to facilitate the development of deep architectures. However, the perspective of propagation can't provide an intuitive and unified interpretation to their effect on preventing over-smoothing.

In this paper, we aim at providing a novel explanation to the question of

Why do attributes propagate in GCNNs?

which not only gives the essence of the oversmoothing, but also illustrates why the GCN extensions, including multi-scale GCN and GCN with initial residual, can improve the performance. To this end, an intuitive Graph Representation Learning (GRL) framework is presented by assuming the topology is accuracy. GRL simply constrains the node representation similar with the original attributes, and encourages the connected nodes possess similar representations (pairwise constraint). Then, by taking consideration the noisy topology, a Robust GRL is obtained by introducing a robust estimation to the pairwise (similarity) constraint. The solution to GRL can be obtained by solving a positive semidefinite linear system, while that to Robust GRL iteratively solves linear system and refines pairwise similarity.

According to the proposed GRL and Robust GRL, exiting GCN and its extensions can be proved as different numerical optimization algorithms, such as gradient descent, of our proposed Graph Representation Learning framework. Specifically, GCN with initial residual can be seen as the gradient descent solution of GRL, multi-scale GCN as the high-order approximation to the analytic solution of GRL. Furthermore, the attention mechanism adopted in GCNNs can be regarded as the gradient descent solution to Robust GRL. Therefore, it provides insights of GCN and its variants from the perspective of numerical optimization that:

The propagation as well as its weight learning are not the essence of the GCNs, but induced by the numerical optimization of pairwise similarity requirement.

Based on this finding, a novel Graph Conjugate Convolutional (GCC) network is presented to approximate the solution to GRL with fast convergence. Specifically, inspired by the superiority of conjugate gradient descent compared to common gradient descent, the GCC building block, i.e., Graph Conjugate Convolutional layer, adopts the *obtained* information of the last layer, which can be represented as the difference between the input and output of the last layer, as the input to the next layer. It can significantly alleviates the overfitting issue caused by the accumulation effect of residual connection in GCN.

The main contribution of this paper are summarized as follows:

- We introduce a novel Graph Representation Learning (GRL) framework and its extension, Robust GRL, to take consideration of noisy topology.
- We explain the GCN and its variants from the perspective of the numerical optimization of pairwise similarity requirement, under GRL framework.
- We propose the Graph Conjugate Convolutional (GCC) network, which essentially sublimes the propagation.
- We experimentally verify the superiority of GCC on trasductive and inductive tasks.

Preliminaries

In this section, the notations are given. Then, gradient descent for linear system, which will be used to explain some existing graph neural networks (GNNs), is provided. Finally, some classic and recently proposed GNNs are reviewed.

Notations

A network can be represented by an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. $\mathcal{V} = \{v_i | i = 1, ..., N\}$ is a set of $|\mathcal{V}| = N$ vertices, where v_n is associated with a feature $\mathbf{x}_n \in \mathbb{R}^K$. $\mathbf{X} \in \mathbb{R}^{K \times N}$ is the collection of the features, each column of which corresponds to one node. $\mathcal{E}$ stands for a set of edges, each of which connects two vertices in $\mathcal{V}$. The adjacency matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$ is employed to represent the network topology, where $a_{ij} = 1$ if an edge exists between the vertices v_i and v_j, and vice versa. If the network is allowed to contain self-edges, then $a_{nn} = 1$, otherwise $a_{nn} = 0$. $\mathbf{a}_n$, which denotes the n^{th} column of $\mathbf{A}$, can be utilized to represent the neighbourhood of vertex v_n. $d_n = \sum_j a_{nj}$ is the degree of vertex v_n, and $\mathbf{D} = \mathrm{diag}(d_1, d_2, ..., d_N)$ is the degree matrix of the adjacency matrix $\mathbf{A}$. The graph Laplacian and its normalized form are defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$, respectively.

Gradient Descent for Linear System

Symmetric positive definite linear systems

$$\mathbf{A}\mathbf{u} = \mathbf{x}, \tag{1}$$

where $\mathbf{x}, \mathbf{u} \in \mathbb{R}^N$ are N-dimensional vectors and $\mathbf{A} \in \mathbb{R}^{N \times N}$ are symmetric ($\mathbf{A} = \mathbf{A}^T$) and positive definite ($\mathbf{A} \succ 0$), are widely used in science and engineering, such as regularized least-squares and elliptic PDE. Note that, different from the notation in linear algebra, $\mathbf{u}$ is unknown while $\mathbf{A}$ and $\mathbf{x}$ are given in this paper in order to be consistent with those used in graph neural networks. Matrix $\mathbf{A}$ is positive definite, i.e., $\mathbf{A} \succ 0$, if and only if $\mathbf{u}^T\mathbf{A}\mathbf{u} > 0$ for any non-zero vector $\mathbf{u}$. Unfortunately, its analytic solution $\mathbf{u}^* = \mathbf{A}^{-1}\mathbf{x}$ is computationally expensive, since it requires $\mathcal{O}(N^3)$ operations. Gradient descent methods for symmetric positive definite linear systems benefit from the structure of $\mathbf{A}$, such as sparse and low-rank characteristics. It minimizes the equivalent convex function

$$f(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{A}\mathbf{u} - \mathbf{u}^T\mathbf{x} \tag{2}$$

with the following iterative updating rule,

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} - \mu_t \nabla f(\mathbf{u}^{(t)}) = \mathbf{u}^{(t)} + \mu_t(\mathbf{x} - \mathbf{A}\mathbf{u}^{(t)}), \tag{3}$$

where μ_t denotes the step size in the t^{th} iteration. In practice, the conjugate gradient (CG) method is the most widely used iterative method to solve the symmetric positive definite linear systems (Golub and Van Loan 1996). CG augments the common gradient descent in Eq. (3) with the momentum term as

$$\begin{aligned}\mathbf{u}^{(t+1)} &= \mathbf{u}^{(t)} - \mu_t \nabla f(\mathbf{u}^{(t)}) + \kappa_t(\mathbf{u}^{(t)} - \mathbf{u}^{(t-1)}) \\ &= \mathbf{u}^{(t)} + \mu_t(\mathbf{x} - \mathbf{A}\mathbf{u}^{(t)}) + \kappa_t(\mathbf{u}^{(t)} - \mathbf{u}^{(t-1)}),\end{aligned} \tag{4}$$

which achieves faster convergence than common one.

Another approximation algorithm is to approximate the inverse of $\mathbf{A}$. By denoting the characteristic polynomial of the square matrix $\mathbf{A}$ as

$$\Delta(s) = |s\mathbf{I} - \mathbf{A}| = s^N + \alpha_1 s^{N-1} + \alpha_2 s^{N-N} + ... + \alpha_N,$$

where $\mathbf{I}$ is the identity matrix and $|\mathbf{A}|$ stands for the determinant of matrix $\mathbf{A}$. Cayley-Hamilton theorem (Decell 1965) states that every matrix satisfies its own characteristic equation, that is

$$\Delta(\mathbf{A}) = \mathbf{A}^N + \alpha_1 \mathbf{A}^{N-1} + \alpha_2 \mathbf{A}^{N-N} + ... + \alpha_N \mathbf{I} = 0.$$

Thus, the inverse of matrix $\mathbf{A}$ can be expressed as

$$\mathbf{A}^{-1} = -\frac{1}{\alpha_N}\mathbf{A}^{N-1} - \frac{\alpha_1}{\alpha_N}\mathbf{A}^{N-2} - ... - \frac{\alpha_{N-1}}{\alpha_N}\mathbf{I},$$

And $(\mathbf{I} - \mathbf{A})^{-1}$ can be expressed as

$$(\mathbf{I} - \mathbf{A})^{-1} = \beta_{N-1}\mathbf{A}^{N-1} + \beta_{N-2}\mathbf{A}^{N-2} + ... + \beta_0 \mathbf{I}, \tag{5}$$

Thus, the analytic solution $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ can also be obtained as

$$\mathbf{u}^* = -\sum_{k=0}^{N-1} \frac{\alpha_{N-k-1}}{\alpha_N}\mathbf{A}^k \mathbf{x},$$

by letting $\alpha_0 = 1$. Thus, $\mathbf{u}^*$ can be approximated with $P < N$ terms.

Graph Convolutional Neural Networks

Graph Convolutional Network (GCN) (Kipf and Welling 2017) simplifies previous Graph Convolutional Neural Networks (GCNNs) and motivate the convolutional architecture via a localized first-order approximation of spectral graph convolutions as

$$\mathbf{H}^{(t+1)} = \mathrm{ReLU}\left(\mathbf{W}^{(t+1)}\mathbf{H}^{(t)}\hat{\mathbf{A}}\right), \tag{6}$$

where $\mathbf{W}^{(l)}$ is a layer-specific trainable weight matrix. $\mathbf{H}^{(t)} = [\mathbf{h}^{(t)}] \in \mathbb{R}^{F\times N}$ denotes the representation of t^{th} layer with $\mathbf{H}^{(0)} = \mathbf{X}$. $\mathrm{ReLU}(.) = \max(0, .)$ is a nonlinear activation function. $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. According to GCN (Kipf and Welling 2017), the adoption of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ instead of $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where the two terms are for self-loop and propagation, respectively, is just a renormalization trick. Thus, Eq. (7) can be written as

$$\mathbf{H}^{(t+1)} = \mathrm{ReLU}\left(\mathbf{W}^{(t+1)}\mathbf{H}^{(t)}(\mathbf{I} + \bar{\mathbf{A}})\right), \tag{7}$$

where $\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetric Random-Walk Markov Matrix and is just for propagation without self-loop. The node-wise of Eq. (7) can be written as

$$\mathbf{h}_i^{(t+1)} = \sigma\Big(\mathbf{W}^{(t+1)} \sum_{N(i)\cup\{i\}} \alpha_{ij}\mathbf{h}_j^{(t)}\Big), \tag{8}$$

where the propagation weight α_{ij} fixed. $N(i)$ denotes the neighbourhoods of node v_i. $\sigma(.)$ stands for nonlinear function, such as ReLU and softmax function. The GCN model is only parameterized by the weight matrices $\mathbf{W}^{(t)}$, which can be learned by minimizing the cross-entropy between the ground truth and predictive labels on labelled nodes as

$$\mathcal{F} = -\sum_{v_i\in\mathcal{V}_l}\sum_{k=1}^{K} z_{ik}\ln y_{ik}, \tag{9}$$

where $\mathbf{Y} = [y_{ik}] = \mathbf{H}^{(L)}$ is the output of last layer. Although GCN gain significant performance improvement, its main drawback is the fixed propagation weights α_{ij}.

Multi-scale Extensions. SGC (Wu et al. 2019) simplifies GCN by successively removing nonlinear mapping fucntion and collapsing weight matrices between consecutive layers, and the resulting model is equivalent to

$$\mathbf{H}^{(L)} = \mathbf{W}\mathbf{X}\hat{\mathbf{A}}^L, \tag{10}$$

which is equivalent to adopting L-hop information. $\hat{\mathbf{A}}^L$ denotes the L matrix multiplication of $\hat{\mathbf{A}}$. A natural extension is to employ multiple-hop information. N-GCN (Abu-El-Haija et al. 2019a) and MixHop (Abu-El-Haija et al. 2019b) concatenate the results of multiple hops as

$$\mathbf{H}^{(t+1)} = \Big\Vert_{j=0}^{P} \sigma\left(\mathbf{W}^{(j)}\mathbf{H}^{(t)}\hat{\mathbf{A}}^j\right), \tag{11}$$

where $\Vert$ stands for the concatenation. LanczosNet (Liao et al. 2019) and Krylov GCN (Luan et al. 2019) propose to sum the results of multiple hops as

$$\mathbf{H}^{(t+1)} = \sigma\left(\sum_{j=0}^{P}\mathbf{W}^{(j)}\mathbf{H}^{(t)}\hat{\mathbf{A}}^j\right), \tag{12}$$

and some algorithms, such as Lanczos algorithm and Truncated Krylov algorithm, are adopted to speedup the computation.

Initial Residual in GCN. Recently GCN with Initial residual and Identity mapping (GCNII) (Chen et al. 2020) proposes to alleviate the over-smoothing issue in GCN by adding initial residual to each graph convolutional layer as

$$\begin{aligned}\mathbf{H}^{t+1} &= \sigma\Big(\mathbf{W}^{(t+1)}\big((1-\gamma_t)\mathbf{H}^t\hat{\mathbf{A}} + \gamma_t\mathbf{X}\big)\Big), \\ &= \sigma\Big(\mathbf{W}^{(t+1)}\big((1-\gamma_t)\mathbf{H}^t + (1-\gamma_t)\mathbf{H}^t\bar{\mathbf{A}} + \gamma_t\mathbf{X}\big)\Big)\end{aligned} \tag{13}$$

where γ_t is hyperparameter. Note that, similar to GCN, the $\hat{\mathbf{A}}$ consists of two parts, i.e, diagonal for self-loop and non-diagonal for propagatio The term $\gamma_t\mathbf{X}$ ensures that the final representation of each node retains at least a fraction of the input layer $\mathbf{X}$ even if we stack many layers. By stacking multiple GCN layer with Initial residua, the performance is significantly improved.

Learning Propagation Weight. Based on the propagation perspective of GCN (Gilmer et al. 2017), many efforts have been paid to make propagation learnable. Graph Attention Network (GAT) (Velickovic et al. 2018), Gated Attention Network (GaAN) (Zhang et al. 2018) and Probabilistic GCN (Yang et al. 2020) model the propagation weights as the function of the attributes of connecting nodes via normalized attention mechanism as

$$\alpha_{ij} = \exp(e_{ij}) / \sum_{k \in Ni} \exp(e_{ik}). \tag{14}$$

where e_{ij} denotes the similarity between nodes v_i and v_j. The similarity function of attributes can be specifies as

$$e_{ij}^{GAT} = \text{LeakyReLU}(\mathbf{b}[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j]) \tag{15}$$

$$e_{ij}^{GaAN} = (\mathbf{W}\mathbf{h}_i)^T \mathbf{O} (\mathbf{W}\mathbf{h}_j) \tag{16}$$

$$e_{ij}^{PGCN} = -(\mathbf{W}\mathbf{h}_i - \mathbf{W}\mathbf{h}_j)^T \mathbf{\Sigma} (\mathbf{W}\mathbf{h}_i - \mathbf{W}\mathbf{h}_j) \tag{17}$$

where $\mathbf{b}$, $\mathbf{O}$ and $\mathbf{\Sigma}$ are learnable parameters. Some efforts have been paid to simplify them, such as setting $\mathbf{O}$ as identity matrix, i.e., $\mathbf{O} = \mathbf{I}$ or constraining $\mathbf{\Sigma}$ as diagonal matrix.

Graph Representation Learning Framework

In this section, an intuitive graph representation learning framework is first provided followed by the extension to considering noisy graph topology.

Framework Overview

Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, the following two requirements are natural to obtain the node representation. 1) **Unary Constraint.** The representation of node v_i, denoted as $\mathbf{u}_i$, should be similar with its original attribute $\mathbf{x}_i$. 2) **Pairwise Constraint.** Connected nodes should possess similar representation. Thus, the objective function can be formulated as

$$\mathcal{C}(\mathbf{U}) = \sum_{i=1}^{N} \text{sim}(\mathbf{x}_i, \mathbf{u}_i) + \lambda \sum_{(i,j) \in \mathcal{E}} o_{ij} \text{dis}(\mathbf{u}_i, \mathbf{u}_j), \tag{18}$$

where the first term is for the first requirement with similarity measurement $\text{sim}(.,.)$ and the second term for the second requirement with the distance function $\text{dis}(.,.)$ and the degree of similarity o_{ij}. For simplicity, Euclidean distance is adopted as the measurement as

$$\mathcal{C}(\mathbf{U}) = \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{u}_i||_2^2 + \lambda \sum_{(i,j) \in \mathcal{E}} o_{ij} ||\mathbf{u}_i - \mathbf{u}_j||_2^2, \tag{19}$$

where the second term can be seen as the graph regularization (**?**) with graph structure as $\mathbf{O} = [o_{ij}]$. Sometimes, the inner product can also be employed as the the similarity function as

$$\mathcal{L}(\mathbf{U}) = -2 \sum_{i=1}^{N} \mathbf{x}_i^T \mathbf{u}_i + \lambda \sum_{(i,j) \in \mathcal{E}} o_{ij} ||\mathbf{u}_i - \mathbf{u}_j||_2^2. \tag{20}$$

By denoting $\mathbf{U} = [u_i] \in \mathbb{R}^{F \times N}$ as the collection of node representation, the solution to Eqs. (19) and (20) can be obtain by solving the following linear least-squares problem

$$\mathbf{U}(\mathbf{I} + \mathbf{M}) = \mathbf{X}, \tag{21}$$

$$\mathbf{U}\mathbf{M} = \mathbf{X}, \tag{22}$$

where $\mathbf{I}$ is the identity matrix. And the Laplacian matrix $\mathbf{M}$ is

$$\mathbf{M} = \lambda \sum_{(i,j) \in \mathcal{E}} o_{ij} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T = \lambda (\mathbf{D}_O - \mathbf{O}), \tag{23}$$

where $\mathbf{e}_i$ is an indicator vector with the i^{th} element as 1 and $\mathbf{D}_O = \text{diag}(\mathbf{O}\mathbf{1})$ is the degree matrix of asymmetric matrix $\mathbf{O} = [o_{ij}] \in \mathbb{R}^{N \times N}$. Thus, both $\mathbf{M}$ and $\mathbf{I} + \mathbf{M}$ are symmetric and positive semidefinite, and Eqs. (21) and (22) can be directly solved via

$$\mathbf{U} = \mathbf{X}(\mathbf{I} + \mathbf{M})^{-1}, \qquad \mathbf{U} = \mathbf{X}\mathbf{M}^{-1}. \tag{24}$$

Unfortunately, these analytic solutions are computationally expensive due to the inverse operation.

Noisy Graph Topology

The second requirement in the previous subsection, i.e., connected nodes being similar, is debatable, since the noisy connections is inevitable in real-world data. Thus, the second term in Eq. (19), which is used to model the similarity of the connected nodes, can be reformulated to incorporate some preference as

$$\mathcal{C}(\mathbf{U}) = \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{u}_i||_2^2 + \lambda \sum_{(i,j) \in \mathcal{E}} o_{ij} \rho(||\mathbf{u}_i - \mathbf{u}_j||_2), \tag{25}$$

where $\rho(\cdot)$ is a penalty to model the desired preference. For example, ℓ_0 norm, i.e., $\rho(y) = \delta(y \neq 0)$ with $\delta(.)$ as Dirac delta function, and its convex relaxation ℓ_1 norm is adopted for clustering, which requires the observation from the same latent cluster to collapse into a single point. To alleviate the noisy connection, the function $\rho(\cdot)$ should automatically prune spurious inter-cluster connections while maintaining intra-cluster ones. According to the duality between robust estimation and line processes (), an auxiliary variable l_{ij} is introduced for each observed connection $(i,j) \in \mathcal{E}$ and objective function in Eq. (25) should be equivalent to

$$\begin{aligned} \mathcal{C}(\mathbf{U}, \mathbf{L}) = & \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{u}_i||_2^2 \\ & + \lambda \sum_{(i,j) \in \mathcal{E}} o_{ij} \left(l_{ij} ||\mathbf{u}_i - \mathbf{u}_j||_2^2 + \psi(l_{ij}) \right), \end{aligned} \tag{26}$$

where $\mathbf{L} = [l_{ij}] \in \mathbb{R}^{N \times N}$ is the collection of l_{ij} and $\psi(l_{ij})$ is the penalty of pruning edge (i,j). Thus, $\psi(l_{ij})$ should tend to be 0 when the edge (i,j) is maintained, and tend to be 1 when the edge (i,j) is pruned. A broad variety of robust estimators $\rho(\cdot)$ have corresponding penalty functions $\psi(l_{ij})$ such that Eqs. (25) and (26) are equivalent with respect to $\mathbf{U}$. Optimizing either of the two objectives yields the same set

of representatives $\mathbf{U}$. According to (Shah and Koltun 2017), a well-known Geman-McClure estimator (Geman and McClure 1987) is adopted as

$$\rho(y) = \frac{\mu y^2}{\mu + y^2} \tag{27}$$

where μ is a hyperparameter. It is equivalent to setting the penalty function $\psi(l_{ij})$ in Eq. (26) as

$$\psi(l_{ij}) = \mu \left(\sqrt{l_{ij}} - 1 \right)^2, \tag{28}$$

which satisfies the requirement to $\psi(l_{ij})$ on penalizing the edge pruning. $\psi(l_{ij})$ achieves its maximum penalty, if edge (i, j) is pruned, i.e., $l_{ij} = 0$.

Since the objective function in Eq. (26) is biconvex on $\mathbf{U}$ and $\mathbf{L}$, then it can be optimized by alternatively updating $\mathbf{U}$ and $\mathbf{L}$. When $\mathbf{L}$ is fixed, objective function can be minimized with respect to $\mathbf{U}$ via Eqs. (21) and (22) by setting $o_{ij} = o_{ij} l_{ij}$. When $\mathbf{U}$ is fixed, $\mathbf{L}$ can be obtained by minimizing the second term in Eq. (26) as

$$l_{ij} = \left(\frac{\mu}{\mu + ||\mathbf{u}_i - \mathbf{u}_j||_2^2} \right)^2. \tag{29}$$

Since the refined pairwise similarity between nodes v_i and v_j is $o_{ij} l_{ij}$, l_{ij} plays the role of re-weighting the original similarity according to their latent space distance $||\mathbf{u}_i - \mathbf{u}_j||$. Thus, Robust GRL may gradually refine the noisy topology and obtain the robust node representation.

Analysis of Existing GCNNs

Previous researches illustrate that the success of GCN and its variants mainly attribute to their essence of smoothing from the spatial perspective (Li, Han, and Wu 2018). In this section, existing GCNNs are analyzed under the proposed graph representation learning (GRL) framework by comparing the solution to GRL and the propagation in GCNNs.

According to the SGC (Wu et al. 2019), weight matrices between consecutive layers can be collapsed and the nonlinear mapping function can be removed. Thus, our analysis focuses on the propagation strategy and its combination mechanism without the weight matrix and nonlinear mapping function.

Theorem 1. *The scheme of GCN with initial residual in Eq. (13) is equivalent to solving the Graph Representation Learning in Eq. (20), where pairwise similarity* $\mathbf{O} = [o_{ij}]$ *is set as the symmetric Random-Walk Markov Matrix* $\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$*, with gradient descent.*

Proof. Based on the gradient descent for linear system in Eq. (3), the solution to $\mathbf{UM} = \mathbf{X}$, which is equivalent to the Graph Representation Learning in Eq. (20), can be obtained iteratively as

$$\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} + \mu_t (\mathbf{X} - \mathbf{U}^{(t)} \mathbf{M}) = \mathbf{U}^{(t)} (\mathbf{I} - \mu_t \mathbf{M}) + \mu_t \mathbf{X}. \tag{30}$$

According to Eq. (23) and $\mathbf{O} = \bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$, $\mathbf{M} = \mathbf{D}_O - \mathbf{O}$ is the Laplacian matrix of the similarity matrix $\mathbf{A}$, and $\mathbf{D}_O = \mathbf{I}$. Then, Eq. (30) can be reformulated as

$$\mathbf{U}^{(t+1)} = (1 - \mu_t) \mathbf{U}^{(t)} + \mu_t \mathbf{U}^{(t)} \bar{\mathbf{A}} + \mu_t \mathbf{X}. \tag{31}$$

The three terms in the right hand side, i.e., $\mathbf{U}^{(t)}$, $\mathbf{U}^{(t)} \bar{\mathbf{A}}$ and $\mathbf{X}$, correspond to the self-loop $\mathbf{H}^{(t)}$, propagation $\mathbf{H}^{(t)} \bar{\mathbf{A}}$ and residual $\mathbf{X}$ in the Residual GCN in Eq. (13), thus they are equivalent. □

Corollary 1. *The term* $\mathbf{X}$ *in gradient descent is important to properly approximate the solution to GRL. Thus, the initial residual is important in GCNs.*

Theorem 1 and Corollary 1 illustrate the importance of the initial residual connection in (GCNII) (Chen et al. 2020). It shows that the reason why GCN layers with initial residual connection can outperforms the original GCN is because it can approximates the solution to GRL. In contrast, if the residual connection $\mathbf{X}$ lacks, the approximation error of each gradient descent step can't be ignored, thus the cumulative error is very significant in approximating the solution to GRL. Thus, stacking of GCN without initial residual connection degrades the performance.

Theorem 2. *The scheme of the multi-scale extension to GCN in Eq. (12) is equivalent to solving the Graph Representation Learning in Eq. (20), where pairwise similarity* $\mathbf{O} = [o_{ij}]$ *is set as the symmetric Random-Walk Markov Matrix with self-loop* $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$*, with the higher order approximation to the inverse of* $\mathbf{M}$ *in Eq. (5).*

Proof. Based on the Caley-Hamilton theorem in Eq. (5), $(\mathbf{I} - \mathbf{A})^{-1}$ can be expressed as the summation of finite terms as

$$(\mathbf{I} - \mathbf{A})^{-1} \approx \sum_{j=0}^{J} \beta_j \mathbf{A}^j, \tag{32}$$

where $J < N$. Since Eq. (24) shows that $\mathbf{U} = \mathbf{X}\mathbf{M}^{-1}$, where $\mathbf{M} = \mathbf{D}_O - \mathbf{O} = \mathbf{I} - \hat{\mathbf{A}}$, is the analytic solution to the problem of the Graph Representation Learning in Eq. (20), we get

$$\mathbf{U} = \mathbf{X}(\mathbf{I} - \mathbf{A})^{-1} \approx \sum_{j=0}^{J} \beta_j \mathbf{X} \hat{\mathbf{A}}^j. \tag{33}$$

Thus, the approximation solution to GRL is equivalent to the multi-scale extension to GCN in Eq. (12). □

Theorems 1 and 2 demonstrate that the propagations in GCNs are equivalent to the optimization steps (gradient descent step or high order approximation step) of the our proposed Graph Representation Learning. That is, they are induced by the numerical optimization of pairwise similarity requirement. Thus, the pairwise similarity constraints, which determines how to propagate, is the key to GNNs.

Next, the attention mechanism, such as GAT (Velickovic et al. 2018), GaAN (Zhang et al. 2018) and PGCN (Yang et al. 2020), is compared with our proposed Graph Representation Learning with noisy topology refinement.

Corollary 2. *The solution to our proposed Robust Graph Representation Learning with noisy topology refinement shown in Eq. (25) with* $\rho(\cdot)$ *as in Eq. (27) is similar to the GCNNs with learnable propagation weight as Eq. (17).*

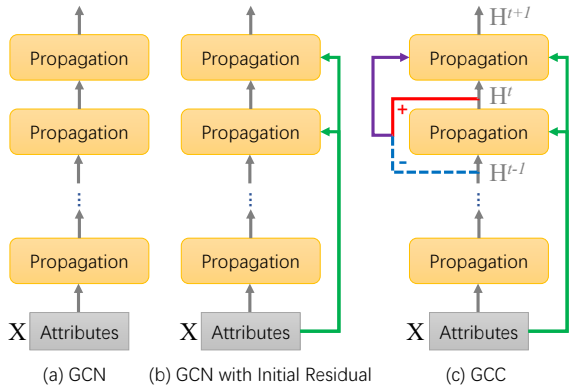

Figure 1: The architectures of GCN, GCN with Initial Residual and our proposed GCC.

Both of Robust GRL and GCNNs with learnable propagation weight iteratively propagate attribute and learn weight (refine similarity). Thus, this corollar ycan be illustrated with following two steps. 1) GCNNs with learnable propagation weight perform the similar propagation as GCNNs without learnable propagation weight shown in Eq. (8). 2) The weight learning function in GCNNs shown in Eq. (17) is similar to the similarity refinement function $o_{ij}l_{ij}$, where l_{ij} is determined in Eq. (29). That is, b†oth of them are the monotonically decreasing function of $||\mathbf{u}_i - \mathbf{u}_j||_2$.

Corollary 2 demonstrates that the learning of propagation weights is also induced by the pairwise similarity robust estimation function $\rho(\cdot)$. Therefore, both propagation and its wights learning in GCNs are determined and induced by the numerical optimization of pairwise similarity objective function. In other words, it provides insights of GCN and its variants from the perspective of numerical optimization that

The propagation as well as its weight learning are not the essence of the GCNs, but induced by the numerical optimization of pairwise similarity requirement.

Graph Conjugate Convolutional Network

According to the explanation to the GCN and its variants from the perspective of the numerical optimization algorithms of our proposed GRL framework in previous section, a novel Graph Conjugate Convolutional (GCC) network is presented to approximate the solution to GRL with fast convergence. The Graph Conjugate Convolutional (GCC) network is constructed by stacking multiple novel Graph Conjugate Convolutional layer. The design of Graph Conjugate Convolutional layer is inspired by fast convergence of conjugate gradient descent compared to the common gradient descent. According to the difference between conjugate gradient descent in Eq. (4) from common gradient descent in Eq. (3), each Graph Conjugate Convolutional layer consists of three components as

$$\mathbf{H}^{(t+1)} = \sigma\Big(\mathbf{W}^{(t+1)}\big((1-\gamma_t)\mathbf{H}^{(t)}\hat{\mathbf{A}} + \gamma_t\mathbf{X} + \kappa_t(\mathbf{H}^{(t)} - \mathbf{H}^{(t-1)})\big)\Big), \quad (34)$$

Table 1: Datasets.

Datasets	Nodes	Edges	Classes	Attributes
Texas	183	328	5	1,703
Cornell	195	304	5	1,703
Wisconsin	262	530	5	1,703
Chameleon	2,277	36,101	4	2,325
Cora	2,708	5,429	7	1,433
Citeseer	3,312	4,732	6	3,703
Pubmed	19,729	44,338	3	500
PPI	56,944	818,716	121	50

where the first two terms are the same as in GCNII in Eq. (13). They act as the propagation, including self-loop, and initial residual. The third term $\kappa_t(\mathbf{H}^{(t)} - \mathbf{H}^{(t-1)})$ is the difference between the representation in t^{th} and $(t-1)^{th}$ layer, as shown in Figure 1(c).

Note that this term is very different from just adding $\mathbf{H}^{(t)}$ term as in GCN (Kipf and Welling 2017). GCN has shown that just adding $\mathbf{H}^{(t)}$, which is equivalent to the residual connection as ResNet (He et al. 2016), can't improve the performance as stacking many layers. This may be caused by that the accumulation effect of representation in low-level layers induces the overfitting as layers increase. In contrast, our introduced term $\kappa_t(\mathbf{H}^{(t)} - \mathbf{H}^{(t-1)})$ only adopts the *obtained* information of the t^{th} layer, which can be represented as the difference between the input and output of the t^{th} layer as shown in the purple arrow of Figure 1(c), as the input to the $(t+1)^{th}$ layer. Thus, it can alleviate the issue in the residual connection.

Besides, our proposed Graph Conjugate Convolutional layer can be employed by other propagation based GCNNs, since it essentially sublimes the propagation process. For example, Graph Conjugate Attention (GCA) network can be obtained by replacing its propagation part with our proposed Graph Conjugate Convolution according to Corollary 2. Furthermore, our proposed GCC and GCA can be applied to both transductive and inductive semi-supervised node classification tasks.

Evaluations

In this section, the performance of our proposed GCC and GCA is experimentally evaluated on transductive and inductive semi-supervised node classification task.

Experimental Setup

Dataset: For transductive learning task, two kinds of networks, citation networks and webpage networks, are adopted. Cora, Citeseer, and Pubmed are citation network benchmark datasets (Sen et al. 2008), where nodes and edges denote research papers and undirected citations, respectively. In addition to the network structure, node content, which is represented by the bag-of-word representation of the documents, is available. According to the disciplines, papers are categorized into various classes. Texas, Cornell and Wisconsin are webpage networks from WebKB, where

Table 2: Comparison on transductive node classification in terms of AC (%).

Methods	Cora	Citeseer	Pubmed	Texas	Cornell	Wisconsin	Chameleon
GCN	81.5	71.1	79.0	52.1	52.7	45.8	28.2
GAT	83.1	70.8	78.5	58.3	54.3	49.4	42.9
PR-GCN	83.3	71.8	80.1	65.4	73.5	69.0	54.3
JKNet	81.1	69.8	78.1	56.4	57.3	48.8	60.1
DropEdge	83.5	72.7	79.5	57.8	61.6	50.2	61.7
GCNII	85.5	73.4	80.2	69.4	74.8	74.1	60.6
GCC	86.1	**74.3**	**81.1**	71.15	76.44	**75.37**	**61.95**
GCA	**86.3**	73.6	**81.1**	**71.62**	**76.72**	74.87	61.27

Table 3: Results on PPI.

Methods	PPI
GraphSAGE	61.27
VR-GCN	97.80
GAT	97.32
JKNet	97.61
GeniePath	98.52
Cluster-GCN	99.33
GCNII	99.53
GCC	**99.60**
GCA	99.58

nodes and edges represent web pages and hyperlinks, respectively. Node features are the bag-of-words representation of web pages. The web pages were manually classified into the five categories. Chameleon is a page-page network on specific topics in Wikipedia, where nodes, edges and features have the similar meaning as in WebKB. For inductive learning task, 24 Protein-Protein Interaction (PPI) networks are employed (Hamilton, Ying, and Leskovec 2017). Dataset statistics are summarized in Table 1.

Baselines: For transductive learning task, the baselines fall into two categories. GCN (Kipf and Welling 2017) and GAT (Velickovic et al. 2018) are two basic models, which may induce over-smoothing and overfitting. Besides, other 5 recently proposed method to overcome the oversmoothing issue are employed. They are PR-GCN (Klicpera, Bojchevski, and Günnemann 2019), JKNet (Xu et al. 2018), DropEdge (Rong et al. 2020) and GCNII (Chen et al. 2020). Note that although edge dropping strategy in (Rong et al. 2020) can be applied to many other basic model, such as GCN, DropEdge is used to represent to the combination of edge dropping and IncepGCN proposed in (Rong et al. 2020). IncepGCN is the extension of inception network (Szegedy et al. 2016) to GNN by combining 1-hop, 2-hop and 3-hop graph convolutional operations in one IncepGCN layer. For inductive learning, in additional to the GAT (Velickovic et al. 2018), JKNet (Xu et al. 2018) and GCNII (Chen et al. 2020), other 4 state-of-the-art methods, i.e., GeniePath (Liu et al. 2019), Cluster-GCN (Chiang et al. 2019) GraphSAGE (Hamilton, Ying, and Leskovec 2017) and VR-GCN (Chen, Zhu, and Song 2018), are employed.

Parameter Setting: Adam SGD optimizer (Kingma and Ba 2015) is adopted with learning rate as 0.001. Besides, early stopping with a patience of 100 epochs and ℓ_2 regularization (0.0006) is employed to prevent overfitting. $\gamma_t = 0.1$ and $\kappa_t = 0.2$ for transductive learning, while $\gamma_t = 0.45$ and $\kappa_t = 0.32$ in inductive learning. Similar to GCNII (Chen et al. 2020) identity mapping is employed to enhance the learnable mapping $\mathbf{W}$. The number of layers (depth) is selected from 8, 16 and 32. Its impact on performance will be investigated in the last subsection.

Experimental Results Analysis

Transductive Learning: The fixed split for training, validation and testing introduced in (Yang, Cohen, and Salakhutdinov 2016), i.e., 20 nodes per class for training, 500 nodes for validation and 1,000 nodes for testing, are adopted for three citation network Cora, Citeseer, and Pubmed. For each webpage network, i.e., Chameleon, Texas, Cornell and Wisconsin, nodes in each class is randomly split into 60%, 20%, and 20% for training, validation and testing. The results in term of accuracy (AC) are shown in Table 2. It demonstrates that our proposed GCC and GCA consistently outperform the state-of-the-art GNNs. They possess the ability to extract more information from high-order neighbourhoods. The performance improvement mainly due to the fast convergence of the conjugate part in GCC. From propagation perspective, it prevents the over-smoothing caused by inaccurate approximation. The impacts of the depth on performance are given in the Appendix.

Inductive Learning: Following (Velickovic et al. 2018), The 24 graphs are divided to 20 graph for training, 2 graphs for validation and 2 graphs for testing. The results in terms of F1 score are shown in Table 3. GCC defeats other state-of-the-art with 8 graph conjugate convolutional layers. It illustrates that GCC can effectively combine multi-hop information with efficient graph conjugate convolutional operation. This efficiency may attributes to that graph conjugate convolution integrates the gained information (difference between output and input) in each graph convolutional layer as shown in Figure 1(c).

Conclusions

In this paper, the propagation, which induces over-smoothing issue in Graph Convolutional Network (GCN) and its variant, is investigated. To this end, an intuitive Graph Representation Learning (GRL) framework, which simply constrains the node representation similar with the original attribute, and encourages the connected nodes possess similar representations (pairwise constraint), is presented. Based on GRL, we show that the propagation as well as its weight learning are not the essence of the GCNs, but induced by the numerical optimization of pairwise similarity requirement. Thus, inspired by the superiority of conjugate gradient descent compared to common gradient descent, a novel Graph Conjugate Convolutional (GCC) network, which adopts the obtained information of the last layer as the input to the next layer, is presented. Extensive experiments on transductive and inductive semi-supervised node classification task shows that GCC can enjoy the deep network via effectively and efficiently multi-hop information combination.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61972442, Grant U1936208 and Grant 61802282, in part by the Key Research and Development Project of Hebei Province of China under Grant 20350802D and 20310802D; in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Hebei Province Innovation Capacity Enhancement Project under Grant 199676146H, in part by the Natural Science Foundation of Tianjin of China under Grant 20JCYBJC00650, and in part by the Key Program of the Chinese Academy of Sciences under Grant QYZDB-SSW-JSC003.

References

Abu-El-Haija, S.; Kapoor, A.; Perozzi, B.; and Lee, J. 2019a. N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification. In *UAI*, 310.

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Steeg, G. V.; and Galstyan, A. 2019b. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *ICML*, volume 97, 21–29.

Balcilar, M.; Renton, G.; Héroux, P.; Gauzere, B.; Adam, S.; and Honeine, P. 2020. Bridging the Gap Between Spectral and Spatial Domains in Graph Neural Networks. *arXiv preprint arXiv:2003.11702* .

Chen, J.; Zhu, J.; and Song, L. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *ICML*, 941–949.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and Deep Graph Convolutional Networks. In *ICML*.

Chiang, W.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; and Hsieh, C. 2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *SIGKDD*, 257–266.

Decell, Jr, H. P. 1965. An application of the Cayley-Hamilton theorem to generalized matrix inversion. *SIAM review* 7(4): 526–528.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, 3837–3845.

Geman, S.; and McClure, D. 1987. Statistical methods for tomographic image reconstruction. *Bulletin of the International Statistical Institute* 52(4): 5–21.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*, 1263–1272.

Golub, G. H.; and Van Loan, C. F. 1996. *Matrix Computations (3rd Ed.)*. USA: Johns Hopkins University Press. ISBN 0801854148.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Jin, D.; Liu, Z.; Li, W.; He, D.; and Zhang, W. 2019. Graph Convolutional Networks Meet Markov Random Fields: Semi-Supervised Community Detection in Attribute Networks. In *AAAI*, 152–159.

Jin, D.; Song, X.; Yu, Z.; Liu, Z.; Zhang, H.; Cheng, Z.; and Han, J. 2021. BiTe-GCN: A New GCN Architecture via Bidirectional Convolution of Topology and Features on Text-Rich Networks. In *WSDM*.

Jin, D.; Yu, Z.; He, D.; Yang, C.; Yu, P. S.; and Han, J. 2020. GCN for HIN via Implicit Utilization of Attention and Metapaths. *arXiv preprint arXiv:2007.02643* .

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

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

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

Li, G.; Müller, M.; Thabet, A. K.; and Ghanem, B. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs? In *ICCV*, 9266–9275. doi:10.1109/ICCV.2019.00936.

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

Liao, R.; Zhao, Z.; Urtasun, R.; and Zemel, R. S. 2019. LanczosNet: Multi-Scale Deep Graph Convolutional Networks. In *ICLR*.

Liu, Z.; Chen, C.; Li, L.; Zhou, J.; Li, X.; Song, L.; and Qi, Y. 2019. GeniePath: Graph Neural Networks with Adaptive Receptive Paths. In *AAAI*, 4424–4431.

Luan, S.; Zhao, M.; Chang, X.; and Precup, D. 2019. Break the Ceiling: Stronger Multi-scale Deep Graph Convolutional Networks. In *NeurIPS*, 10943–10953.

Ma, J.; Cui, P.; Kuang, K.; Wang, X.; and Zhu, W. 2019. Disentangled Graph Convolutional Networks. In *ICML*, 4212–4221.

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

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Gallagher, B.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *AI Magazine* 29(3): 93–106.

Shah, S. A.; and Koltun, V. 2017. Robust continuous clustering. *PNAS* 114(37): 9814–9819.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2818–2826.

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

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.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2020. A comprehensive survey on graph neural networks. *IEEE TNNLS* 1–14.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

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

Yang, L.; Guo, Y.; Gu, J.; Jin, D.; Yang, B.; and Cao, X. 2020. Probabilistic Graph Convolutional Network via Topology-Constrained Latent Space Model. *IEEE Transactions on Cybernetics* 1–14.

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

Yang, L.; Wu, F.; Gu, J.; Wang, C.; Cao, X.; Jin, D.; and Guo, Y. 2020. Graph Attention Topic Modeling Network. In *WWW*, 144–154.

Yang, L.; Wu, F.; Wang, Y.; Gu, J.; and Guo, Y. 2019b. Masked Graph Convolutional Network. In *IJCAI*, 4070–4077.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML*, 40–48.

Zhang, J.; Shi, X.; Xie, J.; Ma, H.; King, I.; and Yeung, D. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *UAI*, 339–349.