AdjMix: simplifying and attending graph convolutional networks

Xun Liu¹ · Fangyuan Lei^{2,3} · Guoqing Xia¹ · Yikuan Zhang¹ · Wenguo Wei⁴

Received: 4 July 2021 / Accepted: 19 October 2021 / Published online: 8 November 2021
© The Author(s) 2021

Abstract

Simple graph convolution (SGC) achieves competitive classification accuracy to graph convolutional networks (GCNs) in various tasks while being computationally more efficient and fitting fewer parameters. However, the width of SGC is narrow due to the over-smoothing of SGC with higher power, which limits the learning ability of graph representations. Here, we propose AdjMix, a simple and attentional graph convolutional model, that is scalable to wider structure and captures more nodes features information, by simultaneously mixing the adjacency matrices of different powers. We point out that the key factor of over-smoothing is the mismatched weights of adjacency matrices, and design AdjMix to address the over-smoothing of SGC and GCNs by adjusting the weights to matching values. Experiments on citation networks including Pubmed, Citeseer, and Cora show that our AdjMix improves over SGC by 2.4%, 2.2%, and 3.2%, respectively, while achieving same performance in terms of parameters and complexity, and obtains better performance in terms of classification accuracy, parameters, and complexity, compared to other baselines.

Keywords Graph convolutional network · Simple graph convolution · Over-smoothing · Complexity · Simplifying and attending graph convolution

Introduction

Motivated by the increasing proliferation of graph structured data in real-world applications and the limited expressive power of convolutional neural networks (CNNs) for learn-

ing such data, graph convolutional networks (GCNs) [21] and subsequent variants have experienced great attention and have become popular topics. These models show promising performance in various fields, including citation networks [21,38], social networks [7], traffic networks [13,25], applied chemistry [29], recommender systems [8,44], and so on.

Each layer in GCNs essentially acts as an operation of Laplacian smoothing [26]. Applying the operation twice, the nodes features in the same connected component are similar, which eases the classification task. As the times increase, the node features are more similar. Eventually, the node features tend to the fixed value [26], which makes them indistinguishable. This is called the over-smoothing problem. A key factor of the over-smoothing is the mismatched weights of adjacency matrices in the various powers. Therefore, we can effectively tackle the over-smoothing by adjusting the weights to appropriate and better values.

In addition to the over-smoothing, GCNs may inherit excess complexity and superfluous computation [40]. To reduce the complexity, a recent surge of research interest has concentrated on simple graph convolution (SGC) [40]. By removing the nonlinearities between GCNs layer and collapsing the weight matrices between consecutive layers, SGC achieves similar or even superior classification accuracy to

✉ Fangyuan Lei
leify@gpnu.edu.cn

Xun Liu
liuxun.stf@gmail.com

Guoqing Xia
xgq@mail.seig.edu.cn

Yikuan Zhang
zyk@mail.seig.edu.cn

Wenguo Wei
447402586@qq.com

¹ Department of Electronics, Software Engineering Institute of Guangzhou, Guangzhou, China

² Guangdong Provincial Key Laboratory of Intellectual Property Big Data, Guangdong Polytechnic Normal University, Guangzhou, China

³ School of Electronic and Information, Guangdong Polytechnic Normal University, Guangzhou, China

⁴ Network and Information Center, Guangdong Polytechnic Normal University, Guangzhou, China

Fig. 1 Our AdjMix architecture ($k = 3$)

GCNs in various application areas while being computationally more efficient and fitting fewer parameters. Nevertheless, SGC fails to adjust the weights of adjacency matrices in the various powers to matching values, thus bringing potential concerns of over-smoothing when taking higher powers. Limited to the over-smoothing, the width (the power of adjacency matrix) of SGC is narrow. This narrow mechanism naturally limits the scale of the receptive field. Due to the restricted receptive field, SGC has difficulty in obtaining more global information and propagating adequate label information.

Here, we propose a novel AdjMix that learns to adjust the weights of adjacency matrices in the different powers to matching values and capture more nodes and global information in an end-to-end fashion (Fig. 1). AdjMix can be viewed as a graph analogue of Inception [36], which allows for convolution model to increase the receptive field by mixing different kernel sizes. The main challenge to combine the adjacency matrices of different powers into a new single adjacency matrix is that the weights of adjacency matrices in the various powers are incompatible. To address the above challenges, we require a model that learns how to adjust the weights to matching values.

Our contributions are three-fold. (1) We provide new insights and point out that the key factor of this over-smoothing is the mismatched weights of adjacency matrices in the various powers. (2) We propose a new AdjMix to tackle the over-smoothing by adjusting the weights to appropriate values. Because AdjMix has no the limits of the over-smoothing, AdjMix can obtain more global information through taking higher powers instead of increasing depth and is scalable to wider structure. (3) Through an empirical assessment on node classification tasks, we show that

the AdjMix matches the complexity and parameters of SGC while outperforming SGC and other state-of-the-art methods significantly in terms of classification accuracy.

Related works

Spectral CNN [6] is the first successful attempt at extending CNNs on graphs for dealing with graph-based tasks. Defferrard et al. [10] propose a localized filter using Chebyshev polynomials to avoid the computationally expensive Laplacian eigendecomposition of [6]. Later, Kipf and Welling [21] propose graph convolutional networks (GCNs) by simplifying the Chebyshev polynomials based on a redefined propagation matrix. Li et al. [26] prove that each convolution of GCNs is actually a special form of Laplacian smoothing, which suffers from potential limits of over-smoothing with many graph convolution layers. GraphSAGE [14] leverages the node information with sampling and aggregation to generate node embeddings for tackling the limitations of transductive learning. FastGCN [7] regards the convolution as integral transforms of embedding functions to significantly reduce the time and memory resources. Xu et al. [42] analyze the learning ability of graph neural networks (GNNs) for capturing different graph structures and propose graph isomorphism network (GIN) that is as powerful as the Weisfeiler–Lehman test for graph isomorphism. Klicpera et al. [22] propose an improved propagation scheme with personalized PageRank to leverage the information from a large neighborhood. Li et al. [27] propose proper low-pass graph convolutional filters that inject graph relations into data features in semi-supervised learning. Velickovi et al. [39] propose a deep graph infomax (DGI) that is based upon mutual information to learn graph embeddings on both transduc-

tive and inductive learning tasks. By combining these works of [9,17,18,32,33,35] with DGI, DGI is more powerful. Several researches obtain multi-scale information via high-order propagation matrix [2,3,24,29,31].

A recent surge of research interest has focused on simpler and linear models to achieve efficient computations for node and graph learning tasks. Thekumparampil et al. [37] propose a linear GCNs model that removes all intermediate non-linear activation functions to simplify computations, which achieves a comparable to GCNs and other state-of-the-art models. SGC [40], as a simpler and efficient variant of GCNs, removes the nonlinearities between GCNs layer and collapses the weight matrices between consecutive layers, has achieved comparable performance to GCNs on a variety of tasks.

Many researches demonstrate that graph attentional models try to assign various edge weights based on node features and have contributed to performance improvement on graph learning tasks [19,23,37,38,45]. Nevertheless, these models with attention mechanism add significant overhead to complexity and parameters. There are many other works [5,41,46] for a more comprehensive review.

Proposed method

Preliminaries

We follow [3,21] to introduce graph notations and problem definition. We represent an undirected graph $\mathcal{G}$ with n vertices and e edges as (ε, ν, A, X) , where ε and ν are the edge set and vertex set, respectively, $X \in \mathbb{R}^{n \times c_0}$ is the node feature matrix assuming each node has c_0 features, and $A \in \{0, 1\}^{n \times n}$ is the adjacency matrix with each entry a_{ij} describing the edge weights between vertex i and j . We use edge weights to indicate connection strengths between vertices (nodes).

Graph convolutional networks. We follow [21] to introduce the convolution of GCNs, which is defined as:

$$H^{(k+1)} \leftarrow \text{ReLU}(SH^{(k)}\theta^{(k)}), \quad (1)$$

where S is the normalized adjacency matrix with added self-loops, with $S = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$. Here $\tilde{A} = A + I$, $I \in \mathbb{R}^{n \times n}$ is the identity matrix, and $\tilde{D}$ is the degree matrix of $\tilde{A}$, with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $H^{(k)}$ and $\theta^{(k)}$ denote the input node representations and the learned weight matrix at layer k . Specially, $H^{(k)} = X$ when $k = 1$, which serves as input to the first GCNs layer. We use the nonlinear activation function ReLU to achieve the pointwise nonlinear transformation, with $\text{ReLU}(x) = \max(0, x)$.

Similar to CNNs, GCNs learn graph representations by stacking multiple layers. Stacking the convolutional layer

twice, the popular two-layer GCNs can be written as:

$$Y_{\text{GCNs}} = \text{softmax}(S\text{ReLU}(SX\theta^{(1)})\theta^{(2)}), \quad (2)$$

where $\theta^{(1)}$ and $\theta^{(2)}$ are different weight matrices. We apply softmax classifier to predict the labels for node classification, with $\text{softmax}(\cdot) = \frac{\exp(\cdot)}{\sum_i \exp(\cdot)}$.

Simple graph convolution. By repeatedly stacking GCNs layers, we describe k -layer GCNs in general form as:

$$\hat{Y} = \text{softmax}(S\text{ReLU}(\cdots S\text{ReLU}(SX\theta^{(1)})\theta^{(2)} \cdots)\theta^{(k)}). \quad (3)$$

Wu et al. [40] show that the nonlinearity between GCNs layer is unimportant for classification tasks. Therefore, for k -layer GCNs, we remove the ReLU functions between each layer. The k -layer GCNs become:

$$\hat{Y} = \text{softmax}(S \cdots S S X \theta^{(1)} \theta^{(2)} \cdots \theta^{(k)}). \quad (4)$$

To simplify notation and computations, we collapse the S into a single matrix S^k by raising S to the k -th power and reparameterize weights into a single matrix θ via $\theta = \theta^{(1)}\theta^{(2)} \cdots \theta^{(k)}$. We write the k -layer GCNs in simple form:

$$\hat{Y}_{\text{SGC}} = \text{softmax}(S^k X \theta). \quad (5)$$

We refer to the simplified model as simple graph convolution (SGC) [40]. In SGC, we reduce the excess complexity of GCNs and achieve a simplified linear model while showing competing performance compared to GCNs.

SGC encounters bottleneck

In traditional CNNs, the depth (stacking layers) increases the receptive field of internal features [15]. In SGC, we increase the receptive field by increasing the k value in Eq. (5). Equation (5) shows that SGC captures the features information of neighbor nodes that are k -hops away in the graph. Considering $S = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ and $\tilde{A} = A + I$, then:

$$S^k \leftarrow (\tilde{D}^{-\frac{1}{2}} (A + I) \tilde{D}^{-\frac{1}{2}})^k. \quad (6)$$

To analyze Eqs. (5) and (6) intuitively and clearly, we take Fig. 2 as an example for numerical experiments. Obviously,

$$\text{in Fig. 2, } A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, A + I = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \text{ and}$$

$$\tilde{D} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}. \text{ Then:}$$

$$S^1 = \begin{bmatrix} 0.33 & 0.29 & 0 & 0 & 0 & 0.33 \\ 0.29 & 0.25 & 0.29 & 0 & 0.25 & 0 \\ 0 & 0.29 & 0.33 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0.33 & 0.29 & 0 \\ 0 & 0.25 & 0 & 0.29 & 0.25 & 0.29 \\ 0.33 & 0 & 0 & 0 & 0.29 & 0.33 \end{bmatrix}, \quad (7)$$

$$S^2 = \begin{bmatrix} 0.31 & 0.17 & 0.08 & 0 & 0.17 & 0.22 \\ 0.17 & 0.29 & 0.17 & 0.17 & 0.13 & 0.17 \\ 0.08 & 0.17 & 0.31 & 0.22 & 0.17 & 0 \\ 0 & 0.17 & 0.22 & 0.31 & 0.17 & 0.08 \\ 0.17 & 0.13 & 0.17 & 0.17 & 0.29 & 0.17 \\ 0.22 & 0.17 & 0 & 0.08 & 0.17 & 0.31 \end{bmatrix}, \quad (8)$$

$$S^3 = \begin{bmatrix} 0.22 & 0.20 & 0.08 & 0.08 & 0.15 & 0.22 \\ 0.20 & 0.20 & 0.20 & 0.15 & 0.20 & 0.15 \\ 0.08 & 0.20 & 0.22 & 0.22 & 0.15 & 0.08 \\ 0.08 & 0.15 & 0.22 & 0.22 & 0.20 & 0.08 \\ 0.15 & 0.20 & 0.15 & 0.20 & 0.20 & 0.20 \\ 0.22 & 0.15 & 0.08 & 0.08 & 0.20 & 0.22 \end{bmatrix}, \quad (9)$$

$$S^4 = \begin{bmatrix} 0.21 & 0.17 & 0.11 & 0.09 & 0.17 & 0.19 \\ 0.17 & 0.21 & 0.17 & 0.17 & 0.19 & 0.17 \\ 0.11 & 0.17 & 0.21 & 0.19 & 0.17 & 0.09 \\ 0.09 & 0.17 & 0.19 & 0.21 & 0.17 & 0.11 \\ 0.17 & 0.19 & 0.17 & 0.17 & 0.21 & 0.17 \\ 0.19 & 0.17 & 0.09 & 0.11 & 0.17 & 0.21 \end{bmatrix}, \quad (10)$$

$$S^5 = \begin{bmatrix} 0.18 & 0.18 & 0.12 & 0.12 & 0.17 & 0.18 \\ 0.18 & 0.20 & 0.18 & 0.17 & 0.20 & 0.17 \\ 0.12 & 0.18 & 0.18 & 0.18 & 0.17 & 0.12 \\ 0.12 & 0.17 & 0.18 & 0.18 & 0.18 & 0.12 \\ 0.17 & 0.20 & 0.17 & 0.18 & 0.20 & 0.18 \\ 0.18 & 0.17 & 0.12 & 0.12 & 0.18 & 0.18 \end{bmatrix}, \quad (11)$$

$$S^{20} = \begin{bmatrix} 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \\ 0.17 & 0.20 & 0.17 & 0.17 & 0.20 & 0.17 \\ 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \\ 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \\ 0.17 & 0.20 & 0.17 & 0.17 & 0.20 & 0.17 \\ 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \end{bmatrix}, \quad (12)$$

$$S^{50} = \begin{bmatrix} 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \\ 0.17 & 0.20 & 0.17 & 0.17 & 0.20 & 0.17 \\ 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \\ 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \\ 0.17 & 0.20 & 0.17 & 0.17 & 0.20 & 0.17 \\ 0.15 & 0.17 & 0.15 & 0.15 & 0.17 & 0.15 \end{bmatrix}. \quad (13)$$

Fig. 2 Undirected graph

We can observe the impact of different k values on these weights between nodes at various distances (the various powers weights of S). The high power weights of S increase from 0 while the low power weights of S gradually decrease when increasing k value. Furthermore, the all weights of S tend to a fixed value when $k \geq 20$, which makes nodes features indistinguishable. In addition, these weights are relatively close when $k \geq 3$. This implies that SGC may suffer from over-smoothing when $k \geq 3$. Limited to the over-smoothing, SGC has difficulty in capturing many nodes features and exploring the global graph structure.

The overall architecture

In SGC, we improve the learning ability by increasing the scale of receptive field. The analysis in Sect. 3.2 shows that SGC fails to further enlarge the receptive field due to the over-smoothing of SGC with larger k value ($k \geq 3$). This hurts the classification performance on graph learning tasks. By expanding Eq. (6), we find that S^k includes a series of weights of adjacency matrices in the various powers. We hypothesize that the weights do not match and the mismatched weights are the key factor of the over-smoothing. Therefore, we can effectively tackle the over-smoothing by adjusting the weights to matching values. Motivated by the above analyses, we propose to adjust the weights by constructing a new adjacency matrix. Based on the adjacency matrix, we develop our AdjMix architecture (Fig. 1). In our AdjMix, we first construct an adjacency matrix with attention mechanism to adjust the weights. In addition, we design a simple and attentional graph convolution model to capture the features information of neighbor nodes that are k -hops away. Finally, we apply a softmax classifier to classify each node.

Simple and attentional graph convolution layer

In our simple and attentional graph convolution layer, node representations are updated in four steps, i.e. multi-scale adjacency matrix with attention mechanism, multi-scale

feature propagation, multi-scale linear transformation, and nonlinear activation. We now introduce each step in detail.

Multi-scale adjacency matrix with attention mechanism. In GCNs, the adjacency matrix A is used to denote the edge weights between nodes that are one hop away. However, the information propagation of graph is inadequate because of its inability to denote the edge weights between nodes that are k -hops ($k > 1$) away. To address the limits, we mix various powers of the A into a single multi-scale adjacency matrix with attention mechanism.

$$\check{S}_{\text{adj}} = \alpha_0 S_{\text{adj}}^0 + \alpha_1 S_{\text{adj}}^1 + \cdots + \alpha_k S_{\text{adj}}^k, \quad (14)$$

where $\check{S}_{\text{adj}}$ is the multi-scale adjacency matrix with attention mechanism, and S_{adj}^1 is a normalized adjacency matrix without added self-loops, with $S_{\text{adj}}^1 = S_{\text{adj}} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. Here S_{adj}^k is the k power of S_{adj} , and D is the degree matrix of A , with $D_{ii} = \sum_j A_{ij}$. We consider node's own feature via S_{adj}^0 , with $S_{\text{adj}}^0 = I$. In addition, we utilize the various powers of the S_{adj}^1 (such as S_{adj}^k) to obtain feature information from all nodes that are k -hops away in the graph. By applying a series of attention scores $\alpha_0, \alpha_1, \dots, \alpha_k$, we can flexibly adjust the weights of these adjacent powers of S_{adj} to avoid the concerns of over-smoothing.

Theorem 1 *Multi-scale adjacency matrix with attention mechanism is an operation of permutation invariance.*

Proof Let $S_{\text{adj}} \in \mathbb{R}^{n \times n}$ (n nodes), then $S_{\text{adj}}^0 \in \mathbb{R}^{n \times n}$, $S_{\text{adj}}^1, S_{\text{adj}}^2, \dots, S_{\text{adj}}^k \in \mathbb{R}^{n \times n}$, and $\check{S}_{\text{adj}} = \alpha_0 S_{\text{adj}}^0 + \alpha_1 S_{\text{adj}}^1 + \cdots + \alpha_k S_{\text{adj}}^k \in \mathbb{R}^{n \times n}$. Because our multi-scale adjacency matrix is element-wise operation, the spatial location of $\check{S}_{\text{adj}}$ is permutation invariant.

Multi-scale feature propagation. In order to aggregate various neighboring features, we design a multi-scale feature propagation that is what distinguishes our convolution from these convolutions of GCNs and SGC. The node feature matrix X is updated along these neighboring nodes at different scales in Eq. (15):

$$\bar{H} = \check{S}_{\text{adj}} X. \quad (15)$$

Intuitively, this step captures multi-scale feature information while keeping larger receptive field. Furthermore, we try to adjust the contributions of nodes features at different scales by applying a series of attention scores for performance improvement.

Multi-scale linear transformation and nonlinear activation. After the multi-scale feature propagation, our convolutional layer is identical to a multi-layer perceptron (MLP). We

use a learned weight matrix W to conduct multi-scale linear transformation. Finally, a nonlinear activation function such as ReLU is applied before outputting feature representation H . We conclude that the feature representation is updated.

$$H = \text{ReLU}(\bar{H} W). \quad (16)$$

Based on Eqs. (15) and (16), our simple and attentional graph convolution is defined as:

$$H = \text{ReLU}(\check{S}_{\text{adj}} X W). \quad (17)$$

Our algorithm is summarized in Algorithm 1. Our convolution model is used to learn k -hops nodes features and global graph structure by aggregating neighborhoods information as described in Eq. (17). In GCNs, the convolution suffers from inability in capturing nodes feature with different neighbors. However, our convolution can explore the interaction of neighboring nodes that are k -hops away. In SGC, the convolution fails to adjust the contributions of nodes features with different neighbors. In our convolution model, we can consider and adjust the contributions by a series of attention scores. In addition, limited to the over-smoothing of GCNs and SGC models, the receptive field of the models is narrow. This limits the expressive power. We adjust the weights to address the over-smoothing via multi-scale adjacency matrix with attention mechanism. Our model can improve the expressive power and learn global graph topology. However, AdjMix needs to set different k values (the highest power of S_{adj}) to achieve the best performance according to different datasets, compared to these baselines such as GCNs and SGC.

Algorithm 1 Simple and attentional graph convolution

Inputs: a normalized adjacency matrix without added self-loops S_{adj} , input node feature matrix X , a learned weight matrix W .

```

1: for  $j = 0$  to  $k$  do
2:    $S_{\text{adj}}^j = S_{\text{adj}}^j$ .
3: end for
4: Calculate multi-scale adjacency matrix with attention mechanism  $\check{S}_{\text{adj}}$  according to Eq. (14).
5:  $\bar{H} = \check{S}_{\text{adj}} X$ .
6:  $H = \text{ReLU}(\bar{H} W) = \text{ReLU}(\check{S}_{\text{adj}} X W)$ .
7: return  $H$ 

```

Output layer

In output layer, following Kipf and Welling [21], we predict the label of nodes using a softmax classifier and adopt the loss function from GCNs. The class prediction Y_{AdjMix} (i.e.

Table 1 Dataset statistics

Dataset	Pubmed	Citeseer	Cora
# Classes	3	6	7
# Nodes	19,717	3327	2708
# Features	500	3703	1433
# Edges	44,338	4732	5429
Label Rate	0.003	0.036	0.052

AdjMix model) takes the following form:

$$Y_{\text{AdjMix}} = \text{softmax}(H) = \text{softmax}(\text{ReLU}(\check{S}_{\text{adj}}XW)). \quad (18)$$

Complexity analysis and parameters comparison

Similar to SGC, we can regard $\bar{H}$ as a fixed feature extraction since the computation of $\bar{H} = \check{S}_{\text{adj}}X$ requires no weight. By precomputing the fixed feature extraction $\bar{H}$, the computation of the proposed model is very efficient. Let $A \in \mathbb{R}^{n \times n}$, $X \in \mathbb{R}^{n \times c_0}$, and $W \in \mathbb{R}^{c_0 \times c_1}$ (c_1 filters). As introduced in Sect. 3.4, our $\check{S}_{\text{adj}} \in \mathbb{R}^{n \times n}$, $\bar{H} \in \mathbb{R}^{n \times c_0}$, and $Y_{\text{AdjMix}} = \text{softmax}(\text{ReLU}(\bar{H}W)) \in \mathbb{R}^{n \times c_1}$, where c_1 denotes the number of classes. Because $\check{S}_{\text{adj}}$ is usually a sparse matrix with m non-zero entries, the time complexity and parameters of AdjMix are $O(m \times c_0 \times c_1)$ and $O(c_0 \times c_1)$, respectively. Compared to GCNs, our AdjMix performs better in terms of the complexity and parameters.

Experiments

We evaluate the benefits of AdjMix against a number of state-of-the-art models, with the goal of answering the following research questions:

Q1 Can AdjMix address the concerns of the over-smoothing when capturing the features information at long distances?

Q2 How does AdjMix compare to the state-of-the-art models on semi-supervised node classification tasks?

Q3 How does AdjMix compare to GCNs and other models in terms of the complexity and parameters on all datasets?

Datasets. To evaluate the effectiveness of AdjMix, we conduct experiments on citation network datasets chosen from benchmarks commonly used on semi-supervised node learning tasks. We use the datasets on public splits [21,43] of Pubmed, Citeseer, and Cora [34], which are summarized in Table 1.

Model configurations. We implement our model using Adam [20] optimizer in TensorFlow [1]. We now report the optimized hyperparameters to achieve the best performance on different datasets. On Cora, we set the learning rate to

0.0003 and dropout to 0.95 for improving the performance in terms of stability and test accuracy. On other datasets, we set the learning rate to 0.01 and dropout to 0 due to the very stable results. In addition, we apply 0.005, 0.000825, 0.0004 L_2 regularization factor and train 240, 1,500, 48,000 epochs on Pubmed, Citeseer, and Cora, respectively. We optimize the highest power (k power) of the normalized adjacency matrix S_{adj} according to prediction performance. We set k to 21, k to 4, k to 8 on Pubmed, Citeseer, and Cora, respectively. We tune these attention scores $\alpha_0, \alpha_1, \dots, \alpha_k$ to suitable values, with $\alpha_0 = 2.7, \alpha_1 = 1.5, \alpha_2 = 1.05, \alpha_3 = 0.85, \alpha_4 = 0.85, \alpha_5 = 0.8, \alpha_6 = 0.75, \alpha_7 = 0.7, \alpha_{12} = 0.6, \alpha_{13} = 0.6, \alpha_{14} = 0.55, \alpha_{15} = 0.55, \alpha_{16} = 0.55, \alpha_{17} = 0.5, \alpha_{18} = 0.5, \alpha_{19} = 0.5, \alpha_{20} = 0.45, \alpha_{21} = 0.45$, and other $\alpha = 0.65$ on Pubmed, with $\alpha_0 = 1.6, \alpha_1 = 1.3, \alpha_2 = 1.12, \alpha_3 = 0.85$, and $\alpha_4 = 0.8$ on Citeseer, and with $\alpha_0 = 1.76, \alpha_1 = 1.65, \alpha_2 = 1.155, \alpha_3 = 0.935, \alpha_4 = 0.935, \alpha_5 = 0.88, \alpha_6 = 0.825, \alpha_7 = 0.77$, and $\alpha_8 = 0.715$ on Cora. The other parameters of the proposed model are consistent with GCNs.

Baselines and experimental setting

In our experiments, we compare AdjMix with various state-of-the-art baselines, including Planetoid [43], gated graph neural networks (GGNN) [28], graph convolutional networks for fingerprint (GCN-FP) [11], message passing neural networks (MPNN) [12], diffusion-convolutional neural networks (DCNN) [4], graph attention networks (GAT) [38], Graph sample and aggregate (GraphSAGE) [14], graph convolutional networks (GCNs) [21], adaptive lanczos network (AdaLNet) [29], Lanczos network (LNet) [29], graph isomorphism network (GIN) [42], deep graph InfoMax (DGI) [39], learned MixHop (MixHop-learn) [3], and simple graph convolution (SGC) [40].

Analysis of over-smoothing

To address Q1, we conduct a variety of numerical experiments in Fig. 3. In Fig. 3c–e, by increasing the power of the S_{adj} , the weights are distinguishable while obtaining more nodes features. It is observed from Fig. 3f, g that further increasing the power of the S_{adj} makes the weights more close and the nodes features more indistinguishable. This naturally leads to over-smoothing. As described in Sect. 3.3, a main factor of the over-smoothing is the mismatched weights. Therefore, we propose multi-scale adjacency matrix to adjust these weights to appropriate values. We observe that these weights of the proposed multi-scale adjacency matrix are distinguishable. Namely, the features of different neighboring nodes in the proposed multi-scale adjacency matrix are distinguishable. This implies that our model can effectively avoid the concerns of over-smoothing when capturing the features information of neighboring nodes at long distances.

Fig. 3 Our multi-scale adjacency matrix with four power in Fig. 2. We summarize these results based on $\tilde{S}_{adj} = \alpha_0 S_{adj}^0 + \alpha_1 S_{adj}^1 + \alpha_2 S_{adj}^2 + \alpha_3 S_{adj}^3 + \alpha_4 S_{adj}^4$, with $\alpha_0 = 1.6$, $\alpha_1 = 1.3$, $\alpha_2 = 1.12$, $\alpha_3 = 0.85$, and $\alpha_4 = 0.8$

Table 2 Classification results in percent

Method	Pubmed	Citeseer	Cora
Planetoid [43]	77.2	64.7	75.7
GCN-FP [11]	76.0 ± 0.7	61.5 ± 0.9	74.6 ± 0.7
MPNN [12]	75.6 ± 1.0	64.0 ± 1.9	78.0 ± 1.1
GGNN [28]	75.8 ± 0.9	64.6 ± 1.3	77.6 ± 1.7
DCNN [4]	76.8 ± 0.8	69.4 ± 1.3	79.7 ± 0.8
GraphSAGE [14]	76.8 ± 0.6	67.2 ± 1.0	74.5 ± 0.8
GCNs [21]	79.0	70.3	81.5
GAT [38]	79.0 ± 0.3	72.5 ± 0.7	83.0 ± 0.7
GIN [42]	77.0 ± 1.2	66.1 ± 0.9	77.6 ± 1.1
AdaLNet [29]	78.1 ± 0.4	68.7 ± 1.0	80.4 ± 1.1
LNet [29]	78.3 ± 0.3	66.2 ± 1.9	79.5 ± 1.8
DGI [39]	76.8 ± 0.6	71.8 ± 0.7	82.3 ± 0.6
MixHop-learn [3]	80.8 ± 0.58	71.4 ± 0.81	81.9 ± 0.40
SGC [40]	78.9 ± 0.0	71.9 ± 0.1	81.0 ± 0.0
Adjmix-2 (ours)	79.5 ± 0.0	72.3 ± 0.0	81.9 ± 0.1
Adjmix (ours)	80.8 ± 0.0	73.5 ± 0.0	83.6 ± 0.1

Adjmix-2 is a variant of Adjmix where the highest power of the proposed multi-scale adjacency matrix is 2. We list the results that are averaged over 10 runs

Therefore, by increasing the powers of the adjacency matrix, our model can enlarge the receptive field and improve the expressive power of learning graph representations.

Results for node classification

Table 2 compares the performance of AdjMix to these state-of-the-art node classification baselines in terms of classification accuracy and stability on citation networks. These results provide positive answers to question Q2. We observe that our AdjMix obtains the highest performance including classification accuracy and stability among all state-of-the-art approaches on Pubmed, Citeseer, and Cora, improves over GCNs [21] by 2.3%, 4.6%, and 2.6%, improves over AdaLNet [29] by 3.4%, 7.0%, and 4.0%, and improves over SGC [40] by 2.4%, 2.2%, and 3.2% respectively. Our AdjMix achieves similar stability while significantly improving classification results compared to SGC [40], and outperforms other baselines by a large margin in terms of stability. Interestingly, our simplified model variant, Adjmix-2, outperforms SGC [40] by a large margin on all datasets. This is because our Adjmix-2 can adjust the weights of adjacency matrices in the different powers to matching values. These results demonstrate the effectiveness of the proposed models for adjusting the weights.

Figure 4 shows the influence of model width and node's own feature on all datasets. We observe that the accuracy of overall trend in our AdjMix without AS and AdjMix without AS and S_{adj}^0 models improves as model goes wider (increases k) until the width (k) of 21, 4, and 8 on Pubmed, Citeseer, and Cora, respectively. By further increasing the model width, the accuracy of our models begins to decrease. This is because our models with wider structure may mix the features from

Fig. 4 Influence of model width (the highest power of S_{adj} , i.e. k) and node's own feature. AdjMix without AS denotes our AdjMix model without these attention scores $\alpha_0, \alpha_1, \dots, \alpha_k$, then $\tilde{S}_{adj} = S_{adj}^0 + S_{adj}^1 +$

$\dots + S_{adj}^k$. AdjMix without AS and S_{adj}^0 denotes our AdjMix model without these attention scores and S_{adj}^0 , then $\tilde{S}_{adj} = S_{adj}^1 + S_{adj}^2 + \dots + S_{adj}^k$

Table 3 Comparison of parameters and complexity

Method	Parameters	Complexity
GCNs [21]	$O(c_0 \times 16)$	$O(m \times c_0 \times 16)$
SGC [40]	$O(c_0 \times 3)$ (Pubmed)	$O(m \times c_0 \times 3)$ (Pubmed)
	$O(c_0 \times 6)$ (Citeseer)	$O(m \times c_0 \times 6)$ (Citeseer)
	$O(c_0 \times 7)$ (Cora)	$O(m \times c_0 \times 7)$ (Cora)
	$O(c_0 \times 60)$	$O(2 \times m \times c_0 \times 20)$
MixHop-learn [3]	$O(c_0 \times 60)$	$O(2 \times m \times c_0 \times 20)$
Adjmix-2 (ours)	$O(c_0 \times 3)$ (Pubmed)	$O(m \times c_0 \times 3)$ (Pubmed)
	$O(c_0 \times 6)$ (Citeseer)	$O(m \times c_0 \times 6)$ (Citeseer)
	$O(c_0 \times 7)$ (Cora)	$O(m \times c_0 \times 7)$ (Cora)
	$O(c_0 \times 60)$	$O(2 \times m \times c_0 \times 20)$
Adjmix (ours)	$O(c_0 \times 3)$ (Pubmed)	$O(m \times c_0 \times 3)$ (Pubmed)
	$O(c_0 \times 6)$ (Citeseer)	$O(m \times c_0 \times 6)$ (Citeseer)
	$O(c_0 \times 7)$ (Cora)	$O(m \times c_0 \times 7)$ (Cora)
	$O(c_0 \times 60)$	$O(2 \times m \times c_0 \times 20)$

various clusters. Compared to AdjMix without AS model, AdjMix without AS and S_{adj}^0 model improves performance at different model widths on all datasets. This demonstrates the importance of node's own feature for designing model. These comparison results show the best model width and prove the benefits of own node features to performance improvement.

Analysis of parameters and complexity

As explained in He et al. [16], the actual running time is sensitive to hardware and implementations. We use the theoretical time complexity to show the complexity, rather than the actual running time, following Liu et al. [30]. To support answering Q3, we compare our methods with these state-of-the-art node classification baselines in terms of parameters and complexity on citation networks. As described in Sect. 3.6, we list the parameters and complexity in Table 3. Interestingly, in terms of parameters and complexity, the proposed models achieve as good performance as SGC [40], and perform better compared to other baselines. These results demonstrate the contribution of designing one-layer model.

Based on Tables 2 and 3, we conclude that the proposed models outperform SGC [40] by a large margin while main-

taining same parameters and complexity, and they achieve state-of-the-art performance in terms of accuracy, parameters, and complexity, compared to other baselines. These results show the significant advantages of the proposed models.

Conclusion

In this paper, we propose a simple and attentional graph convolutional network architecture, AdjMix, for semi-supervised node learning tasks. Notably, our AdjMix can adjust the weights of adjacency matrices in the various powers to address the over-smoothing of SGC and GCNs, thus capturing more nodes features information by taking wider model. By precomputing the fixed feature extraction $\tilde{H}$, our AdjMix achieves as good performance as SGC in terms of parameters and complexity. Experiments on node classification benchmarks show the superiority of capturing the node features and entire graph structure. In the future, we would design a learned network to automatically adjust these weights to matching values and apply the proposed model to more graph learning tasks. Especially, we will study how to apply the proposed model to directed graphs.

Acknowledgements This work was partly supported by the Guangdong Provincial Key Laboratory of Intellectual Property and Big Data (2018B030322016), the National Natural Science Foundation of China (U1701266), Special Projects for Key Fields in Higher Education of Guangdong, China (2020ZDZX3077, 2021ZDZX1042), the Science and Technology Project of Guangzhou City (202002020035), the Characteristic Innovation Projects of Ordinary Universities in Guangdong Province, China (2020KTSCX212, 2019KTSCX245), the Projects of South China Institute of Software Engineering of Guangzhou University (ky202012).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abadi M, Agarwal A, Barham P, Brevdo E (2016) Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint. [arXiv:1603.04467](https://arxiv.org/abs/1603.04467)
- Abu-El-Hajja S, Kapoor A, Perozzi B, Lee J (2019) N-gcn: multi-scale graph convolution for semi-supervised node classification. In: UAI, pp 1–9
- Abu-El-Hajja S, Perozzi B, Kapoor A, Harutyunyan H, Alipourfard N, Lerman K, Steeg G, Galstyan A (2019) Mixhop: higher-order graph convolution architectures via sparsified neighborhood mixing. In: ICML, pp 21–29
- Atwood J, Towsley D (2016) Diffusion-convolutional neural networks. In: NIPS, pp 1993–2001
- Battaglia P, Hamrick J, Bapst V (2018) Relational inductive biases, deep learning, and graph networks. arXiv preprint. [arXiv:1806.01261](https://arxiv.org/abs/1806.01261)
- Bruna J, Zaremba W, Szlam A, LeCun Y (2014) Spectral networks and locally connected networks on graphs. In: ICLR, pp 1–14
- Chen J, Ma T, Xiao C (2018) Fastgcn: fast learning with graph convolutional networks via importance sampling. In: ICLR, pp 1–15
- Chen L, Wu L, Hong R, Zhang K, Wang M (2020) Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach. In: AAAI, pp 27–34
- Chiang H, Chen M, Huang Y (2019) Wavelet-based EEG processing for epilepsy detection using fuzzy entropy and associative petri net. IEEE Access 7:103255–103262
- Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS, pp 3844–3852
- Duvenaud D, Maclaurin D, Aguilera-Iparraguirre J, Gómez-Bombarelli R, Hirzel T (2015) Convolutional networks on graphs for learning molecular fingerprints. In: NIPS, pp 2224–2232
- Gilmer J, Schoenholz S, Riley P, Vinyals O, Dahl G (2017) Neural message passing for quantum chemistry. In: ICML, pp 1263–1272
- Guo S, Lin Y, Feng N, Song C, Wan H (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: AAAI, pp 922–929
- Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: NIPS, pp 1024–1034
- Hariharan B, Arbeláez P, Girshick R, Malik J (2015) Hypercolumns for object segmentation and fine-grained localization. In: CVPR, pp 447–456
- He K, Sun J (2015) Convolutional neural networks at constrained time cost. In: CVPR, pp 5353–5360
- Islas M, Rubio J, Muñoz S, Ochoa G, Pacheco J, Meda-Campaña J, Mujica-Vargas D, Aguilar-Ibañez C, Gutierrez G, Zacarias A (2021) A fuzzy logic model for hourly electrical power demand modeling. Electronics 10(4):448
- de Jesús Rubio J, Lugofer E, Pieper J, Cruz P, Martinez D, Ochoa G, Islas M, Garcia E (2021) Adapting h-infinity controller for the desired reference tracking of the sphere position in the maglev process. Inf Sci 569:669–686
- Kampffmeyer M, Chen Y, Liang X, Wang H, Zhang Y, Xing E (2019) Rethinking knowledge graph propagation for zero-shot learning. In: CVPR, pp 11487–11496
- Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: ICLR, pp 1–15
- Kipf T, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: ICLR, pp 1–14
- Klicpera J, Bojchevski A, Günnemann S (2019) Predict then propagate: graph neural networks meet personalized pagerank. In: ICLR, pp 1–15
- Lee J, Rossi R, Kim S, Ahmed N, Koh E (2019) Attention models in graphs: a survey. ACM Trans Knowl Discov Data (TKDD) 13(6):1–25
- Lei F, Liu X, Dai Q, Ling B, Zhao H, Liu Y (2020) Hybrid low-order and higher-order graph convolutional networks. Comput Intell Neurosci Vol.2020, 3283890:1–3283890:9
- Li J, Han Z, Cheng H, Su J, Wang P, Zhang J, Pan L (2019) Predicting path failure in time-evolving graphs. In: KDD, pp 1279–1289
- Li Q, Han Z, Wu X (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI, pp 3538–3545
- Li Q, Wu X, Liu H, Zhang X, Guan Z (2019) Label efficient semi-supervised learning via graph filtering. In: CVPR, pp 9582–9591
- Li Y, Tarlow D, Brockschmidt M, Zemel R (2015) Gated graph sequence neural networks. In: ICLR, pp 1–20
- Liao R, Zhao Z, Urtasun R, Zemel R (2019) Lanczosnet: multi-scale deep graph convolutional networks. In: ICLR, pp 1–18
- Liu X, Xia G, Lei F, Zhang Y, Chang S (2021) Higher-order graph convolutional networks with multi-scale neighborhood pooling for semi-supervised node classification. IEEE Access 9:31268–31275
- Luan S, Zhao M, Chang X, Precup D (2019) Break the ceiling: stronger multi-scale deep graph convolutional networks. In: NIPS, pp 10943–10953
- Meda-Campaña J (2018) On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs. IEEE Access 6:31968–31973
- de Rubio J (2021) Stability analysis of the modified Levenberg–Marquardt algorithm for the artificial neural network training. IEEE Trans Neural Netw Learn Syst 32(8):3510–3524
- Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–93
- Soriano L, Zamora E, Vazquez-Nicolas J, Hernández G, Madrigal J, Balderas D (2020) Pd control compensation based on a cascade

- neural network applied to a robot manipulator. *Front Neurorobot* Vol.14, 577749:1–577749:9
36. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: *CVPR*, pp 1–9
 37. Thekumparampil K, Wang C, Oh S, Li L (2018) Attention-based graph neural network for semi-supervised learning. *arXiv preprint. arXiv:1803.03735*
 38. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: *ICLR*, pp 1–12
 39. Velickovic P, Fedus W, Hamilton W, Liò P, Bengio Y, Hjelm R (2019) Deep graph infomax. In: *ICLR*, pp 1–19
 40. Wu F, Zhang T, Souza J, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: *ICML*, pp 6861–6871
 41. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu P (2020) A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst* 32(1):4–24
 42. Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: *ICLR*, pp 1–17
 43. Yang Z, Cohen W, Salakhutdinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: *ICML*, pp 40–48
 44. Yu W, Qin Z (2020) Graph convolutional network for recommendation with low-pass collaborative filters. In: *ICML*, pp 10936–10945
 45. Zhang J, Shi X, Xie J, Ma H, King I, Yeung D (2018) Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In: *UAI*, pp 339–349
 46. Zhou J, Cui G, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2018) Graph neural networks: a review of methods and applications. *arXiv:1812.08434*

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.