Adaptive Simplified Graph Convolution

GDL 2023
Group id: 7
Project id: ASGC

Davide Casnici, David Alarcón, Aditya Ujeniya
{davide.casnici, david.alarcon, aditya.ujeniya}@usi.ch

Abstract

In the rapidly evolving field of graph neural networks (GNNs), Simple Graph Convolution (SGC) (Wu et al. [1]) has been a competitive model, yet it heavily relies on the principle of homophily, which is the tendency of nodes to form connections with similar nodes. In this investigation on Chanpuriya and Musco [2] research, we extend our scrutiny to the effectiveness of the Adaptive Simple Graph Convolution (ASGC) model in dealing with heterophily within networks. ASGC, an adaptive and scalable model, proves its mettle in handling both homophilous and heterophilous graph structures without being dependent on deep learning methods. Our aim was to recreate the original authors' experiments and confirm the performance of ASGC on both synthetic and real-world datasets. We found that it often matched the performance of deep learning models for nodes classification, while resulting in way faster approach. This study underscores ASGC's potential as an efficient, understandable, and effective alternative to more complex deep learning models. While there may be a compromise in terms of absolute accuracy (64.2% of accuracy achieved by the GCN with respect to 73.7% for the ASGC), the advantages of computational speed (133 seconds on average for the GCN with respect to 1 seconds for the ASGC) and model interpretability make ASGC an attractive choice for various network contexts. Our findings corroborate that deep learning, though often associated with superior performance, may not always be necessary, even when dealing with heterophilous structures. The results of our study not only validate the initial research but also underline ASGC as a promising tool for managing diverse network scenarios, due to its simplicity, scalability and performance.

1 Introduction

Graph data structures, which model complex relationships between various entities, are fundamental to diverse disciplines such as biology, sociology, and computer science. These structures have inspired the development of numerous machine learning models, particularly those tasked with node classification. Deep learning models like Graph Convolutional Networks (GCNs) (Kipf and Welling [3]) were traditionally employed for these tasks. Despite the successes of GCNs, their complexity and computational demands have stimulated the search for less resource-intensive alternatives.

Recently, a simpler model termed Simple Graph Convolution (SGC) (Wu et al. [1]) was proposed as a viable substitute. The strength of SGC lies in its acceleration of calculations by reducing the role of the graph to a feature extraction step, thereby avoiding the costly end-to-end computations that GCNs require. However, a notable limitation of SGC is its assumption of homophily—the principle that similar nodes are more likely to connect—which may not hold true for all graphs.

In their groundbreaking paper, "Simplified Graph Convolution with Heterophily", Sudhanshu Chanpuriya and Cameron Musco from the University of Massachusetts Amherst addressed this issue by introducing the Adaptive Simple Graph Convolution (ASGC) model [2]. Unlike SGC, ASGC adapts to both homophilous and heterophilous structures, providing a more flexible tool for dealing with diverse graph structure.

In this report, we seek to validate Chanpuriya and Musco's assertions about ASGC's efficacy in handling heterophilous networks and its competitive performance against more complex deep models. Our study involves a thorough replication of the ASGC model, the interpretation of our findings, and a discussion on the broader implications of the results. The report corroborates the authors' claims and emphasizes ASGC's potential as an effective, simpler alternative to complex deep learning models in graph machine learning tasks.

2 Related works

Graph learning models have seen numerous developments over the years. A foundational concept in this field is Message Passing Neural Networks (MPNNs), introduced by Gilmer et al. [4]. These models exploit the local network structure to make decisions but are computationally expensive due to their use of repeated nonlinear transformations.

A more streamlined model, Simple Graph Convolution (SGC), was proposed by Wu et al. [1]. SGC avoids the costly nonlinearities and end-to-end backpropagation of traditional GCNs. It acts as a 'low-pass' filter [5], smoothing out node features along the graph's edges. The graph's role in SGC is primarily for feature extraction, an approach which implies the homophily characteristic of the graph.

The study by Chanpuriya and Musco introduced a significant advancement in this field, termed the Adaptive Simple Graph Convolution (ASGC). The ASGC model extends the concept of SGC, allowing it to adapt to both homophilous and heterophilous structures.

Our investigation focuses on ASGC as it promises to be a fast, scalable, interpretable and competitive alternative to existing models like SGC and MPNN. This report emphasizes the distinctiveness of ASGC while drawing upon previous advancements in the field of graph learning. For a more comprehensive understanding of these methods, readers are recommended to refer to the cited literature.

3 Methodology

We approach the task of optimizing Simplified Graph Convolution (SGC) by implementing Chanpuriya and Musco's adaptive feature propagation mechanism. Unlike SGC's fixed feature propagation, the Adaptive SGC (ASGC) method is designed to fine-tune propagation based on specific feature and graph attributes.

3.1 Simple Graph Convolution

We start by describing the terminologies used in SGC and ASGC algorithm. We know that $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{n \times n}$ matrix denotes adjacency matrix of our graph. We also let **1** denote all-ones column vector, making **d**= $\boldsymbol{A}\mathbf{1}$ the vector with degree of nodes. The degree matrix $\boldsymbol{D}$ is a diagonal matrix of **d**. It is also required to normalise the adjacency matrix, as follows:

$$\boldsymbol{S} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$$

This $\boldsymbol{S}$ is used in SGC and ASGC to filter the raw feature vector and perform feature extraction.

The authors work is inspired by SGC, which performs feature smoothing over feature vector x as follows:

$$x_{SGC} = \tilde{\mathbf{S}}^{K}x$$

This performs smoothing on nodes across all graph edges. Here, $\tilde{\mathbf{S}}$ denotes normalized adjacency matrix with added self loop to each node and K is a hyperparameter to control the radius of filtering, i.e. number of hops between two nodes whose feature can influence other features in filtering process. [1]

3.2 Adaptive Simple Graph Convolution

The primary objective of ASGC is to generate a modified feature vector, x_{ASGC}, from the original raw feature vector, x, such that x_{ASGC} closely resembles x. This process involves multiplication of x by a learned polynomial of the normalized adjacency matrix, $\boldsymbol{S}$. The formulation of x_{ASGC} is as follows:

$$x_{\text{ASGC}} = \left(\sum_{k=0}^{K} \beta_k \boldsymbol{S}^k\right) \approx x$$

In the above equation, K is a hyperparameter that regulates the radius of feature propagation. The coefficients $\beta \in \mathbb{R}^{K+1}$ are determined by minimizing the approximation error in the least squares sense. Unlike SGC, we refrain from adding self-loops to $\boldsymbol{S}$ after considering the following criteria:

- **Different Focus**: ASGC focus more on the feature information coming from the neighboring nodes rather than the node itself. This could be beneficial in some scenarios where the local structure of the graph or the neighborhood relationships are more informative.
- **Avoiding Redundancy**: The ASGC method already consider the original node features in its transformation process, making the inclusion of self-loops redundant.
- **Preserving Sparsity**: Adding self-loops to a graph changes its structure and potentially makes the adjacency matrix denser [1]. In large graphs, preserving sparsity might be critical for computational feasibility.
- **Achieving Different Goals**: The ASGC aim to address a limitation or achieve a performance characteristic that differs from the goals of the SGC method, and the exclusion of self-loops could be a part of this approach.

ASGC (Adaptive Simplified Graph Convolutions) can be also be interpreted using spectral theory. It employs the graph Fourier transform of the feature x and interprets the minimization objective as fitting a K-degree polynomial over the graph's eigenvalues. The polynomial represents how the learned filter scales the component of the feature along the direction of the corresponding eigenvector. Due to the small K and regularization on the zeroth coefficient, the filter adapts to the feature, offering a simple yet effective alternative to more computationally intensive deep learning methods.

3.3 Construction of Krylov Matrix

We define a Krylov matrix $\boldsymbol{T} \in \mathbb{R}^{n \times (K+1)}$, generated by multiplying the raw feature vector $x \in \mathbb{R}^n$ by the normalized adjacency matrix $\boldsymbol{S}$ up to $K \in \mathbb{Z}_{>0}$ times:

$$\boldsymbol{T} = \left(\boldsymbol{S}^0 x, \boldsymbol{S}^1 x, \boldsymbol{S}^2 x, ..., \boldsymbol{S}^K x\right)$$

Here, the leftmost column of $\boldsymbol{T}$ corresponds to the raw feature vector x. Each subsequent column embodies the feature generated by propagating the feature vector to its left across the graph, i.e., by multiplying once by $\boldsymbol{S}$. [2]

3.4 Optimal Coefficient Determination

We produce a filtered version x_{ASGC} of the raw feature x by utilizing a linear combination of the columns of $\boldsymbol{T}$. The expression is $x_{\text{ASGC}} = \boldsymbol{T}\beta$

To determine the optimal coefficients $\beta \in \mathbb{R}^{K+1}$, we minimize a loss function that incorporates the difference between the transformed feature vector and the original feature vector, with an additional regularization term to prevent overfitting. The loss function is defined as:

$$\min_{\beta} \left(\|\boldsymbol{T}\beta - x\|_2^2 + (R\beta_0)^2\right)$$

where the first term represents the least square error between the transformed feature vector and the original feature vector. The term $R\beta_0^2$ acts as a regularization term specifically for β_0, aiming to control the influence of the original feature vector in the transformed version. The regularization strength is controlled by $R \in \mathbb{R}_{\geq 0}$. This regularization term is necessary to prevent the trivial case where $\beta_0 = 1$ and all other coefficients are zero, resulting in no transformation. The loss function is solved for the optimal coefficients β using linear least squares.

3.5 Method Overview and Computational Complexity

Algorithm 1 provides a pseudocode for the proposed ASGC method. The core computations in Algorithm 1 are the creation of the Krylov matrix $\boldsymbol{T}$ by multiplying x by $\boldsymbol{S}$ up to K times, and solving the linear least squares problem. The time complexity of the algorithm is $O(mK)$ for the first part, where m is the number of edges in the graph, and $O(nK^2)$ for the second part, where n is the number of nodes.

After the feature vectors are transformed using the ASGC method, these filtered features are used as input to a logistic regression classifier for node classification. The ASGC method is applied independently to all features in the feature matrix, which makes this process easily parallelizable across features.

3.6 Potential Limitations and Considerations

In theory, as the order K increases, the Krylov matrix $\boldsymbol{T}$ will provide a sufficiently high-rank basis such that x_{ASGC} can be arbitrarily close to x, even if the regularization parameter $\boldsymbol{R}$ is high. In this case, the performance of ASGC would approach that of using the raw feature. To mitigate this, a smaller regularization term for the remaining coefficients could be introduced.

The performance and limitations of ASGC and SGC (Simplified Graph Convolutions) on FSBM (Feature Stochastic Block Models) networks are theoretically confirmed. SGC preserves feature means under specific conditions, depending on the nature of the graph's eigenvalues. By contrast, ASGC preserves means while similarly reducing noise, with more flexible restrictions. Thus, despite differences in the way SGC and ASGC treat feature means, both methods significantly denoise the features, providing robust results even in complex network scenarios. [2]

```
Algorithm 1 Adaptive Simple Graph Convolution (ASGC) Filter
  Input: undirected graph $\boldsymbol{A} \in \{0,1\}^{n \times n}$, node feature $\boldsymbol{x} \in \mathbb{R}^n$,
         number of hops $K \in \mathbb{Z}_{>0}$, regularization strength $R \in \mathbb{R}_{\geq 0}$
  Output: ASGC-filtered feature $\boldsymbol{x}_{\text{ASGC}}$
  $\boldsymbol{D} \leftarrow \text{diag}(\boldsymbol{A}\mathbf{1})$ {degree matrix}
  $\boldsymbol{S} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ {normalized adjacency matrix}
  $\boldsymbol{T} \in \mathbb{R}^{n \times (K+1)} \leftarrow \mathbf{0}$ {$k$-step propagated features}
  $\boldsymbol{T}_0 \leftarrow \boldsymbol{x}$
  for $k = 1$ to $K$ do
    $\boldsymbol{T}_k \leftarrow \boldsymbol{S}\boldsymbol{T}_{k-1}$
  end for
  $\boldsymbol{R} \in \mathbb{R}^{K+1} \leftarrow (R, 0, 0, \ldots, 0)$
  $\beta \leftarrow$ least squares solution of $\binom{\boldsymbol{T}}{\boldsymbol{R}}\beta \approx \binom{\boldsymbol{x}}{0}$
  Return: $\boldsymbol{T}\beta$
```

Figure 1. Algorithm for ASGC method. The ASGC filter takes as input an undirected graph, a node feature, a specified number of hops, and a regularization strength. The algorithm then calculates the degree matrix, the normalized adjacency matrix, and k-step propagated features, before finally determining the optimal coefficient using least squares solution and returning the ASGC-filtered feature.

4 Motivating Example with Synthetic Dataset

In this section, we demonstrate the functionality of Adaptive Simple Graph Convolution (ASGC) in the context of homophilous and heterophilous networks. We will employ Feature Stochastic Block Models (FSBMs) to create synthetic networks for this purpose.

4.1 Feature Stochastic Block Models

A Stochastic Block Model (SBM) is a random graph model with a pre-specified number of nodes partitioned into communities, with intra and inter-community edge probabilities (Holland et al. [6]). In a Featured SBM (FSBM), each node additionally has a feature vector, represented as $x = f + \eta \in \mathbb{R}^n$, where $\eta \sim \mathcal{N}(0, \sigma^2 I)$ is zero-centered, isotropic Gaussian noise and $f = \sum_i \mu_i c_i$, with $\mu_1, \mu_2, \ldots, \mu_r \in \mathbb{R}$ representing the expected feature values of each community. (as per definition in [2])

To generate synthetic networks, we followed the parameters used in the original paper. We established two communities of equal size, denoted C_- and C_+, each with 500 nodes. The feature means for these communities were set to $\mu_- = -1$ and $\mu_+ = +1$ respectively, and we added Gaussian noise with variance $\sigma = 0.5$.

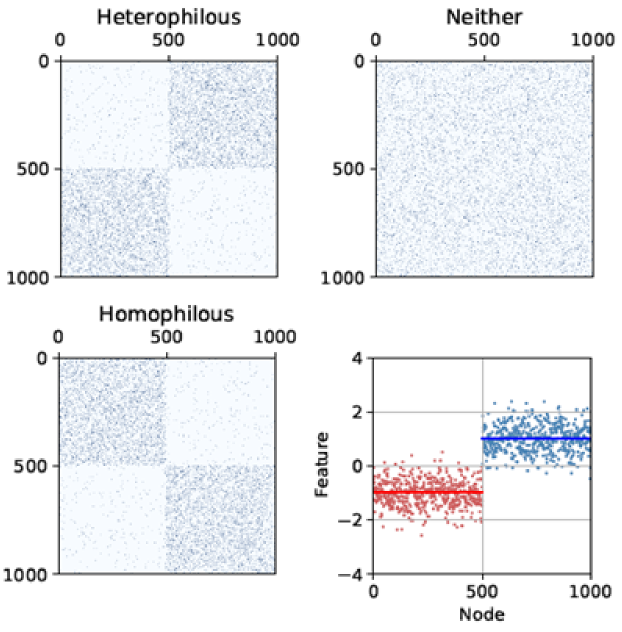

Figure 2. Synthetic data visualization. The top left denotes the adjacency matrix of a highly heterophilous graph. The bottom left shows an adjacency matrix of a highly homophilous graph. The bottom right figure displays feature values per node, illustrating the distinct mean feature values per community.

4.2 Results of SGC and ASGC on Synthetic Dataset

Both SGC and ASGC are employed to denoise the features of the generated synthetic graphs, using a number of hops, $K = 2$. Our findings indicate that these methods show substantial denoising effects, as demonstrated below:

The Mean Squared Error (MSE) and Sign Error metrics show that ASGC performs notably better than SGC on heterophilous graphs, but the performance gap decreases in the case of homophilous graphs.

These results in Figure 3 and 4 suggest that ASGC is more capable of handling a wide range of graphs with varying degrees of homophily or heterophily, as it is able to preserve the distinction between the communities and adhere to the respective community feature means.

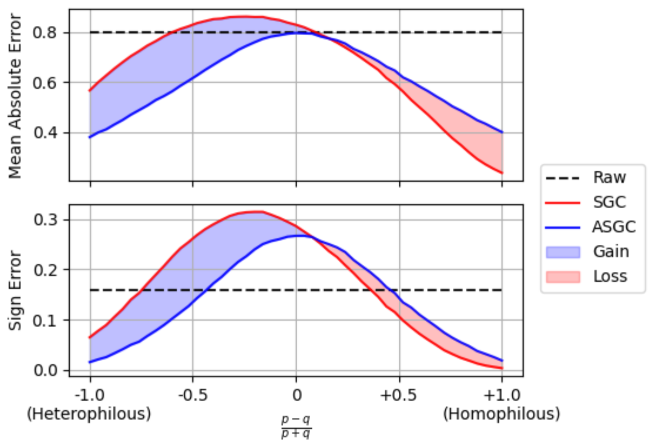

Figure 3. The denoising effects when applying SGC and ASGC to various types of graphs. 'Raw error' refers to the error when no filtering method is used. SGC and ASGC prove effective for graphs with pronounced homophily or heterophily.

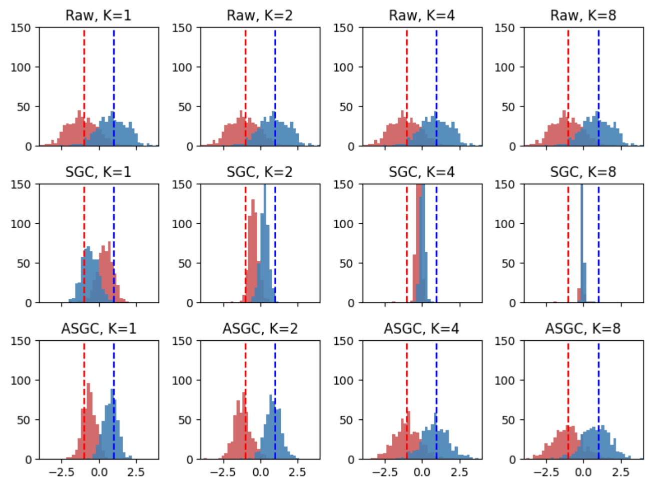

Figure 4. Distribution of feature values based on their true values. The 'Raw' label represents the distribution before filtering, while 'SGC' and 'ASGC' show the distribution after applying these filtering methods. Notably, SGC tends to blend the two communities, while ASGC maintains their distinction.

4.3 Spectral Decomposition

The Spectral Interpretation of ASGC is based on the eigendecomposition of the matrix $\boldsymbol{S}$ as $\boldsymbol{S} = \boldsymbol{Q}\text{diag}(\lambda)\boldsymbol{Q}^T$, where $\boldsymbol{Q}$ is an unitary matrix and $\text{diag}(\lambda)$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{S}$. The graph Fourier transform of the feature vector x is denoted as $\gamma = \boldsymbol{Q}^{-1}x$.

ASGC aims to minimize the norm of the residual in the least squares equation. By leveraging Parseval's Theorem and the orthogonality of $\boldsymbol{Q}$, the norm remains invariant under the graph Fourier transform, as mentioned in the original paper [2]:

$$\begin{aligned} \|x_{\text{ASGC}} - x\|^2 &= \|T\beta - x\|^2 \\ &= \left\|Q^T(T\beta - x)\right\|^2 \\ &= \left\|Q^{-1}(T\beta - x)\right\|^2 \end{aligned}$$

Each column of the matrix T corresponds to $S^i x$ for some power i, with $i \geq 0$. By further processing the equation, we express the minimization objective as:

$$\begin{aligned} \|x_{\text{ASGC}} - x\|^2 &= \left\|Q^{-1}T\beta - Q^{-1}x\right\|^2 \\ &= \|\text{diag}(\gamma)V_\lambda\beta - \gamma\|^2 \\ &= \|\text{diag}(\gamma)(V_\lambda\beta - 1)\|^2 \end{aligned}$$

where V_λ represents the Vandermonde matrix of the powers of the eigenvalues λ of S. Multiplying V_λ by the vector β yields the polynomial values with coefficients β, evaluated at the eigenvalues λ.

This interpretation clarifies that ASGC fits a polynomial of degree K to the graph's eigenvalues, with the target being an all-ones polynomial. The value of the polynomial at each eigenvalue represents how the learned filter scales the component of the feature x along the corresponding eigenvector direction of S. Achieving an all-ones polynomial corresponds to a do-nothing filter, indicating no filtering is applied. The least squares loss is weighted proportionately to the magnitude of this component at each eigenvalue.

As showed in Figure 4, using too high values for K (filtering steps) in ASGC, results in a do-nothing filtering effect mainly due to two main reasons:

1. **Limited eigenvalue information**: The Vandermonde matrix V_λ represents the powers of the eigenvalues λ of S up to K. However, as K becomes very large, it includes higher powers of the eigenvalues that may not significantly contribute to the filtering process. These higher powers can amplify noise and irrelevant information, leading to ineffective filtering.
2. **Regularization on the zeroth coefficient**: ASGC applies L2 regularization on the zeroth combination coefficient β_0 that multiplies the raw feature $S^0 x$. This regularization prevents the do-nothing filter from being the optimal solution. However, as K increases, the regularization effect diminishes. Consequently, the learned filter becomes less constrained and tends to approximate the do-nothing filter.

Therefore, selecting an appropriate value for K is crucial to balance the usage of eigenvalues information and regularization, ensuring the learned filter adapts to the relevant features of the graph while effectively removing the noise.

5 Experiments Implementation

Our implementation of the SGC and ASGC algorithms took inspiration from the original authors' code, which was meticulously optimized for both data structures and algorithmic structure. Our attempts at rewriting the code were aimed at enhancing its efficiency, yet the resultant version mirrored the original one closely due to its already high level of efficiency.

An ambitious endeavor was our effort to execute the SGC and ASGC algorithms on GPU using the TensorFlow framework. The primary focus here was the matrix multiplication parts of the algorithms. Despite successfully speeding up the operations, a significant decrease in accuracy was observed due to the sparse nature of the matrices involved in these computations. Therefore, we decided to continue with the CPU version of the implementation, as prioritizing accuracy over speed in this context yielded better overall results.

To conclude, the implementation of ASGC and SGC, although not significantly different from the authors' original implementation, provided us a profound understanding of the inner workings of these algorithms. It also highlighted the delicate trade-off between accuracy and computational speed, particularly in the context of sparse matrices in graph-based computations. We believe that this understanding can contribute to the development of further improvements or new approaches in the future.

5.1 Datasets

This study leveraged several datasets from the PyTorch Geometric library, encompassing a range of network types and contexts. Notably, these datasets also formed the foundation of the original research by Chanpuriya and Musco.

The chosen datasets cover a diverse array of contexts as [7], including e-commerce, scientific publications, online communities, and filmography, among others. In each case, nodes represent entities (goods [8] [9], publications [10] [11], web pages [12], or actors [13]), while edges denote relationships between these entities, such as co-purchase patterns, citation links, hyperlink connections, or co-occurrence on Wikipedia pages. [14]

Name	Type	Nodes	Edges	Features	Labels
Cora	HM	2,708	1,433	5,278	7
Citeseer	HM	3,327	4,552	3,703	6
PubMed	HM	19,717	44,324	500	3
Computers	HM	13,752	245,861	767	10
Photo	HM	7,650	119,081	745	8
Chameleon	HT	2,277	31,421	2,325	5
Squirrel	HT	5,201	198,493	2,089	5
Actor	HT	7,600	26,752	932	5
Cornell	HT	183	295	1,703	5
Texas	HT	183	280	1,703	5

Table 1. Dataset Information, HM refers to homophilous and HT to heterophilous.

As for the data preprocessing, we used the preprocessing techniques provided by the PyTorch Geometric library. It's important to note that all datasets were preprocessed in the same manner for both the SGC and ASGC methods to ensure a fair comparison. This preprocessing typically involved normalization and the transformation of categorical data into a form suitable for the models.

However, we encountered challenges in terms of computational requirements due to the high dimensionality of some datasets, which affected the time needed for hyperparameter tuning

5.2 Hyperparameters

We conducted experiments on each dataset by testing various combinations of hyperparameters for the SGC and ASGC methods. Specifically, we evaluated the methods using these specific hyperparameter values. Since our intention was to explore a wider range of hyperparameters than those suggested in the paper, we tested more combinations than the ones originally proposed. The hyperparameter values we experimented with are as follows:

1. The hyperparameter ***K***, which represents the number of filtering steps or hops, was set to a fixed range of values: [1, 2, 3, 4, 6, 8]. We performed iterations over different K values and selected the one that yielded the highest accuracy.

2. The hyperparameter ***R***, which represents the regularization strength, was explored using 13 log-spaced values ranging from $1e^{-6}$ to $1e^{6}$. We iterated over these 15 values and determined which R value effectively reduced the least squares (LS) problem in combination with the optimal K value. We indeed have tried of the possible combination of pairs and kept the best one.

5.3 Experimental setup

We conducted experiments on the datasets previously described using various models: logistic regression, logistic regression with SGC and ASGC filtering, multilayer perceptron, and graph convolutional neural network. Each method was evaluated through 10 trials to obtain a better understanding of the model's mean performance and standard deviation. In total, 100 experiments are done, with ten experiments performed for each of the ten datasets, as proposed in the original paper. The results of each method are stored in separate dictionaries for further analysis and evaluation. The mean test accuracy and the 95% confidence interval are printed for each model and dataset combination. We not only conducted experiments to evaluate the claim made in the original paper regarding the competitiveness of ASGC in terms of accuracy performance compared to deep methods. Additionally, we measured the evaluation time for each method and stored the results in dictionaries for further analysis and comparison, in order to prove also the claim regarding its velocity and scalability.

5.4 Computational requirements

We have run all the methods on the CPU to have a more fair comparison of the latter. The amount of RAM required to run the experiment is around 4.5GB and the amount taken by the whole notebook to run depends on the environment. Using Google Colab's environment, it needs around 8 hours to run everything.

6 Results

Our replication of the original authors' experiments confirmed that the Adaptive Simple Graph Convolution (ASGC) method competes well in terms of accuracy against deep learning approaches, which typically are more time and resources consuming.

The results were demonstrated via performance plots, which visually represent the classification accuracy across various datasets. We adhered to the original study's train/valid/test splits [7], i.e., 20/20/60 percent splits for heterophilous datasets and 2.5/2.5/95 percent splits for homophilous datasets [15]. Although the motivation for these split differences are not developed in the original paper, we believe this difference is due to the inherent characteristics and challenges of each type of dataset.

In the case of homophilous networks, where similar nodes are more likely to connect, the assumption is that even a small representative subset of the nodes (for training and validation) can yield a reasonable understanding of the entire graph due to the prevalence of similar characteristics among connected nodes. Thus, more of the data can be reserved for the testing set (95% in this case), allowing a more rigorous evaluation of the model's performance on unseen data.

In contrast, heterophilous networks, where dissimilar nodes tend to connect, present a greater challenge for graph neural networks. The relationships and features in these networks are often more complex and diverse, meaning a larger portion of the data is needed for the model to learn effectively. Consequently, larger training and validation sets (20% each) are required, with a relatively smaller portion left for the testing set (60% in this case).

This difference in data splits illustrates an important aspect of machine learning: dataset partitioning often needs to be tailored to the specifics of the data and the problem at hand. By adopting this approach, we also optimize the learning process considering the particularities of both homophilous and heterophilous datasets.

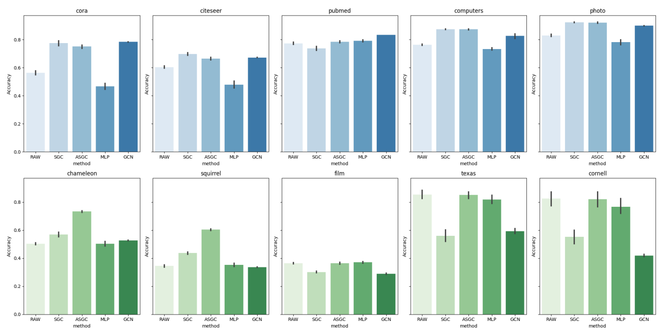

Figure 5. Models' accuracy with respect to each datasets. The blue plots show the performance on the homopihlous graphs, while the green ones show the performance on the heterophilous graphs. The barplots show the mean and the standard error among the ten trials we have done.

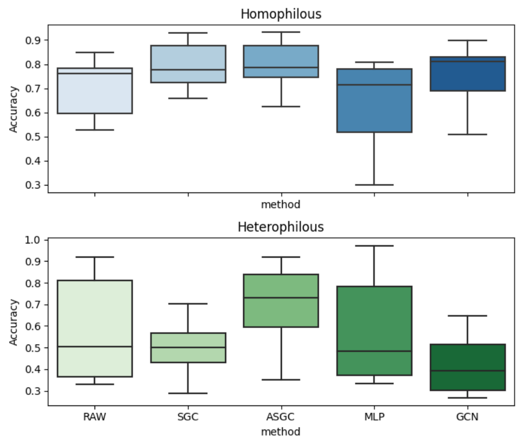

Figure 6. Boxplots representing the accuracy distributions of the ten trials we have done on each dataset, with respect to each method.

The performance box plot for the averaged ASGC method's accuracy accross datasets (Figure 6) exhibits that the ASGC method closely parallels, and in some instances, even surpasses the performance of the deep learning approaches. This trend is even clearer for heterophilous structures.

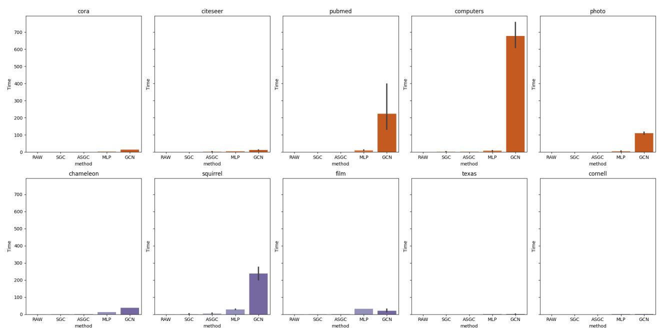

Figure 7. Models' consumed time with respect to each datasets. The purple plots show the time need on the heterophilious graphs, while the orange ones show the time taken by the methods on the homophilious graphs. The barplots show again the mean and the standard error among the ten trials we have done.

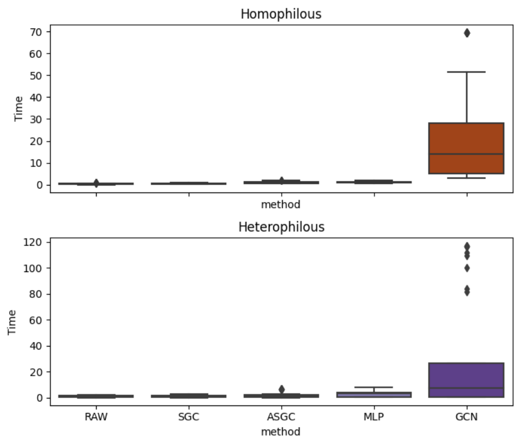

Figure 8. Boxplots representing the needed time distributions of the ten trials we have done on each dataset, with respect to each method.

Our results lend credence to the notion that ASGC, despite its relative simplicity, can deliver comparable performance to deep learning methods. Furthermore, it demonstrates the model's capacity to handle both homophilous and heterophilous graph structures effectively.

However, it's crucial to note that the advantage of the ASGC model isn't just about competitive performance. Equally significant is the reduced computational cost and enhanced interpretability offered by ASGC, presenting it as an appealing alternative to more complex and resource-intensive deep learning methods.

7 Discussion and conclusion

This study embarked on a comprehensive exploration of the Adaptive Simplified Graph Convolution (ASGC) method, a powerful, yet computationally efficient tool for graph representation learning. The salient feature of ASGC lies in its adaptability and robustness, enabling it to accommodate heterophilous network structures while preserving valuable feature means, a property not shared by its Simple Graph Convolution (SGC) counterpart.

Through methodological investigation and rigorous testing, we confirmed the better performance of ASGC, as initially proposed by the original authors. Our synthetic dataset, generated through Feature Stochastic Block Models (FSBMs), served as a test-bed to reveal the efficacy of ASGC, especially in its denoising capabilities. It was observed that, despite the complexity of the data, ASGC robustly maintained the feature means while significantly reducing noise.

An alternative interpretation of ASGC was also put forward based on spectral theory, further shedding light on the underlying mechanisms that allow the method to adapt to the features in a graph. Additionally, we expanded the scope of the original study by exploring heterophilous datasets, proving that the model could handle not only homophilous data but also graphs with more complex interconnections.

Despite ASGC showing promising performance and speed, it is important to note that its effectiveness heavily depends on hyperparameter tuning. This process can be time-consuming due to the need of carefully select the optimal hyperparameters, which are really problem dependent. Additionally, it would be beneficial to explore the performance of ASGC on a big variety of real world datasets to obtain a more comprehensive understanding and empirical evidence of some standard hyperparameters values, given the unique characteristics of each problem. An interesting future work we could do, is to inquire about the usage of a stochastic hyperparameter optimization to see if the process' time need could be improved without performance loss.

Furthermore, the original study used different split percentages for heterophilous and homophilous datasets, suggesting the need for further research to establish standardized norms for dataset division based on their inherent characteristics. This would contribute to a more consistent and reliable evaluation of ASGC and other methods in the future.

In future studies, we plan to test ASGC on a wider variety of datasets, not only to further validate its performance but also to potentially unearth additional nuances in its functionality. It is hoped that with such explorations, the ASGC method can be tuned to perform optimally across a broader range of network data.

In conclusion, our study substantiates the original authors' claim about ASGC's robust and efficient performance in graph representation learning. We believe this contributes to the ongoing discourse on graph neural networks, offering a promising avenue for handling large scale graph data with enhanced computational efficiency and improved adaptability.

References

[1] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr. au2, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks, 2019.

[2] Sudhanshu Chanpuriya and Cameron Musco. Simplified graph convolution with heterophily, 2022.

[3] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[4] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[5] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters, 2019.

[6] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5 (2):109–137, 1983. ISSN 0378-8733. doi: https://doi.org/10.1016/0378-8733(83)90021-7. URL `https://www.sciencedirect.com/science/article/pii/0378873383900217`.

[7] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network, 2021.

[8] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008. doi: 10.1609/aimag.v29i3.2157. URL `https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2157`.

[9] Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. 2012.

[10] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes, 2015.

[11] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2019.

[12] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118(1): 69–113, 2000. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(00)00004-7. URL `https://www.sciencedirect.com/science/article/pii/S0004370200000047`.

[13] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 817–826, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557109. URL `https://doi.org/10.1145/1557019.1557109`.

[14] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding, 2021.

[15] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks, 2020.