Label-informed Graph Structure Learning for Node Classification

Liping Wang[1,2,*], Fenyu Hu[1,2,*], Shu Wu[1,2,3,†], and Liang Wang[1,2]
[1]Center for Research on Intelligent Perception and Computing, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Artificial Intelligence Research, Chinese Academy of Sciences
wangliping2019@ia.ac.cn,fenyu.hu@cripac.ia.ac.cn
{shu.wu,wangliang}@nlpr.ia.ac.cn

ABSTRACT

Graph Neural Networks (GNNs) have achieved great success among various domains. Nevertheless, most GNN methods are sensitive to the quality of graph structures. To tackle this problem, some studies exploit different graph structure learning strategies to refine the original graph structure. However, these methods only consider feature information while ignoring available label information. In this paper, we propose a novel label-informed graph structure learning framework which incorporates label information explicitly through a class transition matrix. We conduct extensive experiments on seven node classification benchmark datasets and the results show that our method outperforms or matches the state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies → Neural networks**.

KEYWORDS

graph neural network, structure learning, node classification

ACM Reference Format:
Liping Wang, Fenyu Hu, Shu Wu, and Liang Wang. 2021. Label-informed Graph Structure Learning for Node Classification. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM ’21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482129

1 INTRODUCTION

As a powerful tool of analyzing graph-structured data, Graph Neural Networks (GNNs) have recently demonstrated great success across various domains, including node classification [4, 5, 8], link prediction [15], recommendation systems [14], etc. Despite GNNs’ powerful ability in learning expressive node embeddings, these methods are sensitive to the quality of graph structures.

Recently, some studies [1, 3] attempt to boost the performance of GNNs through jointly learning a denoised graph structure and node embeddings. These works can be unified under Graph Structure Learning (GSL) [17]. The key rationale behind these works is to remove the suspicious or add a potential edge between two nodes according to the distance or similarity between their embeddings. For example, IDGL [1] first computes weighted cosine similarity between node embeddings. Then, this similarity is used to refine the original graph structure. Lastly, the optimal graph structure can be acquired by directly optimizing downstream tasks such as node classification or link prediction.

However, all of the existing GSL methods ignore available label information. A potential edge between two nodes is added to the graph if they have similar features or embeddings regardless of their labels. These added edges may contain noise and be harmful to the performance. Take a citation network as an example, two papers focusing on the same problem adopt totally different approaches, thus they should be classified into two different categories. Since these two papers co-cite some classic papers solving the same problem, they have some common neighbors in the citation network. Accordingly, the distance between their embeddings learned by GNNs is relatively short. In this case, existing GSL methods tend to add an edge between them, misleading the model to classify them into the same category.

To overcome this limitation, we propose a label-informed graph structure learning framework (LGS) which *incorporates label information into graph structure learning explicitly*. Specifically, we employ a class transition matrix, where each element represents the probability of an edge between nodes of two classes. Different from existing GSL methods, we consider feature similarity and class transition probability at the same time. Intuitively, for two nodes with very similar features, if the transition probability between their corresponding classes is very low, it is still not appropriate to add an edge between them.

The main contributions of this work are summarized as follows:

- Apart from feature similarity, we explicitly consider label information in graph structure learning. We introduce a novel iterative graph structure learning framework for node classification.
- We conduct extensive experiments on both homophily and heterophily graph datasets, demonstrating the superiority of our method.

[*]The first two authors made equal contribution to this work.
[†]To whom correspondence should be addressed.

CIKM ’21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482129

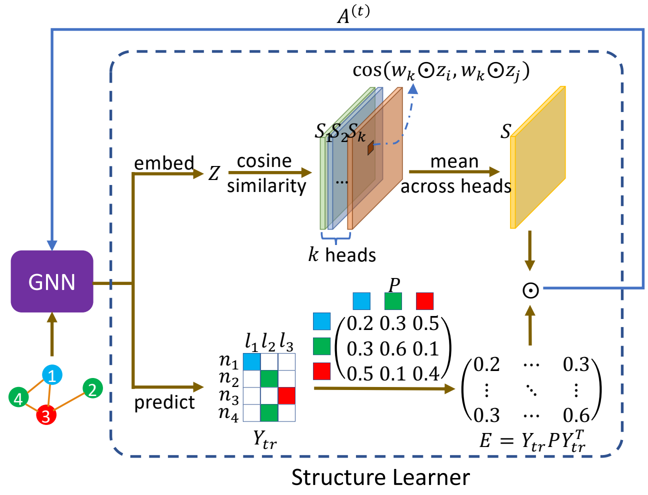

Figure 1: Framework of LGS.

2 METHODOLOGY

2.1 Prelinimary

Problem Formulation. Given a graph with an adjacency matrix $A \in \mathbb{R}^{n \times n}$ and a feature matrix $X \in \mathbb{R}^{n \times d}$. V_l is the set of labeled nodes. Since the original graph structure may be noisy and incomplete, the goal is to learn the optimal graph structure and make predictions for unlabeled nodes simultaneously.

2.2 Overview of LGS Framwork

As illustrated in Figure 1, LGS consists of a GNN and a structure learner. The GNN acts as a feature extractor and a classifier at the same time. On the one hand, the GNN outputs intermediate results Z generated by its last hidden layer as node embeddings which encode feature information. On the other hand, the GNN makes predictions for unlabeled nodes. Combining with the ground truth of labeled nodes, the GNN generates (pseudo) labels Y_{tr} for all the nodes. There are two branches in the graph structure learner, which consider feature information and label information respectively. The first branch computes multi-head weighted-cosine similarity between each pair of nodes according to their embeddings. Then, a feature similarity matrix is obtained by computing the mean across multiple heads. The second branch generates an edge probability matrix $E \in \mathbb{R}^{n \times n}$ based on (pseudo) labels Y_{tr} and a class transition parameter matrix $P \in \mathbb{R}^{c \times c}$.

2.3 GNN Architecture

Without loss of generality, we choose two representative GNN architectures as feature extractor: GCN [8] and ChebNet [2]. For GCN, the graph convolution in the l-th layer can be described as:

$$\begin{aligned} H^{(l)} &= \sigma\left(\tilde{A} H^{(l-1)} W^{(l-1)}\right), \\ \tilde{A} &= \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}, \end{aligned} \tag{1}$$

where $\hat{A} = A + I$ is the adjacency matrix of graph with self-loops, $\hat{D}$ is its corresponding degree matrix with $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$, and σ is non-linear activation function such as ReLU. As to ChebNet, the computation can be formulated as:

$$\begin{aligned} H^{(l)} &= \sigma\left(\sum_{k=0}^{K} \theta_k T_k(\tilde{L}) H^{(l-1)}\right), \\ \tilde{L} &= \frac{2L}{\lambda_{\max}} - I, \end{aligned} \tag{2}$$

where $L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$ is graph Laplacian matrix and T_k is the k-th order Chebshev polonomial.

In order to incorporate class transition matrix P into graph convolution explicitly, we add a label propagation layer weighted with P at the end of GNN similar to [16]. In conclusion, the output of GNN is formulated as:

$$(Z, \hat{Y}) = GNN(A, X), \tag{3}$$

in which A is an adjacency matrix, and X is a feature matrix.

2.4 Label-informed Graph Structure Learning

Feature Similarity Matrix. Although there are various options for distance or similarity computation, such as Euclidean distance, attention mechanism, Mahalanobis distance and cosine similarity. Without loss of generality, we adopt weighted consine similarity as metric function. To further enrich expressiveness, we adopt a multi-head manner similar to GAT[13]. Specifically, in the k-th head, the $n \times n$ similarity matrix S_k is given by:

$$S_k[i][j] = \cos(w_k \odot z_i, w_k \odot z_j), \tag{4}$$

where $\odot$ is element-wise product operator, w_k is a trainable weight and z_i, z_j are the i-th and j-th rows of Z, representing embeddings for node v_i, v_j respectively. Then, a feature similarity matrix $S \in \mathbb{R}^{n \times n}$is obtained by:

$$S = \frac{1}{K} \sum_{k=1}^{K} S_k. \tag{5}$$

Class Transition Matrix. To make full use of available label information, we employ a trainable matrix $P \in \mathbb{R}^{c \times c}$ to reweight similarity score between nodes, where c is the number of classes of node. Intuitively, $P_{s,t}$ could be interpreted as the probability that an edge exists between a s-th class node and a t-th class node.

According to the definition of P, $Y_{tr}^T A Y_{tr}$ serves as a good unnormed estimation. Considering that P should satisfy the double stochastic property (each row and each column sums to one), we propose to adopt the Sinkhorn-Knopp[12] algorithm which operates iteratively to generate a double stochastic matrix. So class transition matrix P is initialized as Sinkhorn-Knopp($Y_{tr}^T A Y_{tr}$).

Learning Graph Structure. Given a class transition matrix P, the probability of edges between each pair of nodes can be obtained according to their labels. Nevertheless, labels of most nodes are unavailable, so we assign pseudo labels to them according to predictions of the current model. Formally, let $Y \in \{0,1\}^{n \times c}$ be ground truth matrix where each row is an one-hot vector responding to each node, and $M \in \{0,1\}^n$ be mask for labeled nodes. Then we define Y_{tr} as:

$$Y_{tr} = M \odot Y + (1 - M) \odot \hat{Y}. \tag{6}$$

The i-th and j-th row of Y_{tr} represent (pseudo) labels of nodes v_i, v_j respectively. Only considering label information, edge probability matrix can be formulated as:

$$E = Y_{tr} P Y_{tr}^T. \tag{7}$$

To make training process more stable, we introduce a hyper-parameter r to control the weight of E. In the real world, underlying graph structures are relatively sparse than fully-connected graphs which not only include noise, but also are computationally expensive. In addition, elements of a typical adjacency matrix are non-negative. Hence, we obtain a sparse non-negative matrix through ϵ-neighborhood sparsification, which masks elements less than ϵ(a non-negative hyper-parameter controlling sparsity) to zero. In summary, the refined adjacency matrix can be formulated as:

$$\widetilde{A} = \epsilon\text{-neighborhood}(S \odot (r * E + (1 - r) * \mathbf{1})), \tag{8}$$

where $\mathbf{1}$ is the all ones matrix with the same shape to E.

```
Algorithm 1: Training of LGS
Input: $X, A, M, Y$
Parameters: $\alpha, \beta, \epsilon, r$
Result: $\hat{Y}^{(t)}$
$\hat{Y}^{(0)}, Z^{(0)} \leftarrow$ GNN(A, X)
compute $S_f$ according to $X$ following Eq 4, 5
for $t \leftarrow 1$ to $T$ do
    compute $S$ according to $Z^{(t-1)}$ following Eq 4, 5
    $Y_{tr} \leftarrow M \odot Y + (1 - M) \odot \hat{Y}^{(t-1)}$
    $E \leftarrow Y_{tr} P Y_{tr}^T$
    $A^{(t)} \leftarrow \epsilon$-neighborhood$(S \odot (r * E + (1 - r) * \mathbf{1}))$
    $\widetilde{A}^{(t)} \leftarrow \alpha A + \beta S_f + (1 - \alpha - \beta) A^{(t)}$
    $\hat{Y}^{(t)}, Z^{(t)} \leftarrow$ GNN$(\widetilde{A}^{(t)}, X)$
end
$\mathcal{L} \leftarrow L_c(\hat{Y}^{(0)}, Y) + \frac{1}{T} \sum_{t=1}^{T} L_c(\hat{Y}^{(t)}, Y) + \Phi(P)$
back-propagating through $\mathcal{L}$ to update parameters
```

2.5 Training

GNN Warm-up. In order to obtain relatively accurate pseudo label for unlabeled nodes, the GNN is trained solely for several epochs with classification loss function

$$L_c(\hat{Y}, Y) = \sum_{i \in V_l} \mathrm{CE}(\hat{Y}_i, Y_i), \tag{9}$$

where $\hat{Y}_i$ is the prediction of the GNN for node v_i, and CE denotes cross entropy loss.

Graph Structure Learning. The graph structure learner and the GNN are jointly optimized in an iterative manner for T steps. In the t-th iteration, given node embedding $Z^{(t-1)}$ and predictions $\hat{Y}^{(t-1)}$computed in the $(t-1)$-th iteration, the graph structure learner compute refined adjacency matrix $A^{(t)}$. Although the original graph structure may be inaccurate and incomplete, it still carries relatively rich and useful information. What's more, empirically, we find that feature similarity matrix S_f computed according to raw feature X serves as a relatively accurate refinement to the original graph structure. As a result, we combine A, S_f and $A^{(t)}$ together:

$$\widetilde{A}^{(t)} = \alpha A + \beta S_f + (1 - \alpha - \beta) A^{(t)}, \tag{10}$$

in which α and β are hyper-parameters that control relative importance assigned. Based on the refined graph structure $\widetilde{A}^{(t)}$, the GNN outputs node embeddings $Z^{(t)}$ and predictions $\hat{Y}^{(t)}$ for next iteration use.

Joint Optimization. After T iterations, total loss function is given by:

$$\Phi(P) = \sum\nolimits_i \left| \sum\nolimits_j P_{ij} \right|,$$
$$\mathcal{L} = L_c(\hat{Y}^{(0)}, Y) + \frac{1}{T} \sum_{t=1}^{T} L_c(\hat{Y}^{(t)}, Y) + \Phi(P), \tag{11}$$

in which L_c is cross-entropy classification loss and $\Phi(P)$ is a regularization item to encourage the sum of each row of transition matrix P to center around zero. Then, the GNN and the graph structure learner are optimized through common gradient descent algorithms.

Table 1: Data Statistics

	Cora	Citeseer	Cornell	Chameleon	Squirrel	Wisconsin	Texas
Homophily Ratio	0.81	0.74	0.3	0.23	0.22	0.21	0.11
# Nodes	2,708	3,327	183	2,277	5,201	251	183
# Edges	5,278	4,676	280	31,421	198,493	466	295
# Features	1,433	3,703	1,703	2,325	2,089	1,703	1,703
# Classes	7	6	5	5	5	5	5

3 EXPERIMENT

In this section, we conduct extensive experiments to verify the effectiveness of the proposed method LGS for node classification on both homophily and heterophily [9] graph datasets.

3.1 Setup

Datasets. For homophily graphs, we choose two citation networks, Cora and Citeseer [10]. For heterophily graphs, we choose Chameleon, Squirrel, Wisconsin and Texas [9]. Statistics for these datasets could be found in Table 1, where the homophily ratio of a graph represents the tendency of a node to have nodes of the same class as its neighbors, and can be computed as:

$$h_G = \frac{1}{n} \sum_{i=1}^{n} h_i = \frac{1}{n} \sum_{i=1}^{n} \frac{|N_i^s|}{|N_i|}, \tag{12}$$

where h_i represents homophily ratio of node v_i and N_i^s is the set of v_i' neighboring nodes with the same label to v_i. Low homophily corresponds to high heterophily. For all datasets, we follow the data splits given in Geom-GCN [9].

Baselines. We compare our methods with following methods from three categories: (1) classic GNN models for node classification: GCN [8], ChebNet [2] and GAT [13], (2) recent methods designed specially for heterophily graphs: GEOM-GCN [9] and CPGNN [16], and (3) the state-of-the-art models with graph structure learning: IDGL [1] and Pro-GNN [6].

3.2 Implementation

Even though our framework is agnostic to the choice of specific GNN architecture, we choose two representative GNNs: GCN and ChebNet, and the corresponding model variants are termed as LGS-GCN and LGS-Cheb respectively.

Table 2: Node classification accuracies.

	Cora	Citeseer	Cornell	Chameleon	Squirrel	Wisconsin	Texas
GCN	86.66 ± 1.45	76.25 ± 1.19	59.73 ± 6.33	38.99 ± 1.86	29.20 ± 1.10	53.92 ± 5.49	59.73 ± 6.33
ChebNet	86.14 ± 1.35	76.34 ± 1.59	74.86 ± 8.02	46.05 ± 1.46	30.30 ± 1.50	76.08 ± 2.29	74.59 ± 8.04
GAT	87.20 ± 1.09	75.92 ± 1.59	59.46 ± 4.01	44.06 ± 2.52	27.49 ± 1.52	54.71 ± 3.66	58.65 ± 4.84
GEOM-GCN	85.26 ± 1.57	**77.99±1.25**	60.54 ± 3.67	60.00 ± 2.81	38.15 ± 0.92	64.51 ± 3.66	66.76 ± 2.72
CPGNN	87.00 ± 1.02	76.07 ± 1.21	75.14 ± 7.43	62.21 ± 3.29	40.16 ± 6.43	76.47 ± 2.77	75.68 ± 7.15
IDGL	87.28 ± 1.00	76.88 ± 1.64	68.11 ± 8.87	38.51 ± 4.65	25.18 ± 2.40	55.69 ± 4.57	66.49 ± 6.07
Pro-GNN	83.52 ± 2.20	72.96 ± 1.99	62.97 ± 7.93	58.25 ± 3.84	32.59 ± 1.04	55.69 ± 5.96	60.81 ± 6.07
LGS-GCN	**87.38 ± 1.25**	76.92 ± 1.75	63.78 ± 7.76	56.27 ± 3.16	34.92 ± 2.21	53.14 ± 6.70	59.73 ± 6.99
LGS-Cheb	86.14 ±1.35	76.65 ± 1.78	**76.76 ± 8.56**	**71.45 ± 2.17**	**48.94 ± 4.44**	**76.86 ± 3.70**	**75.95 ± 7.69**

For a fair comparison, we implement our method and all baselines in the same experimental settings as Pei et al. [9]. We run all methods on all ten splits, and report mean and standard deviation of accuracies on the test set.

For hyper-parameter setting, we set the embedding dimension to 64, the number of layers to 2, ϵ to zero. And α is fixed at 0.8. We train the model using Adam optimizer [7] with an initial learning rate of 0.01. Moreover, for all the datasets, we first train the GNN alone for 400 epochs, then train the GNN and the graph structure learner for 1600 epochs together.

3.3 Main Results

Mean and standard deviation of accuracies for node classification on test sets over 10 splits are reported in Table 2. Our method obtains best performance on almost all the datasets with varing homophily ratios. Compared with graph structure learning (GSL) methods considering only feature information, our method outperforms them by a wide margin, reflecting the necessity to take available label information into consideration.

Compared with GEOM-GCN and CPGNN designed specially for graphs with strong heterophily, our method still achieves significant improvement, owing to the refined graph structure by considering both feature information and label informaiton.

Notably, ChebNet outperforms GCN by a wide margin on graphs with high hetetrophily ratios, and is slightly inferior on homophily graphs. As analyzed in [11], GCN implicitly treats high-frequency components as “noises”, and has them discarded. However, this may hinder the generalizability since high-frequency components can carry meaningful information about local discontinuities, This could also explain why LGS-Cheb performs better than LGS-GCN on heterophily graphs like Chameleon, Squirrel, etc.

3.4 Accuracy versus Homophily

For a better understanding of the success of our method, we analyze the relationship between classification accuracy with homophily ratios of nodes. On Chameleon dataset, we split the range [0, 1] of homophily ratio into ten segments, and analyze the percentage of nodes falling in each one. What’s more, we calculate the classification accuracy for each sub-range. As shown in Figure 2, GCN performs poorly on nodes with low homophily. And IDGL’s graph structure learner may result in negative effect due to its implicit assumption of homophily. In contrast, LGS improves the accuracy of nodes with strong heterophily without harming performance on nodes with high homophily.

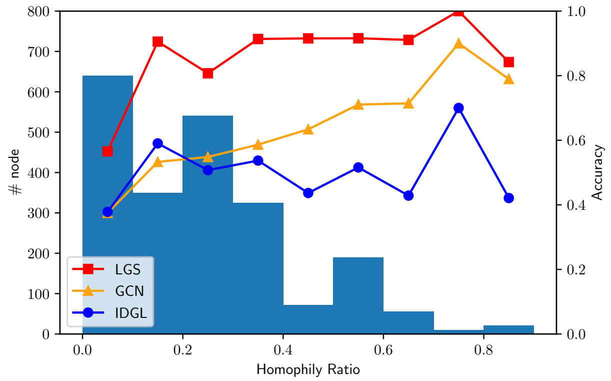

Figure 2: Distribution of nodes with homophily ratio and classification accuracy for LGS, GCN and IDGL on Chameleon dataset.

4 CONCLUSION

In this paper, we introduce a novel label-informed graph structure learning framework (LGS). Apart from feature information, LGS incorporates label information into graph structure learning explicitly through a class transition matrix. We conduct extensive experiments on both homophily and heterophily graph datasets. Experimental results show that LGS improves the accuracy of nodes with strong heterophily without harming the performance on nodes with high homophily, reflecting the superiority of LGS.

ACKNOWLEDGMENTS

This work is supported by National Key Research and Development Program (2018YFB1402600), National Natural Science Foundation of China (61772528), and Shandong Provincial Key Research and Development Program (2019JZZY010119).

REFERENCES

[1] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. *Advances in Neural Information Processing Systems* (2020).

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*.

[3] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*. 1972–1982.

[4] Fenyu Hu, Yanqiao Zhu, Shu Wu, Weiran Huang, Liang Wang, and Tieniu Tan. 2021. GraphAIR: Graph representation learning with neighborhood aggregation and interaction. *Pattern Recognition* (2021).

[5] Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. 2019. Hierarchical graph convolutional networks for semi-supervised node classification. *arXiv preprint arXiv:1902.06667* (2019).

[6] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[7] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[9] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

[10] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* (2008).

[11] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* (2013).

[12] Richard Sinkhorn and Paul Knopp. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.* (1967).

[13] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[14] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*.

[15] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems*.

[16] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2020. Graph Neural Networks with Heterophily. *arXiv preprint arXiv:2009.13566* (2020).

[17] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2021. Deep Graph Structure Learning for Robust Representations: A Survey. arXiv:2103.03036 [cs.LG]