Proceedings of Machine Learning Research 189, 2022 ACML 2022

A Novel Graph Aggregation Method Based on Feature Distribution Around Each Ego-node for Heterophily

Shuichiro Haruta SH-HARUTA@KDDI.COM
Tatsuya Konishi TT-KONISHI@KDDI.COM
Mori Kurokawa MO-KUROKAWA@KDDI.COM
KDDI Research, Inc. 2-1-15 Ohara, Fujimino-shi, Saitama, 356-8502, Japan

Editors: Emtiyaz Khan and Mehmet Gönen

Abstract

In this paper, we propose a novel graph aggregation method based on feature distribution around each ego-node (a node to which features are aggregated) for heterophily. In heterophily graphs, labels of neighboring nodes can be uniformly distributed. In such case, aggregated features by existing GNNs will be always similar regardless of the label of ego-node and fail to capture useful information for a node classification task. Since the existing methods basically ignore label distribution around the ego-node, we attempt to handle heterophily graphs through dynamic aggregations so that nodes with similar vicinity characteristics exhibit similar behavior. In particular, we adjust the amount of aggregation based on the features generated by higher-order neighbors, since they reflect the label distribution around each ego-node. By doing this, we can take the influence of distant nodes into account while adapting local structures of each node. Extensive experiments demonstrate that the proposed method achieves higher performance in heterophily graphs by up to 14.68% compared with existing methods.

Keywords: Graph Neural Network; Graph Convolution; Heterophily; Deep Learning;

1. Introduction

Graphs are well-known data structure, which can express relationships among entities. They are widely used in many real world scenarios, such as social networks (Tang et al. (2009); Li and Goldwasser (2019)), knowledge graphs (Wang et al. (2019)) and so on. Recently, in order to extract hidden patterns or characteristics from graph data, Graph Neural Networks (GNN) have been extensively studied (Zhang et al. (2020)). With the intention to let node embedding involve patterns underlying graphs, typical GNNs utilize neighbor nodes of each node by aggregating their features. In particular, graph convolutional network (GCN) (Kipf and Welling (2017)) has exhibited powerful performance and have achieved great success in many tasks, including community detection (Jin et al. (2021)), recommendation systems (Wang et al. (2019); He et al. (2020)), fraud detection (Dou et al. (2020)), biology (Yan et al. (2019)) and more.

Although GCN and other traditional GNNs, such as GAT (Veličković et al. (2018)), are still popular, recent studies have pointed out that they can be ineffective unless the input graphs have "homophily" characteristics. Homophily is a characteristic of a graph in which two linked nodes are likely to share similar attributes. Citation networks, for example, are typical homophily graphs since papers in close fields tend to be cited. Early studies focused on homophily datasets, such as Citeseer (Giles et al. (1998)), and obtained significant performance gains (Goyal and Ferrara (2018)). However, real world graphs do not necessarily have high homophily characteristics. Instead, they demonstrate high "heterophily". In molecular networks, for instance, different types of amino acids tend to form links with each other (Zhu et al. (2020)). Similarly, in dating networks, most people are likely to date with the opposite gender (Zhu et al. (2021)). These types of graphs are referred to as heterophily graphs. Traditional GNNs do not deal with heterophily graph well because their convolution operations only care about local neighbors or topology. This fails to capture informative nodes with long-term distances (Zheng et al. (2022)). Hence, GNNs that are supposed to have dedicated mechanisms for heterophily graphs have been attracting attention.

In this connection, several methods have been proposed. Most of them try to utilize the nonadjacent nodes and distant nodes since neighboring nodes do not necessarily show useful characteristics in a heterophily graph. Geom-GCN (Pei et al. (2020)) discovers node pairs representing similar latent features and considers them in the aggregation process in GCN. In H_2GCN (Zhu et al. (2020)), the authors argue that aggregating higher-order neighbors and separating them from ego-node (a node to which features are aggregated) embedding works well in a heterophily graph. Although it is known that simply stacking vanilla-GCN layers might incur oversmoothing problems (Wu et al. (2019)), GCNII (Chen et al. (2020)) and GGCN (Yan et al. (2021)) manage to capture higher-order neighbor signals by mitigating this problem. Although various such techniques have been employed, those methods, in principle, attempt to handle heterophily by incorporating the nodes' features through higher-order hops. In particular, as argued in H_2GCN, aggregating higher-order nodes can extract more distinguishable patterns of nodes, which directly contributes to higher performance.

However, whether such aggregation mechanism performs well or not depends on the correlation between each ego-node's label and aggregated higher-order features. In particular, there can be a heterophily graph in which the labels of neighboring nodes tend to be uniformly distributed. In this case, the aggregated features will be always similar regardless of ego-node's label, no matter how many hops away they are aggregated from. Nevertheless, the existing methods basically ignore label distribution around the ego-node due to their uniform aggregation. Accordingly, aggregation mechanisms should be determined based on different vicinity features for each ego-node.

To provide more flexible aggregation for a heterophily graph, in this paper, we propose EVAGNN (short for Ego-node Vicinity Adaptive GNN), a novel graph aggregation method based on feature distribution around each ego-node for heterophily. The goal is to handle heterophily graphs through dynamic aggregations so that nodes with similar vicinity characteristics exhibit similar behavior. In particular, we adjust the amount of aggregation based on the features generated by higher-order neighbors, since they reflect the label distribution around each ego-node. By doing this, we can take the influence of distant nodes into account while adapting local structures of each node. Extensive experiments demonstrate that the proposed method EVAGNN shows the high performance in heterophily graphs and has improvements of up to 14.68% compared with existing methods.

The contributions of this paper are as follows:

- To the best of our knowledge, ours is the first aggregation method, which can adapt weight parameters according to the vicinity of each ego-node in a graph. As shown in the experimental results, the performance is effective, especially in a heterophily graph.
- Through the analysis of real world graphs, we show the reason why the existing methods do not effectively work in specific graphs. We believe that this observation sheds light on a new perspective for handling heterophily graphs.

The rest of this paper is organized as follows. We describe the notations, the formal homophily definition, and a brief GNN introduction in Section 2. Section 3 outlines related studies on both homophily and heterophily graphs and clarifies how our proposed method differs from them. The proposed method EVAGNN is presented in Section 4. Extensive experiments with nine standard benchmark datasets are conducted, and the results are reported in Section 5. Finally, we conclude the paper in Section 6.

2. Preliminaries

We first define the notations used throughout the paper. Furthermore, important concepts, homophily/heterophily definitions and an introduction to GNN are also presented in this section.

2.1. Notations

We assume graphs without edge weights. A graph G consists of a set of n nodes $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_{ij}\} \subseteq V \times V$. The graph structure is denoted by the adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{n \times n}$. $a_{ij} = 1$ if there is an edge e_{ij} between nodes i and j; otherwise, $a_{ij} = 0$. The neighbors of node i are denoted by the node set of $N_i = \{v_j | a_{ij} = 1\}$. Each node has its own feature vector, and all n nodes' features are represented by $X \in \mathbb{R}^{n \times f}$, where f indicates the length of each node's feature. The i-th row of X corresponds to node i's feature, which is hereafter denoted by X_i. The set of node labels is denoted by $C = \{c_1, c_2, \ldots, c_m\}$, where m is the number of classes. Every node has one label, and $C(v_i)$ denotes a node i's label.

2.2. Homophily and Heterophily

The term "homophily" indicates that two linked nodes tend to share similar attributes in a graph. Thus, one way to define homophily on a graph basis is to count how many neighbors with the same label each node has and then taking its average over all nodes. Thus, the homophily ratio h of the graph is defined by

$$h = \frac{1}{|V|} \sum_{N_i} \frac{|\{v_j | C(v_i) = C(v_j)\}|}{|N_i|} \in [0, 1], \tag{1}$$

where $|\cdot|$ is the number of elements in the set. Note that $0 \leq h \leq 1$. The term "heterophily" indicates the opposite of homophily; thus, high homophily corresponds to low heterophily. We use these terms interchangeably throughout the paper.

2.3. Graph Neural Network

GNNs are often used to solve prediction tasks related to graphs. Almost all GNN methods attempt to represent nodes by aggregating their neighbor information. The graph convolutional network (GCN) (Kipf and Welling (2017)) is a popular GNN, and it aggregates neighbor features by

$$X^{(l+1)} = \hat{A}X^{(l)}W, \tag{2}$$

where $\hat{A}$, $X^{(l)}$, and W are the degree normalized adjacency matrix with added self-loop, l-th GCN layer's output, and trainable weight matrix, respectively. Such aggregation operations are frequently employed in both traditional methods (e.g., GAT (Veličković et al. (2018)) and GraphSAGE (Hamilton et al. (2017))) and recently proposed methods described in the next section. A series of procedures in which features (messages) are aggregated and then updated through neighbor nodes is called "message passing".

3. Related Work

Traditional GNNs suffer from degradation when applied to heterophily datasets as we have discussed in Section 1. To deal with heterophily graphs, in which neighbor nodes become less informative than in homophily graphs, several mechanisms try to include distant nodes in aggregation operations. In this section, we introduce representative studies that can handle the heterophily.

Most of them try to utilize the nonadjacent nodes since neighboring nodes show different characteristics in a heterophily graph. Geom-GCN (Pei et al. (2020)) discovers node pairs representing similar latent features, and lets them be considered in the aggregation process in GCN. That scheme enables nodes that are far apart to be taken into account and thereby the performance is improved compared with vanilla-GCN. In H_2GCN (Zhu et al. (2020)), the authors argue that aggregating higher-order neighbors (k-hop away from an ego-node) and separating them from ego-node embedding works well in heterophily graphs. While Geom-GCN and H_2GCN directly aggregate higher-order neighbors, GCNII (Chen et al. (2020)) and GGCN (Yan et al. (2021)) focus on deepening the GNN layers and try to capture higher-order neighbors' signals through repeated aggregations. Although it is known that simply stacking vanilla-GCN layers might incur oversmoothing problems (Wu et al. (2019)), GCNII proposes simulating ResNet's skip connection (He et al. (2016)) in the GNN field and makes more GCN layers available. In GGCN (Yan et al. (2021)), the authors develop a negative message propagation mechanism based on the similarity of node representations. Negative messages can negate messages sent by differently labeled neighbors and adapt to heterophily settings. A similar idea is adopted in FAGCN (Bo et al. (2021)), which aims to separate features into low- and high-frequency components based on message passing.

3.1. Problems

As mentioned in Section 1, we argue that aggregation mechanisms should be determined based on different vicinity features for each ego-node. Fig. 1 shows a toy example of a heterophily graph. In this figure, while most of neighbors of blue nodes are red nodes in region 1 and 2, yellow nodes surround blue nodes in region 3. When using normal aggregation, aggregated features of blue nodes in region 1 and 2 will be similar. On the

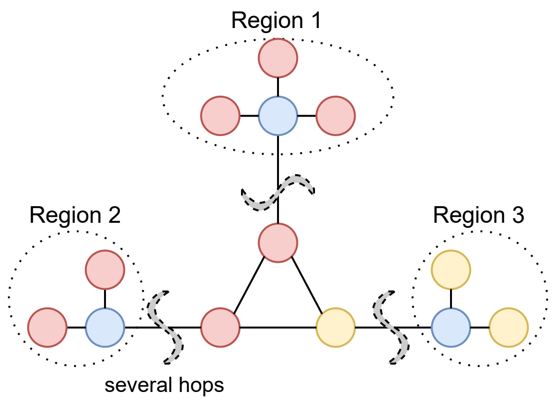
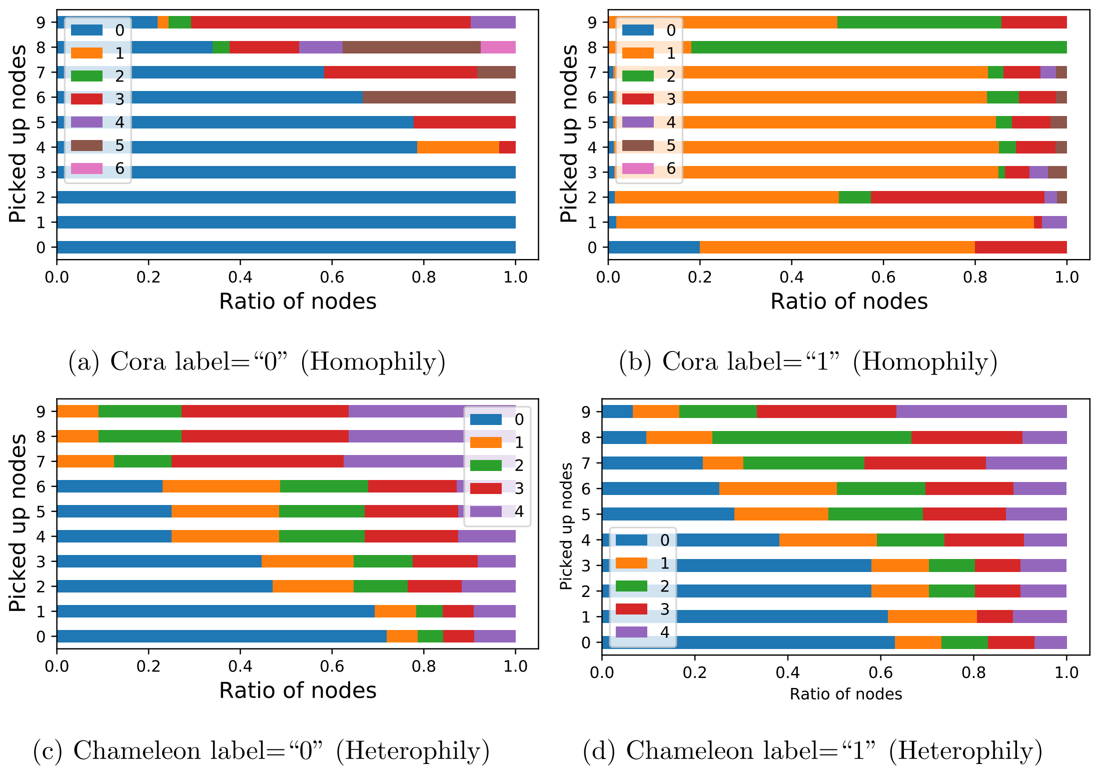


Figure 1: A toy example of a heterophily graph. The colors of the nodes represent their labels. The distribution of labels differs by region, i.e., while most of the neighbors of blue nodes are red nodes in regions 1 and 2, yellow nodes surround blue nodes in region 3.



(a) Cora label="0" (Homophily)

(b) Cora label="1" (Homophily)

(c) Chameleon label="0" (Heterophily)

(d) Chameleon label="1" (Heterophily)

Figure 2: 2-hop neighbor's labels distribution of nodes with same label (Chameleon dataset). 10 nodes that have the same label are randomly picked up for this comparison.

other hand, one of region 3 will not resemble them. Therefore, in order to identify them correctly, it is necessary to change weights according to each blue node in the aggregation process. However, none of existing approaches has regarded this perspective.

To demonstrate the difference of label distribution around each node in the real datasets, we investigate Cora (homophily) and Chameleon (heterophily) dataset (see details in Section 5). Fig. 2 shows 2-hop neighbor's label distribution of the nodes labeled as "0" and "1" in Cora and Chameleon. As shown in Fig. 2a and Fig. 2b, in the homophily graph, most of 2-hop neighbors' labels are the same as the ego-nodes regardless of their labels. On the other hand in the heterophily graph, we can see in Figs. 2c and 2d that there are obvious differences in 2-hop neighbors label distribution from homophily ones. These figures imply that even if the labels of nodes are the same, they do not necessarily resemble each other in terms of higher-order neighbors' labels in heterophily graphs. This is the reason that we believe current state-of-the-art heterophiliy-aware methods, which have never considered it, can be further improved.

4. Proposed Method

To provide more flexible aggregation for a heterophily graph, we propose EVAGNN (short for Ego-node Vicinity Adaptive GNN), a novel graph aggregation method based on feature distribution around each ego-node for heterophily. The goal is to handle heterophily graphs through dynamic aggregations so that nodes with similar vicinity characteristics exhibit similar behavior. In particular, we adjust the amount of aggregation based on the features generated by higher-order neighbors, since they reflect the label distribution around each ego-node. By doing this, we can take the influence of distant nodes into account while adapting local structures of each node.

4.1. Algorithms

First, we define the k-hop vicinity feature of node i by aggregating its neighbors as

$$H_i^k = \sum_{j \in N_i^k} \frac{X_j}{|N_j|} \in \mathbb{R}^d, \tag{3}$$

where N_i^k indicates the k-hop neighbor of node i and $H_i^0 = X_i$. The k-hop neighbors can be computed through Lemma 1.

Lemma 1 *Given an adjacency matrix A, the element (i, j) of A^k gives the number of walks of length k from node i to node j. Therefore, for $k \geq 2$, we can obtain k-hop neighbors by checking $A_{ij}^k > 0$ and $A_{ij}^{k-1} = 0$*

To express the node i-centered vicinity features over the k-hop, we concatenate node i's vicinity features as

$$H_i = H_i^0 \mid\mid H_i^1 \mid\mid \ldots \mid\mid H_i^k \in \mathbb{R}^{d(k+1)}, \tag{4}$$

where $\mid\mid$ denotes concatenation operation. Let H denote vicinity features over k-hop of all n nodes. Using H, we compute the convolution weights. Here, we use the self-attention mechanism (Vaswani et al. (2017)) to capture the relationships of features across hops and robustly represent weights. The query, key and value are all derived from H as

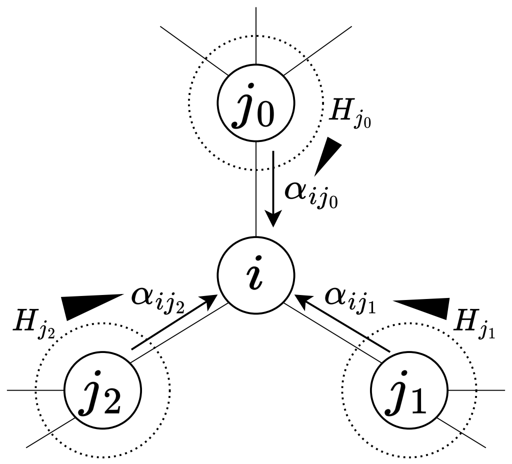


Figure 3: Overview of the proposed convolution. Each neighbor node around node i (e.g., j_0) has another vector (e.g., H_{j_0}) associated with it apart from its feature vector. It is named the "vicinity feature". This new type of feature reflects the vicinity nodes' label distribution for each node (i.e., vicinity characteristics), and based on this the vicinity similarity can be further derived. Using this perspective, the weight of each node (e.g., α_{ij_0}) in convolutions is determined depending on this vicinity feature so that nodes with similar vicinity characteristics exhibit similar behavior.

$$Q = HW_{\mathrm{q}} \in \mathbb{R}^{n\times d}, \tag{5}$$
$$K = HW_{\mathrm{k}} \in \mathbb{R}^{n\times d}, \tag{6}$$
$$V = HW_{\mathrm{v}} \in \mathbb{R}^{n\times d}, \tag{7}$$

where W_q, W_k, W_v, and d are trainable parameters and the size of embedding dimensions, respectively. The proposed convolution weight, α, is obtained by

$$\alpha = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \in \mathbb{R}^{n\times d}. \tag{8}$$

Note that the attention weights might converge to some values at some point across multiple iterations, since the vicinity features are precomputed. However, they keep updated while the weights ($W_{\mathrm{q}}, W_{\mathrm{k}},$ and W_{v}) change. By taking elementwise product of α and X, the features passed to each node are obtained. The output of the proposed convolution is obtained by multiplying the adjacency matrix as

$$X_{\mathrm{conv}} = \sigma(A(\alpha \odot X)). \tag{9}$$

where σ is an activation function (we use ReLU). Similar to $\mathrm{H_2GCN}$, we encode the ego-node separately from aggregated features since an ego-node in a heterophily graph is likely to be dissimilar with its neighbors in terms of class labels. This different encoding can contribute to more representation capacity. Ego-node features are generated by MLP, and the final representation of nodes are

$$X_{\text{out}} = \sigma(XW_{\text{MLP}}) + X_{\text{conv}} = \sigma(XW_{\text{MLP}}) + \sigma(A(\alpha \odot X)). \tag{10}$$

Note that multiple layers can be piled up, although a single convolution is conducted in the above description. In this case, X_{out} will be the input of the following layer. How the proposed aggregation operates is presented in Fig. 3. As shown in this figure, we calculate the convolution weight from H_j, vicinity features of v_j. Since the weights can be changed according to vicinity features, i.e., the local structure of a graph, the proposed convolution can automatically determine which neighbor nodes are important and are worth being passed.

4.2. Optimization objective

We focus on the node classification task. The objective of this task is to predict the labels of unlabeled nodes by using node features and the graph structure. That is, for node i, we need to learn the mapping function $f(X, A) \mapsto C(v_i)$.

To adjust the embedding size to the number of classes, we linearly transform X_{out} by the weight matrix $W \in \mathbb{R}^{d\times m}$ and obtain the probabilities that each node belongs to each class as

$$B = \text{softmax}(X_{\text{out}}W) \in \mathbb{R}^{n\times m}. \tag{11}$$

Let Θ and V_{train} denote all trainable parameters in the model and the set of labeled nodes, respectively. We minimize the cross-entropy loss and get optimal parameters Θ^* as

$$\Theta^* = \underset{\Theta}{\text{argmin}} \sum_{v\in V_{\text{train}}} \mathcal{L}(b_v, C(v)), \tag{12}$$

where $\mathcal{L}$ and b_v are the cross-entropy loss function and the predicted class of node v calculated based on B, respectively.

5. Experiments

In this section, we evaluate EVAGNN on the real world datasets and compare it with the latest baselines.

5.1. Baselines

We compare EVAGNN with 8 approaches. Four of them are latest heterophily-aware GNNs (Geom-GCN (Pei et al. (2020)), H_2GCN (Zhu et al. (2020)), GCNII, (Chen et al. (2020)), and GGCN (Yan et al. (2021))), and three of them are traditional GNN methods (GCN (Kipf and Welling (2017)), GAT (Veličković et al. (2018)), and GraphSAGE (Hamilton et al. (2017))). We also check the performance of MLP.

Table 1: Statistics of datasets. We regard the first six datasets as heterophily datasets, while the last three as homophily datasets.

	Homo. ratio h	# of Nodes	# of Edges	# of classes
Texas	0.11	183	295	5
Wisconsin	0.21	251	466	5
Film	0.22	7,600	26,752	5
Squirrel	0.22	5,201	198,493	5
Chameleon	0.23	2,277	31,421	5
Cornell	0.30	183	280	5
Citeseer	0.74	3,327	4,676	7
Pubmed	0.80	19,717	44,327	3
Cora	0.81	2,708	5,278	6

5.2. Datasets and Training Details

The following nine real datasets are used:

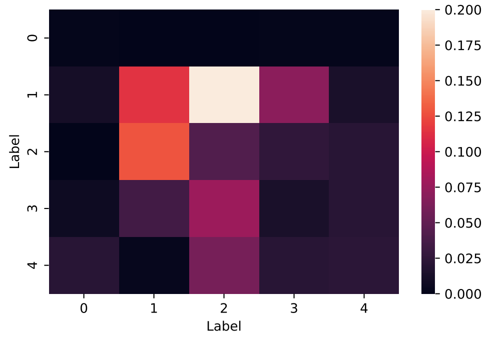
WebKB includes “Texas”, “Wisconsin”, and “Cornell”. Nodes and edges are web pages and their hyperlinks.

Wikipedia networks includes “Chameleon” and “Squirrel”. Nodes are web pages in Wikipedia. Edges are their mutual hyperlinks.

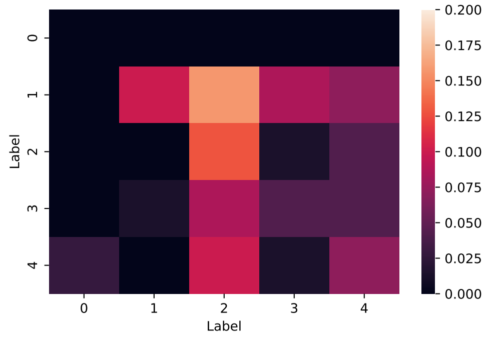
Actor co-occurrence networks includes “Film”. Each node is an actor. Edges are created when two actors appear on the same Wikipedia page.

Citation networks include “Citeseer”, “Pubmed”, and “Cora”. Nodes are papers and edges indicate their citation relationships.

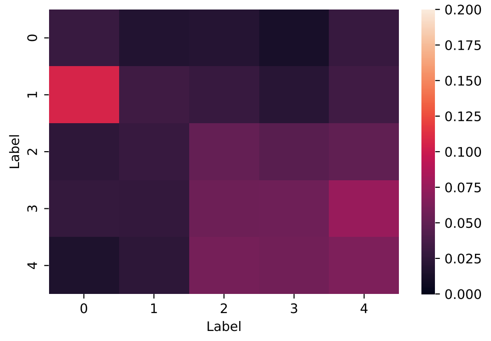
Tab. 1 shows the statistics of each dataset. For these, the homophily ratio h is calculated by Equation (1). As h shows, the graphs in WebKB, Wikipedia networks, and Actor co-occurrence networks tend to be heterophily, and citation networks tend to be homophily. We obtain these data through PyG (Fey and Lenssen (2019)) framework and use the ten data splits provided by the authors of Geom-GCN (Pei et al. (2020)), which are commonly used by researchers who study heterophily-aware GNNs. We report the accuracy averaged over these 10 splits. For MLP, GCN, GAT, and GraphSAGE, we use the same hyper parameter settings as GGCN. For other methods, the original codes provided by the authors are used with the default parameters. For EVAGNN, we fix the number of layers and embedding dimension to 1 and 64, respectively. The tuning range of k is $[1, 5]$. We use multi-head attention, and the number of heads is tuned within the range of $[1, 16]$. Other hyperparameters are tuned by using the same range as GGCN. Each model is optimized by the Adam optimizer (Kingma and Ba (2015)).



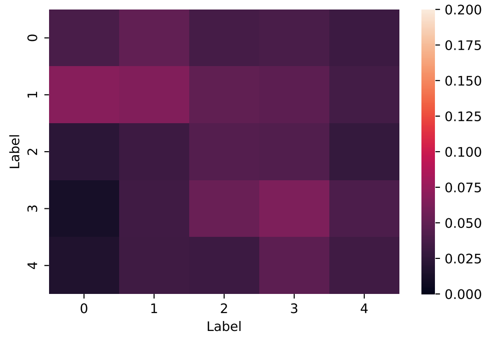
(a) Wisconsin hop= 1



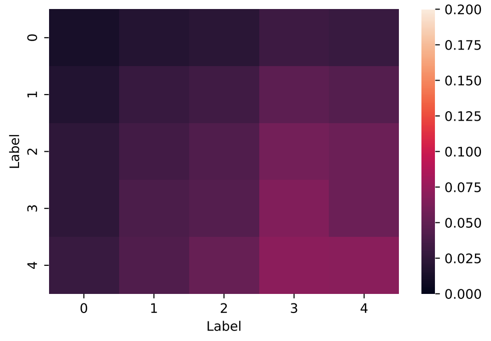
(b) Wisconsin hop= 2



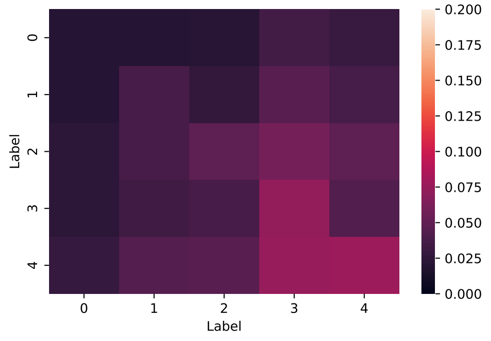
(c) Chameleon hop= 1



(d) Chameleon hop= 2



(e) Film hop= 1



(f) Film hop= 2

Figure 4: The connecting probability among labels on different datasets. In (a), about 20 % of 1-hop neighbors of a node with label 1 are nodes with label 2. We choose representative datasets for each type of datasets, i.e., WebKB, Wikipedia networks, and Actor networks.

Table 2: Node classification accuracy on each dataset. The results of GraphSAGE and H_2GCN are taken from (Zhu et al. (2020)) and (Yan et al. (2021)). Best performance in each dataset is emphasized in **bold**, and second best performance is underlined.

(a) Heterophily dataset

	Texas	**Wisc.**	**Film**	**Squi.**	**Cham.**	**Corn.**
MLP	80.81±4.75	85.29±3.31	36.53±0.70	28.77±1.56	46.21±2.99	81.89±6.40
GCN	55.14±5.16	51.76±3.06	27.34±1.08	53.33±2.09	64.85±2.24	60.54±5.30
GAT	52.16±6.63	49.41±4.09	27.55±0.93	40.72±1.44	60.13±2.43	61.89±5.05
GraphSAGE	82.43±6.14	81.18±5.56	34.23±0.99	41.61±0.74	58.73±1.68	75.95±5.01
Geom-GCN	67.84±3.91	65.29±3.83	31.59±1.19	38.46±0.71	60.42±2.80	60.27±2.72
H_2GCN	84.86±7.23	**87.65±4.98**	35.70±1.00	36.48±1.86	60.11±2.15	82.70±5.28
GCNII	77.57±3.83	80.39±4.30	37.48±1.31	38.23±1.74	62.26±3.70	78.92±4.65
GGCN	84.86±4.55	86.67±3.70	**37.49±1.57**	54.65±1.69	71.12±1.79	**85.68±6.63**
EVAGNN	**87.57±3.67**	87.25±3.75	35.72±1.00	**69.33±1.73**	**78.49±1.31**	84.32±6.92

(b) Homophily dataset

	Cite.	**Pubm.**	**Cora**
MLP	74.02±1.90	87.16±0.37	75.69±2.00
GCN	76.67±1.51	88.37±0.52	86.90±1.03
GAT	76.62±1.33	86.11±0.58	87.22±1.21
GraphSAGE	76.04±1.30	88.45±0.50	86.90±1.04
Geom-GCN	**78.21±1.05**	89.96±0.43	85.37±1.50
H_2GCN	77.11±1.57	89.49±0.38	87.87±1.20
GCNII	77.26±1.68	**90.16±0.28**	**88.37±1.29**
GGCN	77.14±1.45	89.14±0.37	87.95±1.05
EVAGNN	76.26±1.47	88.23±0.52	86.00±1.28

5.3. Main Results

Tab. 2a shows the accuracy of the different methods on heterophily graphs. EVAGNN shows the high performance on these dataset. Especially, in Squirrel and Chameleon dataset, EVAGNN has the the best performance compared with other methods. As mentioned in Section 3.1, the label distribution of these heterophily dataset has less correlation with ego-node's label than in homophily datasets. Thus, uniformly aggregated features at higher hops employed in Geom-GCN and H_2GCN are insufficient to identify labels. EVAGNN can learn how to aggregate features, i.e., which feature should be emphasized, and this brings better performance.

Table 3: Node classification accuracy of EVAGNN without MLP on heterophily datasets. The results of normal EVAGNN are shown again.

	Texas	**Wisc.**	**Film**	**Squi.**	**Cham.**	**Corn.**
EVAGNN w/o MLP	72.97±10.26	67.65±5.21	25.61±1.20	**71.62±1.94**	78.42±1.14	57.30±3.38
EVAGNN	87.57±3.67	87.25±3.75	35.72±1.00	69.33±1.73	78.49±1.31	84.32±6.92

To further analyze the results, we investigate the connecting probability among labels. Fig. 4 shows the connecting probability among labels in three datasets (see caption for how to read figures). In Wisconsin dataset shown in Figs. 4a and 4b, we can see that the areas with high probability (i.e., areas with light color) at hop=2 do not always appear also at hop=1 and the number of connection patterns is small. In such cases, higher-order aggregation performs well since the aggregated features show distinguishable characteristics according to the hops. EVAGNN also works well since it can easily capture local structures around each node. On the other hand, Chameleon dataset has more patterns, and there are small difference among connecting probabilities, as shown in Figs. 4c and 4d. We consider it hinders learning efficiency in the existing baselines since aggregated features, which should be distinctive, are prone to become similar. Although the tendency of connecting probability of Film dataset in Figs. 4e and 4f looks similar to one of Chameleon dataset, the performance of EVAGNN is not so high. We consider this is because the labels are not balanced, i.e., while every label has almost the same number of nodes in Chameleon and Squirrel dataset, label 4 has more than twice nodes than label 0 in Film dataset. In addition, as mentioned in (Zhu et al. (2020)), low-quality node features might also affect performance.

On the other hand, as shown in Tab. 2b, the performance of EVAGNN on homophily datasets is inferior to the other heterohphily-aware baselines. GCNII and GGCN, which utilize more layers, show better results. EVAGNN achieves the equivalent performance to traditional GNNs. As shown in Section 3.1, 2-hop neighbor aggregation works well in homophily dataset. Although EVAGNN takes distant nodes into account in calculating aggregation weight, it only aggregates 1-hop neighbors. Thus, compared with EVAGNN, Geom-GCN and H_2GCN show better results since they can directly aggregate higher-order neighbors. To more effectively handle homophily graphs is a future work. However, whether a graph is heterophily or homophily can be easily judged just by computing the homophily ratio h as shown in Tab. 1. Given that those results demonstrate the proposed EVAGNN outperforms other baselines or shows at least competitive accuracy to them, people can use either EVAGNN if a graph with low h is obtained or other methods if one with high h is obtained, to stably accomplish high performance.

5.4. Ablation Study

5.4.1. Effectiveness of MLP addition

Motivated by the fact that MLP works well in the heterophily dataset, we evaluate EVAGNN without using MLP output defined in Equation (10). Tab. 3 shows the performance of EVAGNN without MLP. As we can see from this table, the performances degrade compared

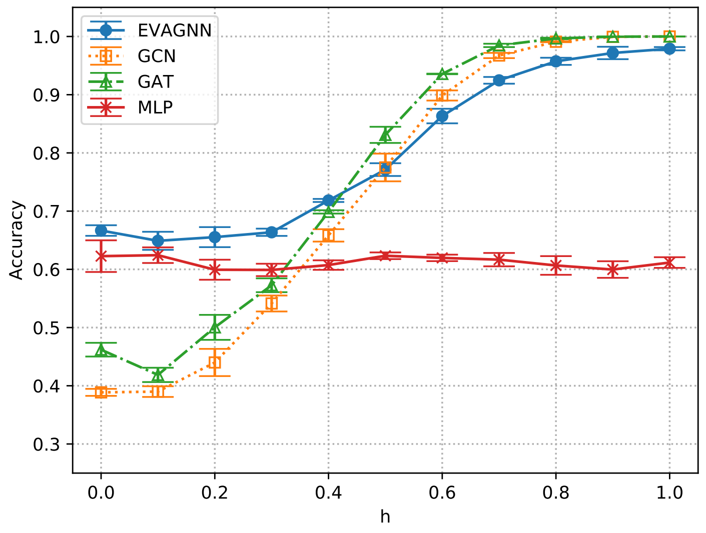


Figure 5: Performance on syn-products. This dataset has three graphs for each h, and the average accuracy is reported here. The number of nodes in each graph is 10,000, and the mean number of edges is 59,467.

with Tab. 2a in all datasets except for Squirrel. As for WebKB datasets (Texas, Wisc, and Cornell), they often include nodes with few edges. That fact seems to impact on the performance of aggregating features on GNNs. It is also demonstrated by 10.11% of decrease in accuracy that MLP is essential for handling Film. By contrast, the performance on Squirrel has further 2.29% improvement, and the performance on Chameleon has little change. As shown in Tab. 2a, MLP does not improve performance in these two dataset. Overall, for datasets in which MLP effectively works by itself, we should separately add the ego-node embedding to the proposed aggregation method as shown in Equation (10). This generates a synergistic effect in terms of learning representation. Otherwise, it might incur slight noise, which leads to poorer performance. We note that GCNII and GGCN also have similar terms which handle ego-node embeddings in aggregation process.

5.4.2. Performance on Synthetic Dataset

To further analyze the performance of EVAGNN, we conduct an experiment on the synthetic dataset (called "syn-products" (Zhu et al. (2020))), in which the homophily ratio h varies from 0 to 1. Fig.5 shows the performance on syn-products where y and x axes indicate the classification accuracy and h, respectively. We can observe almost the same trend as the results from the real dataset, i.e., EVAGNN performs much better than other GNN methods in heterophily graphs (smaller h), while it shows a slightly worse performance in homophily graphs (larger h). Based on the results from the real and synthetic datasets, it is true that the proposed EVAGNN may lose its performance in homophily graphs. As mentioned in Section 5.3, to more effectively handle homophily graphs is a future work.

6. Conclusion

In this paper, we have proposed EVAGNN, a novel graph aggregation method based on feature distribution around each ego-node for heterophily. Through analyzing dataset, we have found the necessity for aggregation method based on the vicinity of each ego-node. EVAGNN modifies the aggregation in existing GNNs, and we adjust the amount of aggregation based on the features generated by higher-order neighbors. Extensive experiments have demonstrated that EVAGNN shows the high performance especially in heterophily graphs and shows competitive performance in homophily dataset.

Acknowledgments

This research is partially supported by JST CREST Number JPMJCR21F2, Japan

References

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond Low-frequency Information in Graph Convolutional Networks. In Proc. of AAAI, 2021.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and Deep Graph Convolutional Networks. In Proc. of ICML, 2020.

Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters. In Proc. of CIKM, 2020.

Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In Proc. of RLGM Workshop at ICLR, 2019.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. CiteSeer: An Automatic Citation Indexing System, 1998.

Palash Goyal and Emilio Ferrara. Graph Embedding Techniques, Applications, and Performance: A Survey. Knowledge-Based Systems, 151:78–94, 2018.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In Proc. of NeurIPS, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In Proc. of CVPR, 2016.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proc. of SIGIR, 2020.

Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip Yu, and Weixiong Zhang. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. IEEE Transactions on Knowledge and Data Engineering, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Proc. of ICLR, 2015.

Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In Proc. of ICLR, 2017.

Chang Li and Dan Goldwasser. Encoding Social Information with Graph Convolutional Networks forPolitical Perspective Detection in News Media. In Proc. of ACL, 2019.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric Graph Convolutional Networks. In Proc. of ICLR, 2020.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social Influence Analysis in Large-scale Networks. In Proc. of SIGKDD, 2009.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In Proc. of NeurIPS, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. In Proc. of ICLR, 2018.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: Knowledge Graph Attention Network for Recommendation. In Proc. of SIGKDD, 2019.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In Proc. of ICML, 2019.

Yujun Yan, Jiong Zhu, Marlena Duda, Eric Solarz, Chandra Sripada, and Danai Koutra. GroupINN: Grouping-based Interpretable Neural Network for Classification of Limited, Noisy Brain Data. In Proc. of SIGKDD, 2019.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. In Proc. of NeurIPS, 2021.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep Learning on Graphs: A Survey. IEEE Transactions on Knowledge and Data Engineering, 2020.

Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph Neural Networks for Graphs with Heterophily: A Survey, 2022.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In Proc. of NeurIPS, 2020.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph Neural Networks with Heterophily. In Proc. of AAAI, 2021.