J Inf Process Syst, Vol.19, No.6, pp.803~816, December 2023
https://doi.org/10.3745/JIPS.04.0295

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Gradient-Based Explanation Method for Node Classification Using Graph Convolutional Networks

Chaehyeon Kim[1], Hyewon Ryu[2], and Ki Yong Lee[1,*]

Abstract

Explainable artificial intelligence is a method that explains how a complex model (e.g., a deep neural network) yields its output from a given input. Recently, graph-type data have been widely used in various fields, and diverse graph neural networks (GNNs) have been developed for graph-type data. However, methods to explain the behavior of GNNs have not been studied much, and only a limited understanding of GNNs is currently available. Therefore, in this paper, we propose an explanation method for node classification using graph convolutional networks (GCNs), which is a representative type of GNN. The proposed method finds out which features of each node have the greatest influence on the classification of that node using GCN. The proposed method identifies influential features by backtracking the layers of the GCN from the output layer to the input layer using the gradients. The experimental results on both synthetic and real datasets demonstrate that the proposed explanation method accurately identifies the features of each node that have the greatest influence on its classification.

Keywords

Explainable Artificial Intelligence, Graph Convolutional Network, Gradient-based Explanation

1. Introduction

Recently, the use of deep learning models has increased significantly in various fields including natural language processing, computer vision, and speech recognition. However, deep learning models usually have complex structures, so it is difficult to understand how or why a deep learning model derived such a prediction. Explainable artificial intelligence (XAI) is an effort to convert the black box model to the white box model, which is a technology that explains the behavior of a deep learning model in a way that humans can understand [1]. In XAI, the word explainability literally means that it can explain what is happening inside the deep learning model. One of the important purposes of XAI is to ensure the reliability of the model. For instance, suppose that we have a deep learning model that predicts whether a patient has diabetes or not based on the patient's features. We can say that the model is at least reliable if the model uses the patient's medical conditions such as weight and blood pressure to predict the patient's diabetes. However, the model would be unreliable if features such as patient's address or name played an important role in the prediction.

Manuscript received August 16, 2022; first revision December 19, 2022; accepted January 21, 2023.

* **Corresponding Author:** Ki Yong Lee (kiyonglee@sookmyung.ac.kr)
[1] Dept. of Computer Science, Sookmyung Women's University, Seoul, Republic of Korea (7chaeny25@sookmyung.ac.kr, kiyonglee@sookmyung.ac.kr)
[2] Dept. of Software Convergence, Sookmyung Women's University, Seoul, Republic of Korea (jg0149@sookmyung.ac.kr)

In recent, research on XAI has been actively conducted to understand the rationale behind the output of a deep learning model [2]. However, until now, most of the XAI research has been focused on convolutional neural networks (CNNs), which are mainly used for image processing. Meanwhile, graph-type data is being actively used in various fields such as chemistry, social networks, and traffic control. Graph-type data is very different from image data in that graph data is represented as nodes and edges connecting them while image data is represented as two- or three-dimensional arrays. As a result, diverse graph neural networks (GNNs) have been developed specially to analyze graph data. Since graph data and image data have different characteristics, GNNs used to analyze graph data have very different architectures from CNNs, which is mainly used to analyze image data. For example, the input of a GNN is two matrices representing a graph, where the first is a node feature matrix that represents the features of each node, and the second is an adjacency matrix that represents the connection between nodes. On the other hand, the input of a CNN is just a single two- or three-dimensional array. For this reason, most of the XAI methods, which are mainly developed for CNNs, are difficult to apply to GNNs directly.

Therefore, in this paper, we propose an explanation method for graph convolutional networks (GCNs) [3], which is a representative type of GNN. More specifically, after a GCN classifies each node in a graph based on the features of that node, the proposed method finds out which features of that node have the greatest influence on its classification. For that purpose, the proposed method numerically computes the influence of each feature on the node classification. To compute the influence of each feature, the proposed method backtracks the gradient of each feature from output layer to the input layer of the GCN layer by layer. The proposed method then computes the influence of each feature by aggregating all the gradients of that feature. In this way, we can identify which features had the greatest influence on the node classification. The experimental results on both synthetic and real datasets demonstrate that the proposed method can accurately identify the features that have the greatest influence on node classification using GCN. Fig. 1 shows the concept of the proposed method. In Fig. 1, the letter within each node in the graph (i.e., A, B, C) represents its class (i.e., label) and the list of boxes next to each node represents its features. After a GCN classifies each node into A, B, or C, the proposed method computes the influence of each feature of each node on that node's classification. In Fig. 1, the larger the influence, the darker the color of the feature.

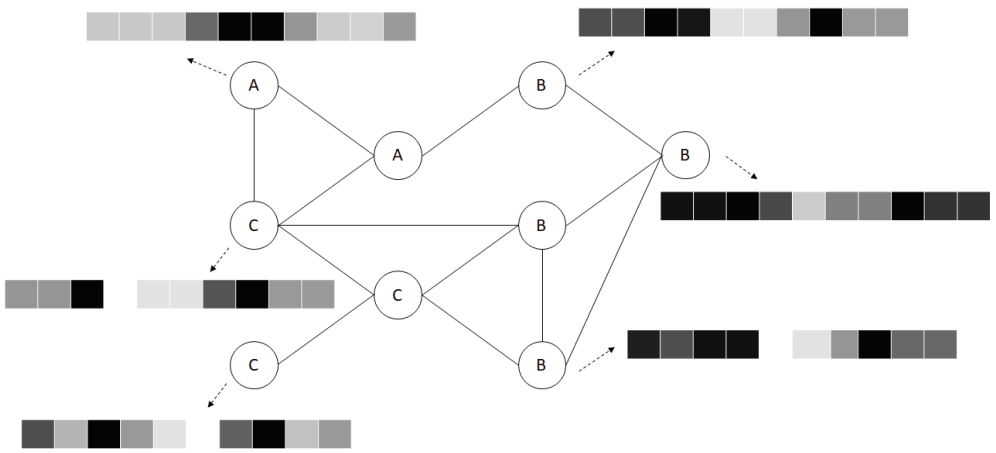


Fig. 1. The concept of the proposed method.

The rest of the paper proceeds as follows. Section 2 briefly describes GCN and present related work on XAI. Section 3 describes the proposed explanation method for identifying the features that have the greatest influence on the node classification using GCN. Section 4 presents experimental results on both synthetic and real datasets to demonstrate the effectiveness of the proposed explanation method. Section 5 concludes the paper.

2. Related Work

2.1 Graph Convolutional Network

Graphs can express various information in real life, such as social networks and protein structures, which consist of a set of nodes and a set of edges connecting the nodes. A graph is typically represented by two matrices, where the first is a node feature matrix representing the features of each node and the second is an adjacency matrix that represents the connection between nodes. The GNN is a deep learning model used to learn on graph-structured data. To this end, a GNN extracts the features of each node using the node itself and its neighboring nodes. Inspired by CNN, which has been successfully used in computer vision, researchers have begun to apply convolutional operations to graphs.

GCN is a representative type of GNN that applies convolutional operations on graphs. Fig. 2 shows the architecture of GCN. After passing through a graph convolutional layer (i.e., Gconv), the node feature matrix is updated by reflecting the information of neighboring nodes to the features of each node. Eq. (1) represents the operation of a graph convolutional layer.

$$H^{(l+1)} = \sigma\left(AH^{(l)}W + B\right). \tag{1}$$

Let $X \in \mathbb{R}^{n\times d}$ be the input node feature matrix, where n is the number of nodes and d is the number of features of a node. Here, $H^{(l)} \in \mathbb{R}^{n\times d}$ represents the node feature matrix after passing through the l^{th} graph convolutional layer. $H^{(l)}$ also corresponds to the hidden state of the l^{th} graph convolutional layer. We assume that $H^{(0)} = X$. $A \in \mathbb{R}^{n\times n}$ is the input adjacency matrix. $W \in \mathbb{R}^{d\times d}$ and $B \in \mathbb{R}^{n\times d}$ represent learnable parameters, which are shared across all graph convolutional layers. σ is an activation function, which is assumed to be ReLU in this paper. $H^{(l+1)} \in \mathbb{R}^{n\times d}$ represents the node feature matrix updated by the $(l+1)^{th}$ graph convolutional layer to reflect the features of neighboring nodes in $H^{(l)}$ to each node. Note that by multiplying the adjacency matrix A, the features of each node in $H^{(l)}$ is updated using only the features of its neighboring nodes in $H^{(l)}$. As shown in Fig. 2, there can be multiple graph convolutional layers, and as the number of graph convolutional layers increases, the features of nodes farther away are reflected into the features of each node [4]. From the last graph convolutional layer, we can obtain the final node feature matrix, which can be used for a downstream task such as node classification. For example, a multilayer perceptron (MLP) and softmax layer can be additionally used to classify each node based on the final node feature matrix.

2.2 Explainable Artificial Intelligence Methods

XAI refers to techniques to make the prediction results of a deep learning model understandable and reliable. Existing XAI methods can be largely divided into three categories: (1) activation-based methods

(ABMs), (2) backpropagation-based methods (BBMs), and (3) perturbation-based methods (PBMs), according to how the method determines which part of the input data influenced the model's prediction. This section briefly describes each category of XAI methods.

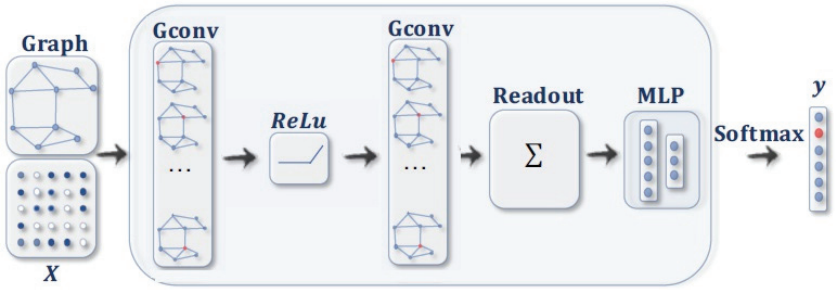


Fig. 2. The architecture of GCN [4].

2.2.1 Activation-based methods

ABMs visualize which part of the input image influenced the model's prediction. ABMs use activation values (or feature maps) in the final convolutional layer to measure the importance of the input features. A class activation map (CAM) [5] and gradient-class activation map (Grad-CAM) [6] are two representative methods of ABMs, which are developed for CNNs. Given a CNN with a global average pooling (GAP) layer, CAM identifies important class-specific features using the GAP layer, which outputs the average value of each feature map. CAM uses the output values of the GAP layer to obtain a class activation map, which shows which regions in the image contributed the most to the predicted class. However, CAM is not available for models without the GAP layer. Grad-CAM solves this problem by utilizing the gradients calculated in the backpropagation process. As a result of its execution, Grad-CAM displays the importance of pixels in the image to classification. To do so, Grad-CAM leverages the gradients flowing into the last convolutional layer of the CNN. Grad-CAM uses the gradient and feature map of the convolutional layer to form a heat map of the input image.

2.2.2 Backpropagation-based methods

BBMs compute the importance of each pixel in the input image to classification by backtracking the prediction results from the output layer to the input layer in the deep learning model. Layer-wise relevance propagation (LRP) [7] is a representative method of BBMs, which shows which neurons in the deep learning model had the largest influence on making such a decision. Given an input of d-dimensions, LRP calculates the contribution (or influence) of each neuron on deriving the output. Fig. 3 shows the overall process of LRP. As shown in Fig. 3, given the output of a model, the relevance is redistributed from the output layer to the input layer in a backpropagation fashion. Finally, LRP obtains the relevance score of each pixel, which represents how much the pixel affects the output or how much the change in the pixel causes a change in the output. In LRP, each neuron must have a certain degree of relevance and the total relevance score for each layer must be preserved. After obtaining the relevance score of each pixel, which represents the contribution of that pixel for classification, LRP generates a heap map where each pixel is associated with its relevance score to show the contribution of each pixel to the model's final output.

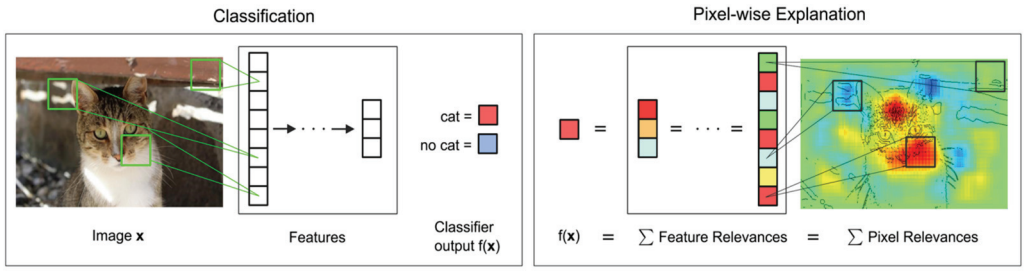


Fig. 3. The architecture of LRP [7].

2.2.3 Perturbation-based methods (PBMs)

PBMs find out the important features for classification by assessing how the predicted value will change according to the perturbation of the input. Local interpretable model-agnostic explanations (LIME) [8] is a representative model of PBMs. LIME is an algorithm that learns an interpretable model locally around the prediction. As shown in Fig. 4, LIME determines how the prediction will change by applying perturbations to the input value to learn the behavior of the underlying model. LIME assigns the weight of a training instance according to the similarity to the instance to be explained, and learns a linear regression model with lasso regularization to select k important features. In this way, LIME decomposes the input image into small units to be interpretable, and the divided regions are combined into a single image that the model can best classify.

As we emphasized, most of the XAI methods have focused on CNN [9]. However, because GNNs have very different architectures from CNN, it is difficult to apply existing XAI methods to GNNs directly. Recently, XAI studies on GNN models have been attempted, but only very limited studies have been conducted for GCN so far [10]. Although [11] has proposed the initial idea of explaining node classification in GCN, the authors of [11] presented only a limited explanation method, which is difficult to apply to general graphs. Therefore, to date, there are only limited explanation methods for node classification in GCN, which motivates our work.

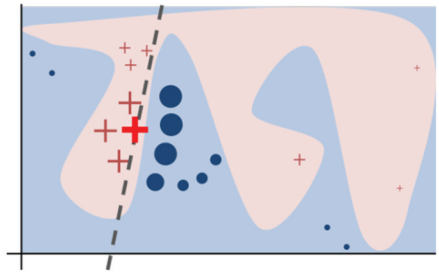


Fig. 4. The intuition of LIME [8].

3. Proposed Method

3.1 Overview

In this paper, we propose an explanation method for node classification using CGN, which identifies

the features of each node that have the greatest influence on the classification result for that node. Fig. 5 shows the overall flow of the proposed explanation method. The proposed explanation method mainly consists of two steps: (1) node classification step using an GCN and (2) explanation step for the node classification. As shown in Fig. 5, the second step backtracks the GCN from the final classification result to the first graph convolutional layer to measure the influence of each node feature on the classification result. In the next subsections, we describe each step in detail.

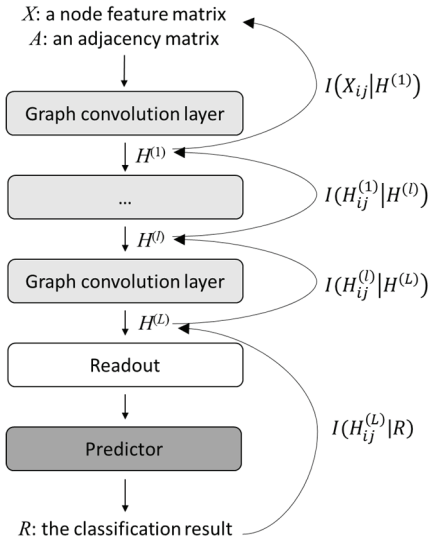


Fig. 5. The overall flow of the proposed explanation method.

3.2 Node Classification using GCN

GCN is a type of graph neural networks that applies convolutional operations to graph-structured data. As described in Section 2.1, the input of an GCN is two matrices. The first is a node feature matrix $X \in \mathbb{R}^{n \times d}$ and the second is an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where n is the number of nodes and d is the number of features of a node. The detailed process of classifying each node in a given graph using GCN is as follows. Given two matrices X and A representing the node features and topology of a given graph respectively, the GCN extracts the latent features of each node through several graph convolutional layers. Each graph convolutional layer collects the information of the nodes connected to each node, and updates the features of each node using the information of its neighboring nodes. Fig. 5 shows an example of a GCN, which contains three graph convolutional layers. Eq. (2) represents the operation performed in each graph convolutional layer in a GCN.

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right). \tag{2}$$

As in the description of GNN in Section 2.1, $H^{(l)} \in \mathbb{R}^{n \times d}$ refers to the hidden state of the lth graph convolutional layer, which represents the updated node feature matrix using up to the lth graph convolutional layer. $\tilde{A}$ is $A + I_N$, where I_N is the identity matrix. Thus, $\tilde{A}$ represents the modified adjacency matrix where each node is connected to itself. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, which is a diagonal matrix

representing the degree of each node. W is the learnable weight parameters, and σ is a nonlinear activation function such as ReLU. By computing $\tilde{D}^{-1/2}\,\tilde{A}\,\tilde{D}^{-1/2}$, $\tilde{A}$ is normalized by the number of edges connected to each node. Therefore, Eq. (2) creates $H^{(l+1)}$ by updating the features of each node in $H^{(l)}$ with the weighted sum of the features of that node and the features of the nodes connected to the node.

After obtaining the final node feature matrix $H^{(L)} \in \mathbb{R}^{n\times d}$ through L graph convolutional layers, a readout layer converts $H^{(L)}$ into a single $n \times d$ dimensional vector. Finally, a predictor layer receives the output vector of the readout layer and outputs the probability of each node belonging to each class. For the predictor layer, we can use any layers used for classification, e.g., a fully connected layer followed by a softmax layer. In the next subsection, we present how to explain the classification results produced by the GCN, i.e., how to identify the most important features for classifying each node using GCN.

3.3 Explanation of Node Classification Results

Once all the nodes of the graph are classified by the GCN as described in Section 3.2, the next step is to backtrack the features that had the greatest influence on the classification result of each node. For this purpose, the proposed method measures the *influence* of each element in X on the final classification results. Let X_{ij}be the element in the ith row ($i = 1, \dots, n$) and jth column ($j = 1, \dots, d$) of X. That is, X_{ij} represents the jth feature of the ith node. Let $I(X_{ij})$ be the influence of X_{ij} on the final classification results. For all X_{ij}, the proposed method calculates $I(X_{ij})$ by backtracking the layers of the GCN. Now we describe how to calculates $I(X_{ij})$.

Let $H_{ij}^{(l)}$ be the element in the ith row and jth column of $H^{(l)}$, which represents the jth feature of the ith node after passing through to the l^{th} graph convolutional layer. Let $H^{(L)}$ be the final node feature matrix output by the last graph convolutional layer (i.e., the Lth graph convolutional layer). Let $R \in \mathbb{R}^{n\times c}$ be the final classification results output by the GCN, where c is the number of classes. R_{ij} represents the probability of the ith node belonging to the jth class.

First, we measure the influence of each element $H_{ij}^{(L)}$ on the final classification results R, which is denoted by $I(H_{ij}^{(L)}|R)$, as follows:

$$I(H_{ij}^{(L)}|R) = \text{sum}(\frac{\partial R}{\partial H_{ij}^{(L)}}). \tag{3}$$

That is, we measure the influence of $H_{ij}^{(L)}$ on R as the rate of how much the values of R changes when the value of $H_{ij}^{(L)}$ changes, which can be expressed as the gradient of R with respect to $H_{ij}^{(L)}$. Because $\partial R/\partial H_{ij}^{(L)}$ is an $n \times c$ matrix, we use sum() operator to sum all the elements in $\partial R/\partial H_{ij}^{(L)}$.

Next, we measure the influence of each element $H_{ij}^{(l)}$ on the next node feature matrix $H^{(l+1)}$, which is denoted by $I(H_{ij}^{(l)}|H^{(l+1)})$, as follows:

$$I(H_{ij}^{(l)}|H^{(l+1)}) = \text{sum}(\frac{\partial H^{(l+1)}}{\partial H_{ij}^{(l)}}) \tag{4}$$

Similar to Eq. (3), we measure the influence of $H_{ij}^{(l)}$ on the next node feature matrix $H^{(l+1)}$ as the gradient of $H^{(l+1)}$ with respect to $H_{ij}^{(l)}$. Also in this case, because $\partial H^{(l+1)}/\partial H^{(l+1)}$ is an $n \times d$ matrix, we use sum() operator to sum all the elements in $\partial H^{(l+1)}/\partial H^{(l+1)}$. Using Eq. (4), we can measure the influence of $H_{ij}^{(l)}$ on $H^{(l+1)}$ after $H^{(l+1)}$ is created from $H^{(l)}$ through the *l*th graph convolutional layer.

Finally, we calculate $I(X_{ij})$ using Eq. (3) and (4) as follows:

$$I(X_{ij}) = I\left(X_{ij}\middle|H^{(1)}\right) \cdot I\left(H_{ij}^{(1)}\middle|H^{(2)}\right) \cdot \ldots \cdot I\left(H_{ij}^{(L-1)}\middle|H^{(L)}\right) \cdot I(H_{ij}^{(L)}|R). \tag{5}$$

Eq. (5) means that the influence of X_{ij} on the final classification results is calculated by multiplying all the influence of the *j*th feature of the *i*th node on the output of the next layer. In other words, we multiply $I\left(X_{ij}\middle|H^{(1)}\right)$, $I\left(H_{ij}^{(1)}\middle|H^{(2)}\right)$, …, $I(H_{ij}^{(L)}|R)$ to obtain the total influence of the *j*th feature of the *i*th node on the final classification results. Recall that $X_{ij} = H^{(0)}$. By using Eq. (5), we can obtain the influence of X_{ij} on the final classification results output by the GCN.

Although we use sum() operator in Eqs. (3) and (4), we can also use avg() operator in Eqs. (3) and (4), which gets the average of all the elements in a matrix. In this case, we use the following equations instead of Eqs. (3) and (4).

$$I(H_{ij}^{(L)}|R) = \text{avg}\left(\frac{\partial R}{\partial H_{ij}^{(L)}}\right), \tag{6}$$

$$I(H_{ij}^{(l)}|H^{(l+1)}) = \text{avg}\left(\frac{\partial H^{(l+1)}}{\partial H_{ij}^{(l)}}\right). \tag{7}$$

If we use Eq. (3) and (4), the influence of a feature on the next node feature map is calculated by summing all its influences on all elements in the next node feature map. On the other hand, if we use Eq. (6) and (7), the influence of a feature on the next node feature map is calculated as the average of its influence on all elements in the next node feature map. We present the effect of using two methods in Section 4.

4. Experiments

4.1 Experimental Setup

In this section, we report experimental results of the proposed explanation method. We investigated how accurately the proposed method identifies the features that have a large influence on the classification of each node in the graph. We implemented the proposed method using PyTorch. In order to confirm that the proposed method accurately finds out the features that had the greatest influence on the classification of each node, we used the following datasets:

1) Synthetic dataset: We synthetically generated a graph with 75 nodes, where each node has 30 features. Thus, the node feature matrix X has a size of 75×30, and the adjacency matrix A has a size of 75×75. We generated this synthetic dataset to intentionally create important features that

affect node classification. More specifically, for the 1st to 25th nodes, we set the values of the 1st to 10th features of them to 1 and then set the labels of these nodes to 'A'. Similarly, we set the values of the 11st to 20th features of the 26th to 50th nodes to 1 and then set the labels of these nodes to 'B'. Finally, we set the values of the 21st to 30th features of the 51th to 75th nodes to 1 and then set the labels of these nodes to "C." Therefore, in classifying the 1st to 25th nodes, the 1st to 10th features play the most important role, while the 11st to 20th features play the most important role in classifying the 26th to 50th nodes. Similarly, the 21st to 30th features play the most important role in classifying the 51th to 75th nodes.

2) Real dataset: We used the Cora dataset [12], which is a real scientific publication citation network dataset. The Cora dataset consists of 2,708 machine learning publications classified into one of the seven classes (e.g., neural network, reinforcement learning, etc.). The citation network consists of 2078 nodes and 5429 edges, where each edge represents the citation relationship between two papers. Each publication has 1433 features, where each feature indicates the absence (0) or presence (1) of the corresponding word from the dictionary. Consequently, the Cora dataset is represented by the node feature matrix X of a size of 2708×1433 and the adjacency matrix A of a size of 2708×2708.

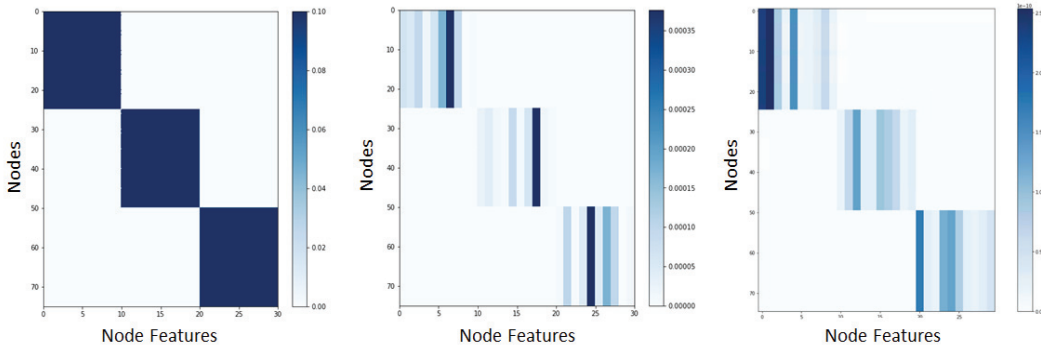
For both synthetic and real datasets, we split the dataset into a training set, a validation set and a test set. We used 65%, 15% and 20% of the dataset as the training, validation and test set, respectively. For all graph convolutional layers, we used ReLU as the activation function. For the GCN model, we used 3 graph convolution layers and trained the GCN model with the learning rate of 0.01 and dropout rate of 0.1. In the experiments, we evaluated whether the proposed model accurately finds the node features that affect the node classification results most. For this goal, we performed the following experiments: (1) Given a graph, we classify each node in the graph using the GCN, (2) After the GCN classifies each node, we use the proposed method to calculate the influence of each feature in X on the classification results. In the next subsection, we present the experimental results of the proposed method for finding the important features for node classification.

4.2 Experimental Results

4.2.1 Experimental results on synthetic dataset

In this section, we report the experimental results on the synthetic dataset, following a method similar to an evaluation method mentioned in [13]. In this experiment, we used the synthetic dataset described in Section 4.1. Fig. 6 shows the visualization of the experimental results on the synthetic dataset, where the larger the element value, the darker the color, and the smaller the element value, the lighter the color. Fig. 6(a) visualizes the input node feature matrix X of the synthetic dataset. As described in Section 4.1, for the 1st to 25th nodes (whose label is 'A'), the 1st to 10th features have a value of 1, while the other features have a value of 0. Meanwhile, for the 26th to 50th nodes (whose label is 'B'), the 11st to 20th features have a value of 1, while the other features have a value of 0. Finally, for the 51th to 75th nodes (whose label is 'C'), the 21st to 30th features have a value of 1, while the other features have a value of 0. Fig. 6(b) visualizes $I(X_{ij})$, which is the influence of X_{ij} on the final classification results, for all X_{ij} $(i = 1, \dots, 75, j = 1, \dots, 30)$, which are calculated by the proposed method using Eq. (3) and (4). On the other hand, Fig. 6(c) visualizes $I(X_{ij})$ calculated by the proposed method using Eq. (6) and (7). In Fig.

6(b) and (c), we can see that $I(X_{ij})$ calculated by the proposed method accurately identifies the features that affect the classification results. That is, for the 1st to 25th nodes, $I(X_{ij})$ of the 1st to 10th feature is high, which is consistent with the fact that only the 1st to 10th features of the 1st to 25th nodes have a value of 1. Similarly, for the 26th to 50th nodes, $I(X_{ij})$ of the 11st to 20th feature is calculated as high correctly. Also, for the 51st to 75th nodes, $I(X_{ij})$ of the 21st to 30th feature is calculated as high. In this experiment, the proposed method using Eq. (6) and (7) showed slightly better performance, but there was no significant difference in performance between Fig. 6(b) and (c). Consequently, we can conclude that the proposed method can accurately find out which features had the greatest influence on the classification results when each node in the graph is classified using GCN.



(a) (b) (c)

Fig. 6. Important features of the synthetic dataset found by the proposed method. (a) The visualization of the input node feature matrix X of the synthetic dataset. (b) The visualization of the influence of X_{ij} on the final classification results calculated by the proposed method using Eqs. (3) and (4). (c) The visualization of the influence of X_{ij} on the final classification results calculated by the proposed method using Eqs. (6) and (7).

Fig. 7 shows the influence of the important features found by the proposed method on classification. To show the influence of the important features found by the proposed method, we investigate how the test loss and accuracy change when the values of the top N features with the largest $I(X_{ij})$ are set to 0. The intention of this experiment is to show whether the test loss or accuracy actually worsen when the values of the influential features are set to 0. In Fig. 7, N refers to the number of features with the largest $I(X_{ij})$ whose values are set to 0. Note that different classes have different important features. For example, the most influential feature found by the proposed method for classes "A," "B," and "C" was the 7th feature, 18th feature, and 24th feature, respectively. Given a value of N, we selected N features for each node that had the greatest influence on its classification result and set the values of those features to 0. We then used the GCN to classify each node again and compared the test loss and accuracy with those when classifying each node using the original feature values. Fig. 7(a) and 7(b) show the experimental results when the proposed method uses Eqs. (3) and (4), and Eqs. (6) and (7), respectively. As shown in Fig. 7, the test loss gradually decreases as the values of a larger number of influential features are set to 0. This implies that that the influential features identified by the proposed method actually have some impact on node classification. However, note that the accuracy remains at 1 even when the values

of the top 1 to 5 features with the largest $I(X_{ij})$ are set to 0. This is because the number of important features is 10 for each class and the important features are mutually exclusive for each class, setting the values of 1 or 5 of these features to 0 does not affect the classification. Nevertheless, considering the fact that the test loss increased significantly from 0.7742 to 1.0174 or 1.0812 even when only one feature identified by the proposed method was set to 0, we can confirm that the proposed method can find out accurately which features had the greatest influence on node classification.

	Test Loss	Accuracy
Original	0.7742	1
N=1	1.0174	1
N=5	1.0936	1

(a)

	Test Loss	Accuracy
Original	0.7742	1
N=1	1.0182	1
N=5	1.0929	1

(b)

Fig. 7. The influence of the important features found by the proposed method when the original features values are 1 and 5. (a) The experimental results when the proposed method uses Eqs. (3) and (4). (b) The experimental results when the proposed method uses Eqs. (6) and (7).

4.2.2 Experimental results on real dataset

In this section, we report the experimental results on the real dataset described in Section 4.1. We performed the experiment in exactly the same way as in Section 4.2.1. However, in the case of the real dataset, features with a value of 1 do not represent important features. Thus, instead of visualizing the experimental results, we show the 25 most important features found by the proposed method for each class in Table 1.

Table 1 shows the top 25 node features that have most influence on the classification of each of the 7 classes. For example, the most influential feature for “Theory” papers was the 168th feature and the most influential feature for “Neural Network” papers was the 619th feature. We can interpret that the words corresponding to these features played a large role in determining the class of each paper.

As in Section 4.2.1, we also analyzed the influence of the important features found by the proposed method on classification. Fig. 8 shows how the test loss and accuracy change when the values of the top N features with the largest $I(X_{ij})$ are set to 0. The experimental method is exactly the same as that of Section 4.2.1. That is, given a value of *N*, we selected *N* features for each node that had the greatest influence on its classification result and set the values of those features to 0. We then used the GCN to classify each node again and compared the test loss and accuracy with those when classifying each node using the original feature values. Fig. 8(a) and 8(b) show the experimental results when the proposed method uses Eqs. (3) and (4), and Eqs. (6) and (7), respectively. In Fig. 8, we can see that both the test loss and accuracy gradually decrease as the values of a larger number of influential features are set to 0. Considering that only 15 or 30 features out of 1433 are set to 0, Fig. 8 implies that the influential features identified by the proposed method actually have a significant impact on node classification. Consequently, we can conclude again that the proposed method can find out accurately which features had the greatest influence on node classification even in the case of the real dataset.

Table 1. Important features of the real dataset found by the proposed method.

Class	The top 25 important features found by the proposed method
Theory	168, 0, 380, 723, 1208, 252, 1132, 1209, 184, 514, 725, 958, 1024, 1198, 1263, 1339, 1351, 88, 203, 337, 483, 755, 852, 1042, 1257
Neural Networks	619, 0, 724, 1170, 171, 820, 1273, 1389, 167, 456, 474, 587, 698, 1097, 1187, 1328, 1351, 168, 218, 565, 860, 1094, 1171, 1198, 1330
Case Based	130, 0, 750, 1163, 223, 507, 758, 1131, 1219, 1266, 1328, 4, 97, 192, 299, 666, 749, 1071, 1177, 1325, 552, 623, 647, 757, 810
Reinforcement Learning	38, 1178, 801, 96, 1322, 1177, 474, 302, 89, 213, 1286, 1209, 4, 0, 97, 229, 507, 513, 581, 779, 819, 1263, 100, 422, 596
Probabilistic Methods	19, 0, 1426, 1177, 1290, 1143, 168, 93, 972, 551, 211, 507, 526, 1198, 874, 339, 469, 610, 648, 661, 808, 826, 1209, 1423, 135
Rule Learning	68, 0, 380, 723, 1208, 252, 1132, 1209, 184, 514, 725, 958, 1024, 1198, 1263, 1339, 1351, 88, 203, 337, 483, 755, 852, 1042, 1257
Genetic Algorithms	140, 0, 988, 1355, 126, 225, 495, 639, 1149, 1177, 1267, 1340, 194, 284, 409, 507, 718, 778, 1077, 1291, 1386, 395, 722, 945, 1072

	Test Loss	Accuracy
Original	0.5148	0.8620
N = 15	0.6014	0.8280
N = 30	0.6027	0.8160

(a)

	Test Loss	Accuracy
Original	0.5148	0.8620
N = 15	0.5926	0.8240
N = 30	0.6381	0.8040

(b)

Fig. 8. The influence of the important features found by the proposed method when the original features values are 15 and 30. (a) The experimental results when the proposed method uses Eqs. (3) and (4). (b) The experimental results when the proposed method uses Eqs. (6) and (7).

5. Conclusion

In this paper, we proposed an explanation method for GCN, which identifies which node features had the greatest influence on classifying each node in the graph using GCN. For this purpose, for each layer in the GCN, the proposed method calculates the influence of each node feature on the output of that layer using the gradient. The proposed then aggregates all the influences of each node feature across all the layers of the GCN to obtain the total influence of each node feature on the final node classification results. By using the proposed method, we can explain why a node is classified into a certain class using the values of its influential features. The experimental results on both the synthetic and real datasets demonstrate that the proposed method accurately finds out the features that determine the class of each node in the graph. Therefore, the proposed method can be used successfully to determine the cause of the classification result of a node when GCN is used for classification. In future work, we will consider not only node classification but also more diverse tasks such as graph classification and subgraph classification using various types of GNNs.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2021R1A2C1012543). This paper is the extended version of the Annual Spring Conference of KIPS (ASK 2022) held in Seoul, Republic of Korea dated May 19-21, 2022 [11].

References

[1] A. B. Arrieta, N. Diaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, et al., "Explainable Artificial Intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82-115, 2020. https://doi.org/10.1016/j.inffus.2019.12.012

[2] S. Lee, S. Lee, D. Chang, M. H. Song, J. Y. Kim, and S. Lee, "Explainable machine learning based a packed red blood cell transfusion prediction and evaluation for major internal medical condition," *Journal of Information Processing Systems*, vol. 18, no. 3, pp. 302-310, 2022. https://doi.org/10.3745/JIPS.04.0243

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017 [Online]. Available: https://arxiv.org/abs/1609.02907.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4-24, 2021. https://doi.org/10.1109/TNNLS.2020.2978386

[5] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 2921-2929. https://doi.org/10.1109/CVPR.2016.319

[6] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, 2017, pp. 618-626. https://doi.org/10.1109/ICCV.2017.74

[7] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Muller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS One*, vol. 10, no. 7, article no. e0130140, 2015. https://doi.org/10.1371/journal.pone.0130140

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1135-1144. https://doi.org/10.1145/2939672.2939778

[9] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (XAI): toward medical XAI," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793-4813, 2021. https://doi.org/10.1109/TNNLS.2020.3027314

[10] J. Hu, T. Li, and S. Dong, "GCN-LRP explanation: exploring latent attention of graph convolutional networks," in *Proceedings of 2020 International Joint Conference on Neural Networks (IJCNN)*, Glasgow, UK, 2020, pp. 1-8. https://doi.org/10.1109/IJCNN48605.2020.9207639

[11] C. Kim and K. Y. Lee, "A gradient-based explanation method for graph convolutional neural networks," in *Proceedings of Annual Spring Conference of KIPS (ASK 2022)*, 2022, pp. 670-673.

[12] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93-93, 2008. https://doi.org/10.1609/aimag.v29i3.2157

[13] G. Vilone and L. Longo, "Notions of explainability and evaluation approaches for explainable artificial intelligence," *Information Fusion*, vol. 76, pp. 89-106, 2021. https://doi.org/10.1016/j.inffus.2021.05.009

Chaehyeon Kim **https://orcid.org/0000-0002-6443-1286**

She received her B.S. in both Statistics and Computer Science from Sookmyung Women's University, Seoul, Republic of Korea, in 2021. She is now M.S. student in the Department of Computer Science of degrees at Sookmyung Women's University, Seoul, Korea. Her current research interests include databases, data mining, and deep learning.

Hyewon Ryu **https://orcid.org/0000-0001-5799-3478**

She received her B.S. in both Statistics and Software Convergence from Sookmyung Women's University, Seoul, Republic of Korea, in 2022. She is now M.S. student in the Department of Software Convergence of degrees at Sookmyung Women's University, Seoul, Korea. Her current research interests include data mining, deep learning and big data.

Ki Yong Lee **https://orcid.org/0000-0003-2318-671X**

He received his B.S., M.S., and Ph.D. degrees in Computer Science from KAIST, Daejeon, Republic of Korea, in 1998, 2000, and 2006, respectively. From 2006 to 2008, he worked for Samsung Electronics, Suwon, Korea as a senior engineer. From 2008 to 2010, he was a research assistant professor of the Department of Computer Science at KAIST, Daejeon, Korea. He joined the faculty of the Division of Computer Science at Sookmyung Women's University, Seoul, in 2010, where currently he is a professor. His research interests include database systems, data mining, and big data.