On the modelling and impact of negative edges in graph convolutional networks for node classification

On the modelling and impact of negative edges in graph convolutional networks for node classification

Trang Dinh
University of Manchester
thutrang.dinh@manchester.ac.uk

Luis Ospina-Forero
University of Manchester
luis.ospina-forero@manchester.ac.uk

Julia Handl
University of Manchester
julia.handl@manchester.ac.uk

Abstract

Signed graphs are important data structures to simultaneously express positive and negative relationships. Their application ranges from structural health monitoring to financial models, where the meaning and properties of negative relationships can play a significant role. In this paper, we provide a comprehensive examination of existing approaches for the integration of signed edges into the Graph Convolutional Network (GCN) framework for node classification. Here, we use a combination of theoretical and empirical analysis to gain a deeper understanding of the strengths and limitations of different mechanisms and to identify areas for possible improvement. We compare six different approaches to the integration of negative link information within the framework of the simple GCN. In particular, we analyze sensitivity towards feature noise, negative edge noise and positive edge noise, as well as robustness towards feature scaling and translation, explaining the results obtained on the basis of individual model assumptions and biases. Our findings highlight the importance of capturing the meaning of negative links in a given domain context, and appropriately reflecting it in the choice of GCN model. Our code is available at `https://github.com/dinhtrang24/Signed-GCN`.

1 Introduction

Signed graphs are widely used to represent interactions in systems such as social networks [8, 22, 25, 20], biological networks [15], international relations networks [5], and financial networks [11]. Unlike their unsigned counterparts, which only represent positive relationships, signed graphs incorporate both positive and negative edges to capture a more nuanced set of interactions.

In recent years, a number of graph representation learning techniques have been proposed to realize the potential of the graph data in a machine learning context. One prominent approach is the Graph Convolutional Network (GCN) [19], which has demonstrated its effectiveness in various graph-based tasks on unsigned networks. However, adapting GCNs to signed networks requires consideration of the distinct meaning of positive and negative edges. There are two key considerations when designing a signed GCN approach: (1) allowing for the differential treatment of negative and positive edges, as they represent different types of relations; whilst simultaneously (2) considering the special meaning of signs, which suggests some potential interdependency in their correct treatment. There is potential conflict between these ambitions, and simultaneously addressing both is not straightforward. Consequently, existing approaches in the literature have tended to focus on one of the two aspects.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

The Relational Graph Convolutional Network (R-GCN), introduced by [33], although proposed for graphs with an arbitrary number of edge types, can be deployed to model positive and negative edges as distinct types of relations, allowing for the independent (or partially aligned) learning of relation-specific weights. This approach enables the model to capture the distinct patterns and relative importance of each relation, but does not explicitly address the meaning of two "opposed" edge types. On the other hand, methods such as the Signed Graph Convolutional Network (SGCN) [4] and Signed Network Embedding via Graph Attention (SNEA) [28] leverage structural social theories (specifically, balance theory) as their organizing principle to integrate positive and negative edge information. Nonetheless, methods based on balance theory view positive and negative edges as relative interacting components rather than "opposed" edge types. Lastly, Laplacian-GCN (Lap-GCN), introduced by [23], which tries to tackle both considerations, utilizes the concept of antipodal proximity to handle graphs containing negative edges. However, the approach is not invariant to translations in the feature space.

Contributions Here, we provide a comprehensive examination of existing approaches for the integration of signed edges into the GCN framework for node classification. Our objective is to gain a deeper understanding of the strengths and limitations of different mechanisms and to identify areas for possible improvement. Drawing insights from this analysis, we propose two variations that attempt to explicitly capture the meaning of negative edge signs, and highlight opportunities for further extension. Our paper includes an empirical comparison of the different approaches on synthetic and real data sets, highlighting how model assumptions impact the models' relative sensitivity to aspects such as edge degree, feature transformations, and feature and edge noise.

2 Background

This work mainly considers extensions to the unsigned GCN proposed in [19], which aims to generalize the principles of Convolutional Neural Networks (CNNs) to graph-structured data. Therefore, we first start this section by providing a recap of relevant terminology and the basic principle of convolutions on graph data. Subsequently, we discuss the GCN proposed in [19], as well as existing extensions that aim to model signed networks. We include some discussion of alternatives outside of the GCN framework. These are not the primary focus of our analysis but are relevant to highlight, for wider positioning of the work.

In the following, the terms "graph" and "network" are used interchangeably when referring to graph objects. Meanwhile, the term "neural network" will be used when referring to the associated machine learning models.

Definition 1 (Undirected graph): A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical structure composed from a node set $\mathcal{V}$ and edge set $\mathcal{E}$, with n nodes $i \in \mathcal{V}$, and edges $(i, j) \in \mathcal{E}$ indicating the presence of an undirected edge between node i and j with $i, j \in \mathcal{V}$. This graph can be expressed by an adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{ij} = 1$ if there is an edge between node i and node j; otherwise, $\mathbf{A}_{ij} = 0$.

This definition of graphs can be extended to cases where the interactions between nodes are weighted or signed; this generalisation is simply translated to an adjacency representation $\mathbf{A}_{ij}$ which can take values other than unity, to show stronger and weaker connections.

For a graph $\mathcal{G}$ with a non-negative adjacency matrix $\mathbf{A}$, i.e. $\mathbf{A}_{ij} \geq 0 \, \forall i, j \in \mathcal{V}$, the (unsigned) Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$ is defined as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{1}$$

with the diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ given by $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$. The two versions of normalized graph Laplacians, including symmetric normalized Laplacian and random walk normalized Laplacian, are given by $\mathbf{L}_{sym}$ and $\mathbf{L}_{rw}$ respectively as follows:

$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{1/2} \tag{2}$$

$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \tag{3}$$

Definition 2 (Signed undirected graph): Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$ be a signed graph, where $\mathcal{V}$ is the set of n nodes, $\mathcal{E}^+$ denotes the set of positive edges, and $\mathcal{E}^-$ denotes the set of negative edges such that $\mathcal{E}^+ \cap \mathcal{E}^- = \phi$. This graph can also be represented by a signed adjacency matrix $\mathbf{A}$, where $\mathbf{A_{ij}} = 1$ if $(i, j) \in \mathcal{E}^+$, and $\mathbf{A_{ij}} = -1$ if $(i, j) \in \mathcal{E}^-$; otherwise, $\mathbf{A_{ij}} = 0$ and it denotes the absence of an edge.

Convolutions on graphs: The success of Convolutional Neural Networks (CNNs) in the computer vision domain has motivated the generalization of the concept of convolution to graphs. There are two distinct ways of achieving this: spatial and spectral-based approaches.

Spatial approaches define convolution directly in the vertex domain, in line with the operation of conventional CNNs. For each vertex, convolution is defined as a weighted average over all of its neighbouring nodes, where the weighting function reflects the influence of the neighbours on the target vertex [31]. The main challenge of this approach lies in the definition of convolution operations in a manner that can accommodate neighborhoods of different sizes, whilst preserving the weight sharing property of CNN. Due to their adaptability and scalability, spatial approaches have attracted significant attention [10]. For example, GraphSAGE [10] uniformly samples a fixed-size collection of neighbours, rather than the entire neighbourhood of each node, and then introduces different weighting functions to aggregate information from the sampled neighbourhood. The graph attention network (GAT) [36] incorporates a self-attention mechanism into learning the weighting function. Furthermore, several methods provide us with a general framework for designing spatial methods; for instance, Mixture model network (Monet) [31] takes convolution as the weighted average of multiple functions assigning weights for neighbours according to their pseudo-coordinates.

Spectral-based approaches define graph convolutions as filters, drawing on methodologies from graph signal processing [34]. From this perspective, the graph convolution operation can be considered a technique for decreasing noise in graph signals. Specifically, it involves transforming the graph signals into the frequency domain by calculating the eigen-decomposition of the graph Laplacian matrix, where eigenvalues are employed as frequencies and its eigenvectors are interpreted as the corresponding pure harmonics [32]. A graph signal is decomposed to its pure harmonic coefficients by the definition of the graph Fourier transform, and filters are defined as diagonal matrices with entries that correspond to the scaling factor for each frequency component. For instance, the spectral CNN [1] is the first attempt at leveraging the graph Fourier transform and defining graph convolution in the spectral domain. [3] introduced a Chebyshev polynomial approximation to localize filters and reduce the expensive computation of eigenvalues, providing a fast localized spectral filtering method. [19] further simplified ChebyNet by using a first-order approximation of the polynomial to define the graph convolution operation. The resulting GCN model, created by stacking multiple localised graph convolutional layers to create a layer-wise linear formulation, has been shown to achieve state-of-the-art performance on various graph-based tasks [19].

Graph Convolutional Network: The GCN introduced in [19] is one of several convolutional neural network architectures for learning over graphs and has shown strong performance in addressing node classification problems on unsigned graphs. The GCN algorithm can be classified as a spectral-based approach as it uses the spectral graph convolution introduced in [1] as a starting point for the model's construction.

The GCN draws on the definition of convolutions in the spectral domain, as the convolution of a signal $x \in \mathbb{R}^n$ (a feature vector of all vertices of a graph where x_i is the value of the i^{th} node) with a spectral filter g_θ (a function of the eigenvalues of the normalized graph Laplacian L_{sym}) [1, 34]. However, computing the Laplacian eigenvectors can be computationally expensive, especially for large graphs. Thus, several subsequent models, such as the Chebyshev Spectral CNN (ChebNet) [3] and the GCN [19], propose approximations and simplifications to reduce the computational complexity.

ChebNet [3] approximates the filter g_θ by taking the Chebyshev polynomials of the diagonal matrix of eigenvalues up to $\mathbf{K}^{th}$ order. The convolution of a graph signal x with the defined filter g_θ can then be written as:

$$g_\theta \star x \approx \sum_{k=0}^{\mathbf{K}} \theta' T_k(L) x \tag{4}$$

where $\star$ denotes the convolution operator, T_k represents the Chebyshev polynomials, and $\theta' \in \mathbb{R}^{\mathbf{K}}$ is a vector of Chebyshev coefficients. By recursively computing $\mathbf{K}$-localized convolutions via the polynomial approximation, the cost to filter a signal x can be reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(\mathbf{K}|\mathcal{E}|)$ [3] where $\mathcal{O}(\cdot)$ is a mathematical notation for an algorithm's complexity in terms of the input size.

[19] further simplified the model by limiting $\mathbf{K} = 1$ as multiple localised graph convolutional layers using the first-order approximation of the graph Laplacian can be stacked to create a layer-wise linear formulation. This allows a deeper architecture to capture the graph structure without being constrained by the polynomials' explicit parameterization. Furthermore, because the approximation

order is limited to one, the layer-wise linear structure is parameter-efficient and highly efficient for large-scale graphs. Approximating the largest eigenvalue $\lambda_{max} = 2$, the convolution becomes:

$$g_\theta \star x = \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})x \tag{5}$$

where θ is the only Chebyshev coefficient left.

[19] further introduces a renormalization strategy to address the problem of exploding or vanishing gradients:

$$\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \rightarrow \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} \tag{6}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and the diagonal degree matrix $\tilde{\mathbf{D}}_{ii} = \sum_{j=1}^{n} \tilde{\mathbf{A}}_{ij}$.

GCN formulation: The above definition of convolution can be generalized to a graph signal with p channels of inputs, represented as $\mathbf{X} \in \mathbb{R}^{n \times p}$. Here, each vertex within the graph is associated with a p-dimensional feature vector. By applying f spectral filters, the propagation rule of this simplified model is defined as follows:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{7}$$

where $\mathbf{H}^{(l)}$ is the activation matrix in the l-th layer, which is obtained by applying the activation function to the convolved signal matrix, with $\mathbf{H}^{(0)}$= $\mathbf{X}$. $\sigma(\cdot)$ denotes the non-linear activation function, such as $ReLU(\cdot) = \max(0, \cdot)$, and $\mathbf{W}^{(l)}$ is the trainable weight matrix in the l-th layer.

This simplified model is named GCNs, which provides the foundation of our approach in this paper. The corresponding representation vector of node i of $(l+1)$-th hidden layer is:

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{\tilde{\mathbf{A}}_{ij}}{\sqrt{\tilde{\mathbf{D}}_{ii}\tilde{\mathbf{D}}_{jj}}} \mathbf{H}_j^{(l)}\mathbf{W}^{(l)} \right) \tag{8}$$

where $\mathcal{N}_i$ denotes the set of neighbour indices of node i.

From a spatial perspective, the GCN model can be interpreted as the employment of neural message passing, where vector messages are exchanged between nodes in a graph and updated using neural networks [6]. The hidden state at the $(l+1)$-th layer of every node i is generated by a specific update rule: vector messages (reflecting representations in the previous(l)-th layer) are aggregated across the i th's node neighbourhood $\mathcal{N}_i$ and node i itself, a weighted average of these messages is taken, and subsequently adjusted using a weight matrix $\mathbf{W}^{(l)}$ and activation function σ. Thus, after the message passing, the new representation of node i is a combination of the previous representations of the target node itself and its neighbours.

Extensions of GCNs for signed graphs: Some naive approaches to handling the negative links in signed graphs are simply removing them [30, 37], ignoring edge signs initially discussed by [4] or renormalizing edges to ensure $\mathbf{A}_{ij} \in \{0, 1\}$ [27]. The attractive aspect of these approaches is of course that methods originally designed for unsigned graphs can be applied directly to signed graphs. However, by removing negative links, valuable information in the data may be overlooked. Similarly, ignoring the differentiation between positive and negative links may introduce noise, and / or lead to the loss of important insights and nuances specific to signed graph data.

More refined approaches are available in the form of the Laplacian-GCN (Lap-GCN) [23] and the Relational-GCN (R-GCN) [33]. Lap-GCN expands spectral theory to signed graphs by using a modified degree matrix to define a signed Laplacian matrix. Core to the Lap-GCN's approach is the concept of "antipodal" proximity [23], which attempts to capture the meaning of negative links by reflecting the coordinates of (negatively linked) neighbouring nodes through the origin in the feature space, in the context of message passing. The resulting reflected points, used during subsequent analysis, are called "antipodal points".

R-GCN has a more general focus on handling graphs reflecting a range of different types of relations. This model can be applied to signed graphs by treating positive and negative edges as two distinct classes. Information aggregated from positive and negative edges, as well as information from each node itself (captured through a self-loop), is learned using three separate weight matrices. This method offers a more flexible message-passing mechanism than conventional GCNs, allowing for the

independent optimization of weights for each information source. Specifically, the message passing rule of R-GCN [33] is given as follows:

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\mathbf{W}_r^{(l)}\mathbf{H}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{H}_i^{(l)}\right) \tag{9}$$

where $\mathcal{N}_i^r$ denotes the set of neighbour indices of node i regarding relation $r \in \mathcal{R}$. $c_{i,r}$ is a normalization constant that can be learned or chosen in advance, such as $c_{i,r} = |\mathcal{N}_i^r|$, i.e. to normalize by the frequency of each edge type.

Non-GCN based approaches: An alternative approach to the analysis of signed networks draws inspiration from social theories, in particular balance theory [12]. Employing balance theory to understand tensions and conflicts in interpersonal networks with both friendly and adversarial relations, [2] defines the global structure of balanced signed networks. Such networks can be partitioned into two factions where each faction contains only cooperative relationships while all inter-faction relationships are antagonistic. By studying the local and global structure of three large-scale networks, [24, 25] demonstrated that the underlying mechanism for creating edge signs in these networks complies with balance theory. Recently, balance theory has served as the key principle in designing learning frameworks to address the consideration of signed edges in a variety of learning problems [38, 17, 16, 4, 13, 28, 29, 14].

3 Message exchange along negative edges

It is thought that one of the key driver behind the GCN's remarkable performance for node classification is the exploitation of homophily [9]. Homophily, the tendency for nodes to share properties with their neighbouring nodes in the graph, is reflected via positive edges. However, the notion of homophily is not directly applicable to negative edges, as they tend to connect nodes from different classes, which are likely to have different or opposing attributes [35].

Homophily in existing approaches: Considering the approaches discussed in subsection 2, in the context of homophily, it is evident that failure to differentiate between edge sign is likely to have a negative impact on GCN performance, potentially diluting the signal accumulated along positive edges. Similarly, Lap-GCN side-steps this issue through the direct generation of antipodal points for negatively linked nodes with the implied assumption that homophily can be assumed after the completion of this transformation step. A core limitation of this approach is the limited theoretical basis for the concept of antipodal proximity: A reflection around the origin is unlikely to reflect a meaningful interpretation of negative relations in every given context - its appropriateness will fundamentally depend on the meaning of the features. Linked to this, it is evident that the generation of antipodal points is not invariant with regard to the simple translation of feature values.

As R-GCN does not directly consider the negative sign nor the meaning of such sign, it may get stuck in following the principle of homophily for each edge type, and encouraging representational similarity between nodes connected by any edge type unless the method is able to learn the implications of negative edges and adapt its strategy. This is an additional limitation in settings where negative edges are present and provide an indicator of heterophily or differentiation in class membership.

An alternative proposal: Based on the above observations, we describe two alternative mechanisms for integrating negative edges within the GCN framework. Intuitively, our approach aims to capture the specific nature of the typical association between positive and negative edges, reflecting **polarity** of the **same** relation. We translate this into modelling components in the following way: (1) the calculation of a unit vector which defines the direction of information exchange between two neighbouring nodes (independently of sign); (2) the calculation of a scale measurement that considers the distance and sign of neighbours in deciding the scale of any update. Mathematically, the GCN's update rule in our approach can be expressed as follows:

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}}\frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}}\left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij})||s_{ji}^{(l)}||\overrightarrow{u_{ji}^{(l)}}\right)\mathbf{W}^{(l)}\right) \tag{10}$$

where $\overrightarrow{u_{ji}^{(l)}}$ and $||s_{ji}^{(l)}||$ are a mathematically derived unit vector and scale that define the direction and the magnitude of the updated information from node j to node i in the l-th layer, respectively. The signed diagonal degree matrix $\tilde{\mathbf{D}}' \in \mathbb{R}^{n\times n}$ given by $\tilde{\mathbf{D}}'_{ii} = \sum_j |\tilde{\mathbf{A}}_{ij}|$, and $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$.

Our approach first incorporates the original features of the target node i, and from that, develops the updated information received from neighbouring nodes j. These updated features will be transformed using a normalized version of the absolute adjacency matrix, equivalent to the approach used in Lap-GCN. Consistently with Lap-GCN, a single weight matrix is used for updates along all edges. Differently to Lap-GCN, for a neighbouring node j that is connected to i by a negative edge, our approach generates the equivalent of an "antipodal point" using a projection through node i itself (rather than the origin). Mathematically, this is achieved by calculating the unit vector as the unit vector of the difference between feature vectors of node i and j, i.e. $\overrightarrow{u_{ji}^{(l)}} = \frac{(\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)})}{||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}||}$. It has the desired effect that updating information received along a negative edge will lead to an update that is directly opposite to the update that would have happened if that information had arrived along a positive edge. In doing so, we aim to reflect the intuitive (polar) meaning of positive and negative signs in many settings, and ensure that nodes connected by a positive edge will be pulled closer to each other (consistent with the principle of homophily), whilst those linked by negative edges will be repulsed.

Next, we propose two versions of our approach, namely Difference-GCN (or Diff-GCN) and Nonlinear-GCN (or Nlinear-GCN), which differ solely in the definition of the scale parameter, i.e. the magnitude of the update received (prior to weighting) during message passing.

Signed GCN (linear): In Diff-GCN, the scale of the updated information is defined as the magnitude of the difference between features of the target node i and the neighbouring node j. This is in line with the (unweighted) scale of updates in the traditional GCN framework (for edges with a positive edge sign, the update is equivalent to the original GCN update), and implies that nodes that are close to each other (in terms of their features/representations) have less of an impact than those that are far apart. This instantiation of the update rule in Eq. 10 is expressed as follows:

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}} \left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij}) \frac{\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}}{||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}||} ||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| \right) \mathbf{W}^{(l)} \right) \tag{11}$$

Signed GCN (non-linear): Nlinear-GCN adjusts only the scale portion of the equation to embrace the intuitive concept of "repulsion", implying that a negative edge should induce a stronger effect for those nodes that are close together (in terms of their current feature vectors/representations). To achieve this, the scale of the update remains unchanged for positive edges but, for negative edges, is set to $\frac{1}{||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| + 1}$. The appropriate instantiation of Eq. 10, capturing the correct behaviour for both types of edges in Nlinear-GCN, is thus given as:

$$\mathbf{H}_{ij}^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}} \left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij}) \frac{(\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)})}{||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}||} \left(||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| + \frac{1 - sign(\tilde{\mathbf{A}}_{ij})}{2} \right)^{sign(\tilde{\mathbf{A}}_{ij})} \right) \mathbf{W}^{(l)} \right) \tag{12}$$

Geometrical interpretation: Consider updating the representation of node A with negative links to

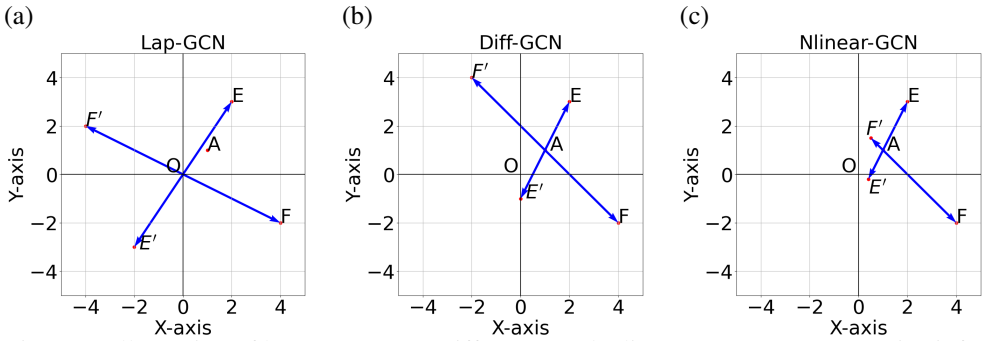

Figure 1: Illustration of how Lap-GCN, Diff-GCN, and Nlinear-GCN capture negative information given that the target node A has negative links with two neighbouring nodes E and F. E' and F' are the images of E and F, following the relevant projection.

two nodes E and F. Figure 1-a illustrates how Lap-GCN considers the respective antipodal points

E' and F' in order to update the representation of node A. In contrast, Diff-GCN and Nlinear-GCN consider a reflection through node A Figure 1-b,c. This strategy allows the message passing in Diff-GCN to be scale invariant. On the other hand, the scale parameter in Nlinear-GCN introduces a dependency w.r.t. feature scale. Nlinear-GCN inversely scales the magnitude of the "negative" information to be used in the update of node A by the distance to each respective "negative" neighbour of node A, thus allowing for larger "repulsion" of nodes closer to A. Lastly, another benefit of Diff-GCN and Nlinear-GCN is that the location of the antipodal points in the feature space is invariant to translations of the data, whilst not being the case for Lap-GCN.

4 Experimental setup

Based on the preceding discussions, we select six different approaches for empirical comparison. As simple baseline techniques, we consider the GCN ignoring edge signs (GCN), as well as the GCN applied to the positive edges only (Pos-GCN). In terms of existing formal frameworks for the integration of negative edges, we consider the use of Lap-GCN and R-GCN, as well as our new, alternative update rules introduced in Section 3 above.

Datasets: We use **synthetic data** to explicitly control and explore the impact of node degree/(edge density) and feature noise. Our generating model assumes three, equally-sized communities, and two node features which are Normally Distributed within each community. Parameters defining the generation of node features for two different scenarios are specified in the supplementary material. We generate between and within-community edges varying the average degree w.r.t. each class of edge from 1 to 24, to investigate the models' sensitivity to this aspect. In the noise-free setting, edge signs reflect the nature of the edge, with negative signs used to reflect between-community edges only. We separately control noise for each type of edge through the introduction of previously unused edges of a given class, with flipped signs, e.g. 10% neg: increase negatives edges by 10%, by adding from unobserved within community edges.

Additional experiments are conducted using **real-world networks**, specifically the **Cora** and **Citeseer** citation data sets [39]. The data consists of sparse bag-of-words feature vectors for each document (node), along with a list of citation links between documents, and a class label for each document, reflecting subject areas (see Appendix B for more details). Here, the presence/absence of citation links is modelled using undirected edges. In line with the synthetic benchmark, edge signs are set to positive for within-communities edges, and to negative for between-communities edges.

Another type of real-world network used to validate our proposed model is social networks. The **Bitcoin-OTC** network [21], which shows how users give ratings to each other based on their trust/distrust relationships, has been studied widely as benchmark signed network data [4, 29, 26, 27]. Positive (negative) links correspond to the connections with the positive (negative) values of the rating. Due to the anonymity of Bitcoin users, maintaining reputation records is crucial to prevent transactions with fraudulent and risky users. This can be considered a classification task, differentiating between trustworthy users and risky users. Table 1 summarizes the statistics of the processed undirected signed Bitcoin network. (See Appendix B for more details on how we process the Bitcoin-OTC data).

Table 1: Real networks

Datasets	Nodes	Edges	Degree	Positive degree	Negative degree	Features	Classes
Cora	2,708	5,278	3.9	3.2	0.7	1,433	7
Citeseer	3,327	4,552	2.7	2.0	0.7	3,703	6
Bitcoin-OTC	5881	21,492	7.21	6.22	1.09	5,881	2

In our synthetic benchmark, we have direct control over feature quality. To consider the impact of feature noise in the real-world data, we randomly perturb individual feature values with a specific incidence rate. The approach to investigating edge noise is equivalent in synthetic and real data sets.

Feature transformations: To understand the impact of scale/translation invariance, we conduct experiments using z-score standardization, min-max scaling and the original feature values.

Experimental setup: The learning task considered is node classification, using the graph structure and node features. Given a level of class imbalances in the real-world networks, both accuracy and F1-macro score are used to assess model performance. Our own models and three of the contestant models

(GCN, Pos-GCN, and Lap-GCN) adopt the symmetrically normalized adjacency matrix described in [19]. For R-GCN, we choose the normalization constant $c_{i,r} = |\mathcal{N}_i^r|$ recommended in [33]. Finally, to understand the impact of this normalization difference, we run additional experiments integrating an equivalent normalization within our models (Nlinear-GCN-Split and Diff-GCN Split).

For the Bitcoin-OTC data, we add the F1-binary score specifically for the minority group - risky users - due to the high imbalance between the two classes. To strengthen the comparison with prior work, another simple extension of GCN for signed graphs, Norm-GCN [27], is added as a competing model. Following [27], Norm-GCN renormalizes edge weights to [0,1] by $\tilde{e}_{ij} = e_{ij}/20 + 1/2$ where e_{ij} is the edge weight between nodes i and j.

Across all methods, we adopt optimized hyperparameters from [19], i.e. we use the weight initialization described in [7], and 16 hidden nodes. We train all models for a maximum of 300 epochs using Adam [18] with a learning rate of $1e-3$, and early stopping (stagnation over 50 epochs). For the synthetic and Bitcoin data, the training/validation/test ratio is 10/30/60, and we report average results over 30 random splits. For the real networks, we follow the data splits suggested in [39], using a training/validation/test split of 5/18/37 for Cora, and a 4/15/30 split for Citeseer. Results are reported as averages over 30 random weight initializations.

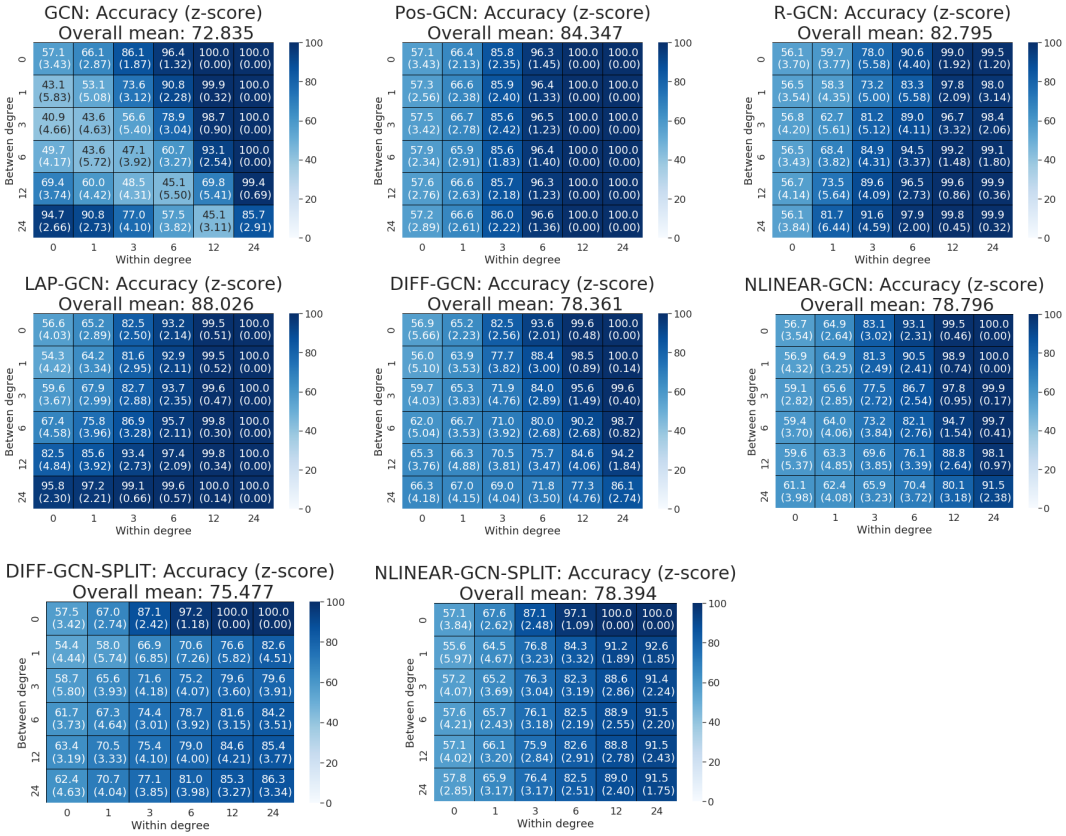

Figure 2: Accuracy of the different models on the synthetic benchmarks when varying the within and between-community degree. Averages over 30 train/validation/test splits, using z-score normalized features.

5 Results

The full set of experiments is included in Appendix C. In the following, we focus on the discussion of key findings and insights only.

Learning from shared enemies: Our experiments on synthetic data (see Figure 2) highlight an interesting phenomenon regarding GCN's performance. As expected, its performance suffers with the introduction of additional negative edges, as these weaken the signal distributed along genuine

positive edges. However, once a sufficient number of between-community edges are available, and between-community degree exceeds within-community degree, this behaviour reverses - we observe a steady increase in model performance for further increases in between-community degree.

What this indicates is the ability of the model to learn from considering "shared enemies", i.e. each node is learning not directly from its enemies' feature values, but from the fact that nodes within the same class will have negative edges to the same communities of "enemies" (and not to their own community). It is evident (and results on synthetic and real data confirm) that this signal will weaken in the context of a larger number of classes and /or in situations with low between-community degrees.

We observe the same effect benefiting R-GCN and Lap-GCN, on the synthetic data, rendering Lap-GCN, in particular, a surprisingly strong performer. R-GCN appears to find it more difficult to benefit, and, in those settings with a complete absence of within-community structure, is not able to exploit between-community edges at all. This is likely an artefact of its normalization approach, which reduced the impact of edge information relative to the node's own features.

Table 2: F1-macro score (in percentage) of the different models on the real-world citation networks for various noise settings. Averages over 30 weight initialization. The best-performing model is highlighted.

Data	Settings	GCN	Pos-GCN	R-GCN	Lap-GCN	Diff-GCN	Nlinear-GCN
Cora	No noise	79.70 ± 0.52	$\mathbf{86.52 \pm 0.42}$	84.57 ± 0.79	82.29 ± 0.79	83.07 ± 0.47	82.90 ± 0.39
	10% neg.	80.48 ± 0.50	$\mathbf{86.49 \pm 0.32}$	83.86 ± 0.80	80.67 ± 0.96	81.45 ± 0.60	81.17 ± 0.49
	30% neg.	81.56 ± 0.50	$\mathbf{86.59 \pm 0.27}$	83.76 ± 0.65	77.25 ± 1.13	78.92 ± 0.56	78.08 ± 0.98
	10% pos.	77.22 ± 0.74	$\mathbf{83.26 \pm 0.65}$	80.62 ± 0.81	79.22 ± 0.79	80.05 ± 0.87	79.73 ± 0.62
	30% pos.	72.80 ± 0.74	$\mathbf{77.15 \pm 1.09}$	75.92 ± 1.16	74.23 ± 0.86	74.83 ± 1.04	74.38 ± 0.97
	10% both	77.75 ± 0.62	$\mathbf{82.81 \pm 1.07}$	80.84 ± 1.33	77.68 ± 0.59	78.66 ± 0.83	78.06 ± 0.79
	30% both	75.53 ± 0.73	$\mathbf{77.21 \pm 0.79}$	75.14 ± 1.43	69.53 ± 1.27	71.22 ± 0.83	70.36 ± 1.07
	10% feat.	30.60 ± 8.80	36.59 ± 7.46	20.80 ± 6.80	$\mathbf{38.60 \pm 1.22}$	30.12 ± 6.81	36.94 ± 2.86
	30% feat.	16.36 ± 11.86	19.94 ± 13.67	13.33 ± 5.44	$\mathbf{44.44 \pm 4.58}$	39.80 ± 5.50	39.44 ± 9.49
Citeseer	No noise	68.21 ± 0.55	$\mathbf{73.57 \pm 0.32}$	71.14 ± 0.48	65.40 ± 0.49	68.47 ± 0.42	67.16 ± 0.36
	10% neg.	68.88 ± 0.67	$\mathbf{73.66 \pm 0.48}$	70.74 ± 0.45	63.72 ± 1.28	67.17 ± 0.59	64.77 ± 0.87
	30% neg.	70.25 ± 0.79	$\mathbf{73.52 \pm 0.31}$	70.78 ± 0.99	60.33 ± 1.27	64.19 ± 1.02	60.64 ± 1.58
	10% pos.	65.17 ± 0.79	$\mathbf{69.81 \pm 1.04}$	68.21 ± 0.67	63.40 ± 0.95	66.91 ± 0.69	64.87 ± 0.66
	30% pos.	61.60 ± 1.18	64.22 ± 1.10	$\mathbf{64.28 \pm 1.44}$	58.73 ± 1.65	63.35 ± 1.23	60.81 ± 1.02
	10% both	66.53 ± 1.04	$\mathbf{70.02 \pm 1.34}$	68.05 ± 0.74	61.54 ± 1.39	65.31 ± 0.96	62.83 ± 1.29
	30% both	64.68 ± 1.00	$\mathbf{64.93 \pm 0.90}$	63.97 ± 1.26	54.24 ± 2.28	58.81 ± 1.14	54.48 ± 2.01
	10% feat.	27.55 ± 5.39	30.18 ± 4.65	17.72 ± 3.40	$\mathbf{33.08 \pm 2.72}$	25.97 ± 7.17	30.17 ± 2.20
	30% feat.	17.93 ± 6.77	22.25 ± 8.07	18.13 ± 4.40	$\mathbf{31.56 \pm 3.64}$	19.70 ± 7.00	28.54 ± 4.62

Learning from enemies: The mechanisms built into Diff-GCN and Nlinear-GCN do not allow it to consider information from "joint enemies" in this fashion, so we do not see the same benefits from dealing with a small number of classes. Nevertheless, we find that, when normalizing by edge degree in Diff-GCN, (i.e. for Diff-GCN-Split), a consistent improvement in performance for an increase in between-community degree can be seen. This highlights that the update rule allows it to directly incorporate feature information from "enemies" in a constructive manner. Nlinear-GCN-split, on the other hand, shows either consistent improvement (min-max scaling) or avoids reducing the performance (z-score scaling). Furthermore, our models appear competitive on the real data sets (see Table 2), which feature a low between-community degree and a larger number of target classes. Finally, we find that the proposed update rules may have a stronger bias towards maintaining meaningful feature information: unlike all alternative models, the classification performance of Nlinear-GCN is significantly more robust towards the introduction of additional GCN layers (results not shown).

Learning from signed graph properties: When employing signed models on the citation networks, one of our primary concerns is the interpretation of negative links, as they are artificially created from cross-community citation links. Thus, their meaning does not inherently differ from positive links; both imply citation from one publication to another. To address this, we aim to validate the signed models on a different type of network — social networks — where positive and negative links have different inherent meanings given by trust and distrust relationships among individuals.

The results on the Bitcoin data shown in Table 3 indicate that Pos-GCN no longer maintains its status as the top-performing model, as observed in the citation networks. This shift is primarily attributed to

positive edges no longer corresponding exclusively to within edges, thereby posing a more challenging classification task. Furthermore, the additional insights from negative links contribute substantial value. Only signed models that have a separate treatment for negative links (R-GCN, Lap-GCN, Diff-GCN, and Nlinear-GCN) exhibit significantly improved performance compared to simpler models that either disregard negative information or treat negative links as positive (GCN, Pos-GCN, and Nlinear-GCN).

Table 3: Summary of results (in percentage) of the different models on the Bitcoin-OTC networks. The best-performing model for each metric is printed in bold

Methods	F1-Binary	F1-Macro	Accuracy
GCN	49.51 ± 2.74	88.83 ± 1.22	71.62 ± 1.70
Pos-GCN	28.62 ± 2.54	91.91 ± 0.32	62.17 ± 1.32
Norm-GCN	30.19 ± 2.02	91.68 ± 0.36	62.88 ± 1.06
R-GCN	66.69 ± 3.23	92.77 ± 1.21	81.32 ± 1.97
Lap-GCN	67.44 ± 1.74	93.55 ± 0.30	81.93 ± 0.94
Diff-GCN	$\mathbf{72.05 \pm 1.91}$	$\mathbf{95.83 \pm 0.26}$	$\mathbf{84.90 \pm 1.02}$
Nlinear-GC	62.79 ± 2.90	93.76 ± 0.74	79.69 ± 1.64

Our model, Diff-GCN, notably outperforms competing models in the context of the Bitcoin data, possibly due to the distinct network structure featuring a power law distribution of degrees and varied edge weights. For instance, in the trustworthy group, it is shown that nodes with high degrees tend to form strong positive connections, while the edge weight distribution appears random in the risky group (refer to Appendix B for more details).

6 Discussion and Conclusion

Our work highlights that differences in the modelling of negative edges introduce significant assumptions into how models convey "messages" along negative edges. In existing approaches, nodes with the same "enemies" will receive the same update, implying that nodes sharing a significant number of enemies will be classified as belonging to the same group of "friends" even when they are not (e.g. enemies of enemies may still be enemies). In contrast, our approach aims to distinguish information from each individual enemy, based on the enemy's features relatively to the target node. Consequently, even nodes with the same enemies may experience significantly different updates - removing the assumption that shared enemies necessarily contribute to similarity.

Our results confirm a significant impact of feature scaling, but this is shared across all models and suggests sensitivities of the GCN learning process, in the context of unscaled features. Nevertheless, we are conscious that for Nlinear-GCN, in particular, feature scaling will impact the relative scale of positive and negative edge updates, and will consider mechanisms to address this in future work.

We also note that R-GCN offers improved flexibility in controlling information flow from different edge types. The use of separate weight matrices for positive and negative edges, as well as self-loops, is likely to be particularly impactful if there are differences in the reliability of these three information sources, as this can be accommodated during the learning process. This flexibility of R-GCN comes at the cost of increased complexity and potential for overfitting, which has led to the suggestion of regularisation techniques, including basis decomposition and block diagonal decomposition [33]. These regularizations can ensure some consistency in feature weighting whilst preserving R-GCN's ability to upweight the most relevant edge types for a given task. In terms of our own approach, we assume a single shared weight matrix, but the future addition of three individual, learnable weights may help moderate the importance (reliability) of each information source, improving robustness.

Finally, the strong performance of Pos-GCN on the real citation data highlights the limitations of those particular experiments: as negative edges were introduced on the basis of existing (cross-community) citation links, they may not be good candidates for that role, explaining why negative edges add little value. The competitiveness of our proposed models on the Bitcoin Data set highlights the potential promise of our approach on a realistic signed network. In future work, we are keen to conduct a comprehensive study across a much wider range of realistic signed networks with different structures, including networks with more than two classes.

References

[1] Bruna, J., Zaremba, W., Szlam, A. and LeCun, Y. [2013]. Spectral networks and locally connected networks on graphs, *arXiv preprint arXiv:1312.6203* .

[2] Cartwright, D. and Harary, F. [1956]. Structural balance: a generalization of heider's theory., *Psychological review* **63**(5): 277.

[3] Defferrard, M., Bresson, X. and Vandergheynst, P. [2016]. Convolutional neural networks on graphs with fast localized spectral filtering, *Advances in neural information processing systems* **29**.

[4] Derr, T., Ma, Y. and Tang, J. [2018]. Signed graph convolutional networks, *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, pp. 929–934.

[5] Doreian, P. and Mrvar, A. [2019]. Structural balance and signed international relations, *Journal of Social Structure* **16**(1).

[6] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. and Dahl, G. E. [2017]. Neural message passing for quantum chemistry, *International conference on machine learning*, PMLR, pp. 1263–1272.

[7] Glorot, X. and Bengio, Y. [2010]. Understanding the difficulty of training deep feedforward neural networks, *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, pp. 249–256.

[8] Guha, R., Kumar, R., Raghavan, P. and Tomkins, A. [2004]. Propagation of trust and distrust, *Proceedings of the 13th international conference on World Wide Web*, pp. 403–412.

[9] Hamilton, W. L. [2020]. Graph representation learning, *Synthesis Lectures on Artifical Intelligence and Machine Learning* **14**(3): 1–159.

[10] Hamilton, W. L., Ying, R. and Leskovec, J. [2017]. Representation learning on graphs: Methods and applications, *arXiv preprint arXiv:1709.05584* .

[11] Harary, F., Lim, M.-H. and Wunsch, D. C. [2002]. Signed graphs for portfolio analysis in risk management, *IMA Journal of management mathematics* **13**(3): 201–210.

[12] Heider, F. [1946]. Attitudes and cognitive organization, *The Journal of psychology* **21**(1): 107–112.

[13] Huang, J., Shen, H., Hou, L. and Cheng, X. [2019]. Signed graph attention networks, *International Conference on Artificial Neural Networks*, Springer, pp. 566–577.

[14] Huang, J., Shen, H., Hou, L. and Cheng, X. [2021]. Sdgnn: Learning node representation for signed directed networks, *arXiv preprint arXiv:2101.02390* .

[15] Iacono, G., Ramezani, F., Soranzo, N. and Altafini, C. [2010]. Determining the distance to monotonicity of a biological network: a graph-theoretical approach, *IET Systems Biology* **4**(3): 223–235.

[16] Islam, M. R., Prakash, B. A. and Ramakrishnan, N. [2018]. Signet: Scalable embeddings for signed networks, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, pp. 157–169.

[17] Kim, J., Park, H., Lee, J.-E. and Kang, U. [2018]. Side: representation learning in signed directed networks, *Proceedings of the 2018 World Wide Web Conference*, pp. 509–518.

[18] Kingma, D. P. and Ba, J. [2014]. Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* .

[19] Kipf, T. N. and Welling, M. [2016]. Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* .

[20] Kumar, S., Hamilton, W. L., Leskovec, J. and Jurafsky, D. [2018]. Community interaction and conflict on the web, *Proceedings of the 2018 world wide web conference*, pp. 933–943.

[21] Kumar, S., Spezzano, F., Subrahmanian, V. and Faloutsos, C. [2016]. Edge weight prediction in weighted signed networks, *2016 IEEE 16th International Conference on Data Mining (ICDM)*, IEEE, pp. 221–230.

[22] Kunegis, J., Lommatzsch, A. and Bauckhage, C. [2009]. The slashdot zoo: mining a social network with negative edges, *Proceedings of the 18th international conference on World wide web*, pp. 741–750.

[23] Kunegis, J., Schmidt, S., Lommatzsch, A., Lerner, J., De Luca, E. W. and Albayrak, S. [2010]. Spectral analysis of signed graphs for clustering, prediction and visualization, *Proceedings of the 2010 SIAM international conference on data mining*, SIAM, pp. 559–570.

[24] Leskovec, J., Huttenlocher, D. and Kleinberg, J. [2010a]. Predicting positive and negative links in online social networks, *Proceedings of the 19th international conference on World wide web*, pp. 641–650.

[25] Leskovec, J., Huttenlocher, D. and Kleinberg, J. [2010b]. Signed networks in social media, *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1361–1370.

[26] Li, X., Saúde, J., Reddy, P. P. and Veloso, M. M. [2019]. Classifying and understanding financial data using graph neural network.

[27] Li, X. et al. [2020]. Explain graph neural networks to understand weighted graph features in node classification, *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, pp. 57–76.

[28] Li, Y., Tian, Y., Zhang, J. and Chang, Y. [2020]. Learning signed network embedding via graph attention, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 4772–4779.

[29] Liu, H., Zhang, Z., Cui, P., Zhang, Y., Cui, Q., Liu, J. and Zhu, W. [2021]. Signed graph neural network with latent groups, *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1066–1075.

[30] Loe, D., Chang, S.-l. and Chau, J. [2021]. Stock market movement prediction using graph convolutional networks.

[31] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J. and Bronstein, M. M. [2017]. Geometric deep learning on graphs and manifolds using mixture model cnns, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5115–5124.

[32] Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. and Vandergheynst, P. [2018]. Graph signal processing: Overview, challenges, and applications, *Proceedings of the IEEE* **106**(5): 808–828.

[33] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I. and Welling, M. [2018]. Modeling relational data with graph convolutional networks, *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, Springer, pp. 593–607.

[34] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A. and Vandergheynst, P. [2013]. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, *IEEE signal processing magazine* **30**(3): 83–98.

[35] Tang, J., Chang, Y., Aggarwal, C. and Liu, H. [2016]. A survey of signed network mining in social media, *ACM Computing Surveys (CSUR)* **49**(3): 1–37.

[36] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y. [2017]. Graph attention networks, *arXiv preprint arXiv:1710.10903* .

[37] Vos, W. B. W. [2017]. End-to-end learning of latent edge weights for graph convolutional networks, *Master's thesis, University of Amsterdam* .

[38] Wang, S., Tang, J., Aggarwal, C., Chang, Y. and Liu, H. [2017]. Signed network embedding in social media, *Proceedings of the 2017 SIAM international conference on data mining*, SIAM, pp. 327–335.

[39] Yang, Z., Cohen, W. and Salakhudinov, R. [2016]. Revisiting semi-supervised learning with graph embeddings, *International conference on machine learning*, PMLR, pp. 40–48.

A Mathematical formula

In this section, we present the mathematical formula of eight signed models examined in the paper. They include: the GCN ignoring edge signs (GCN), the GCN applied to the positive edges only (Pos-GCN), Lap-GCN [23], R-GCN [33], and our two models namely Diff-GCN and Nlinear-GCN with two additional variants having a different way of normalization called Diff-GCN-Split and Nlinear-GCN-Split. The models Diff-GCN-Split and Nlinear-GCN-Split have been specifically designed to normalize the relative impact of positive and negative edges as a function of their frequency, as suggested within R-GCN.

The hidden representation vector of node i of $(l+1)$-th hidden layer for each model is shown below.
GCN formula

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}'}_{ii}\tilde{\mathbf{D}'}_{jj}}}\mathbf{H}_j^{(l)}\mathbf{W}^{(l)}\right) \tag{13}$$

where $\mathcal{N}_i$ denotes the set of neighbour indices of node i. The signed diagonal degree matrix $\tilde{\mathbf{D}'}$ given by $\tilde{\mathbf{D}'}_{ii} = \sum_j |\tilde{\mathbf{A}}_{ij}|$, and $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. $\sigma(\cdot)$ denotes the non-linear activation function. In our experiments, we use the activation function $ReLU(\cdot) = \max(0\cdot)$. And $\mathbf{W}^{(l)}$ is the trainable weight matrix in the l-th layer.

Pos-GCN formula

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i^{(+)}\cup\{i\}} \frac{\tilde{\mathbf{A}}_{ij}^{(+)}}{\sqrt{\tilde{\mathbf{D}'}_{ii}^{(+)}\tilde{\mathbf{D}'}_{jj}^{(+)}}}\mathbf{H}_j^{(l)}\mathbf{W}^{(l)}\right) \tag{14}$$

where $\mathcal{N}_i^{(+)}$ denotes the set of neighbour indices which have positive links with node i. The adjacency matrix $\tilde{\mathbf{A}}_{ij}^{(+)}$ and its corresponding degree matrix $\tilde{\mathbf{D}'}_{ii}^{(+)}$ are calculated based on only positive links.

Lap-GCN formula

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}} sign(\tilde{\mathbf{A}}_{ij})\mathbf{H}_j^{(l)}\mathbf{W}^{(l)}\right) \tag{15}$$

where $sign(\tilde{\mathbf{A}}_{ij})$ denotes the sign of the link between nodes i and j.

R-GCN formula

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}}\mathbf{W}_r^{(l)}\mathbf{H}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{H}_i^{(l)}\right) \tag{16}$$

where $\mathcal{N}_i^r$ denotes the set of neighbour indices of node i regarding relation $r \in \mathcal{R}$. $c_{i,r}$ is a normalization constant which can be learned or chosen in advance. In our experiment, we choose $c_{i,r} = |\mathcal{N}_i^r|$ as suggested in [33].

Diff-GCN formula

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}}\left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij})\frac{\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}}{||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}||}||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}||\right)\mathbf{W}^{(l)}\right) \tag{17}$$

Equation (17 is the expanded form of Diff-GCN which aims to show clearly the unit vector and scale that define the direction and the magnitude of the updated information. It can be further reduced as below:

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}}\left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij})(\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)})\right)\mathbf{W}^{(l)}\right) \tag{18}$$

Diff-GCN-Split formula

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\left(\sum_{j\in\mathcal{N}_i^{(+)}\cup\{i\}} \frac{1}{|\mathcal{N}_i^{(+)}+1|}\left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij})(\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)})\right) + \sum_{j\in\mathcal{N}_i^{(-)}} \frac{1}{|\mathcal{N}_i^{(-)}|}\left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij})(\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)})\right)\right)\mathbf{W}^{(l)}\right) \tag{19}$$

where $\mathcal{N}_i^{(-)}$ denotes the set of neighbour indices which have negative links with node i. Notice that in the equation (19), we have two different normalization terms for positive and negative edges. When we insert $sign(\tilde{A}_{ij}) = 1$ for positive links, and $sign(\tilde{A}_{ij}) = -1$ for negative links, we can further simplify the formula as follows:

$$\mathbf{H}_i^{(l+1)} = \sigma\left(\left(\sum_{j\in\mathcal{N}_i^{(+)}\cup\{i\}} \frac{1}{|\mathcal{N}_i^{(+)}+1|}\mathbf{H}_j^{(l)} + \sum_{j\in\mathcal{N}_i^{(-)}} \frac{1}{|\mathcal{N}_i^{(-)}|}(2\mathbf{H}_i^{(l)} - \mathbf{H}_j^{(l)})\right)\mathbf{W}^{(l)}\right) \tag{20}$$

Nlinear-GCN formula

$$\mathbf{H}_{ij}^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i^{(*)}} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}}\left(\mathbf{H}_i^{(l)} + sign(\tilde{\mathbf{A}}_{ij})\frac{(\mathbf{H}_j^{(l)}-\mathbf{H}_i^{(l)})}{||\mathbf{H}_j^{(l)}-\mathbf{H}_i^{(l)}||}\left(||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| + \frac{1-sign(\tilde{\mathbf{A}}_{ij})}{2}\right)^{sign(\tilde{\mathbf{A}}_{ij})}\right)\mathbf{W}^{(l)}\right) \tag{21}$$

where $\mathcal{N}_i^{(*)}$ is the set of neighbouring nodes j satisfying $||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| \neq 0$. In case that $||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| = 0$ and $\mathcal{N}_i^{(**)}$ denotes the set of neighbouring nodes j satisfying $||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| = 0$, the message passing of Nonlinear-GCN is:

$$\mathbf{H}_{ij}^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i^{(**)}\cup\{i\}} \frac{|\tilde{\mathbf{A}}_{ij}|}{\sqrt{\tilde{\mathbf{D}}'_{ii}\tilde{\mathbf{D}}'_{jj}}}\mathbf{H}_i^{(l)}\mathbf{W}^{(l)}\right) \tag{22}$$

Note that $\mathcal{N}_i = \mathcal{N}_i^{(*)} \cup \mathcal{N}_i^{(**)}$. In case two nodes i and j have identical feature values, Nlinear-GCN keeps the node's own features as the update in equation (22). In this case, the direction of updated information cannot be identified; thus, keeping the original node features at least will not cause any harm to the update. However, it does not mean that the negative link, in this case, is not utilized; instead, it will wait until the next update to be used. After aggregating information from all neighbouring nodes, nodes i and j likely do not have the same feature values as before, except that they have exactly the same set of positive and negative neighbouring nodes, which may not usually happen in real-world networks. The negative link between node i and j can now be utilized to update their information in the next layer using equation (21).

Nlinear-GCN-Split formula

$$\mathrm{H}_i^{(l+1)} = \sigma\left(\left(\sum_{j\in\mathcal{N}_i^{(+)}\cup\{i\}} \frac{1}{|\mathcal{N}_i^{(+)}+1|}\mathbf{H}_j^{(l)} + \sum_{j\in\mathcal{N}_i^{(-)(*)}} \frac{1}{|\mathcal{N}_i^{(-)}|}\left(\mathbf{H}_i^{(l)} - \frac{(\mathbf{H}_j^{(l)}-\mathbf{H}_i^{(l)})}{||\mathbf{H}_j^{(l)}-\mathbf{H}_i^{(l)}||}\frac{1}{(||\mathbf{H}_j^{(l)}-\mathbf{H}_i^{(l)}||+1)}\right) + \sum_{j\in\mathcal{N}_i^{(-)(**)}} \frac{1}{|\mathcal{N}_i^{(-)}|}\mathbf{H}_i^{(l)}\right)\mathbf{W}^{(l)}\right) \tag{23}$$

where $\mathcal{N}_i^{(-)(*)}$ and $\mathcal{N}_i^{(-)(**)}$ are both the sets of neighbour indices which have negative links with node i and satisfy $||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| \neq 0$ and $||\mathbf{H}_j^{(l)} - \mathbf{H}_i^{(l)}|| = 0$, respectively.

B Experimental setup

In this section, we present a full description of the experimental setup for synthetic and real networks.

B.1 Synthetic networks

An undirected graph consists of 300 nodes assigned to three equal communities, namely Group 1, Group 2 and Group 3.

Node features generation Every node in the three groups has been attached with two features whose values are generated by normal distributions. Feature 1 ranges from (0,120] while Feature 2 varies from (0, 100]. We create two scenarios with different levels of overlap in the feature values of three classes: the first scenario is the small overlap in feature values of three classes, and the second scenario is the large overlap in feature values of three classes. Parameters defining the generation of

Table 4: Synthetic networks

Scenario	Community 1		Community 2		Community 3	
	Feature 1	Feature 2	Feature 1	Feature 2	Feature 1	Feature 2
Small feature overlap	25 ± 20	15 ± 20	60 ± 20	45 ± 20	95 ± 20	75 ± 20
Large feature overlap	60 ± 20	15 ± 20	80 ± 20	35 ± 20	100 ± 20	45 ± 20

node features for two different scenarios are specified in Table 4. It shows mean $\pm$ standard deviation of each feature for every group.

Edges generation To generate three communities with the same number of within edges, we randomly choose the same amount of edges from all possible within-community links in Group 1, Group 2 and Group 3. Similarly, to make sure they have the same number of between edges, we randomly choose the same amount of edges from all possible between-community links connecting Group 1 and Group 2, Group 1 and Group 3, Group 2 and Group 3. Each edge is assigned a corresponding weight. Here, the weights of within-community links (or positive links) are equal to 1, and the weights of between-community links (or negative links) are set to -1.

Edges' sign generation The signs of edges are created based on a simple intuition that nodes belonging to the same class tend to be near each other and share a certain degree of similarity. Thus, they should be connected by positive links. On the contrary, nodes in different classes are likely dissimilar; therefore, they should be linked by negative connections.

Normalization In order to test for translation invariance and scale invariance in selected models, we perform two feature scaling techniques, including Min-Max scaling and z-score normalization. We apply Min-Max scaling to scale features in a range from 0 to 1 using the following formula:

$$x' = (x - x_{min})/(x_{max} - x_{min}) \tag{24}$$

The formula for calculating the z-score of a point, x, is as follows:

$$x' = (x - \mu)/\sigma \tag{25}$$

where μ is the mean and σ is the standard deviation. Notice that instead of normalizing the entire dataset, we apply feature normalization separately to the training data. The validation and test data are subsequently normalized using the mean and variance computed from the training data. This approach ensures that no future information is introduced during training, allowing us to evaluate the model's ability to generalize effectively to new, unseen data points.

B.2 Citation networks

The Cora and Citeseer citation datasets can be modified to increase the difficulty of the classification problem by introducing edge noise and feature noise.

Edge noise In real networks, it is not necessarily that within (or between) edges are always assigned as positive (negative) links. To evaluate the robustness of models in such scenarios, we introduce edge noise by randomly selecting a specific proportion of previously unused positive or negative links and assigning them opposite weights. For instance, with 10% negative noise, we combine the original negative links with additional previously unused positive links, which are now considered as negative links and constitute 10% of the original negative links. This approach allows us to assess the model's performance under different levels of edge noise in the network.

Feature noise In the context of feature noise, we introduce randomness by selecting a specific number of features and assigning them random values. For instance, with 10% feature noise in the Cora data, we randomly choose 10% of the features for each node and assign them values of 0 or 1, considering that the features in the Cora dataset are binary. This approach allows us to evaluate the model's robustness to variations in feature values and assess its performance under different levels of feature noise.

B.3 Bitcoin-OTC

Bitcoin-OTC [21] is collected on a monthly basis over the period between 2010 and 2016 to show how Bitcoin users rate their trust in other users. The Bitcoin-OTC network is a directed graph where

source nodes (raters) give ratings to target nodes (rates) on a scale of -10 to +10 (excluding 0) in steps of 1. According to OTC's guidelines, a rating of -10 is considered as total distrust, whereas a rating of +10 means total trust.

As the Bitcoin-OTC data does not contain node label information, we first create labels for every user according to their received ratings. The users whose total number of positive ratings received was higher than those of negative ratings received are classified as trustworthy users (label 0); otherwise, the users are classified as risky (label 1). After that, we convert the original directed graph to an undirected graph by not considering the direction of the edges between every pair of vertices; in other words, the edges become bidirectional. If there are multiple edges between a pair of nodes during a specific period, the edge weight is taken as the mean value of the edge weights. As no node feature is available, we only use onehot encoding reflecting the identity of each node in the graph as the node features. It can be said these 5881-dimensional features do not carry any specific information about nodes. Therefore, in this case, the only available information is edge information, including edge signs and weights.

The Bitcoin network exhibits a distinct graph structure shown in Figure 3. The first column represents the node degree distribution of the unweighted graph. It only counts the number of edges without accounting for their weights when computing node degree. In the second column, node degrees are calculated based on the values of positive ratings. In the third column, node degrees are computed based on the absolute values of negative ratings.

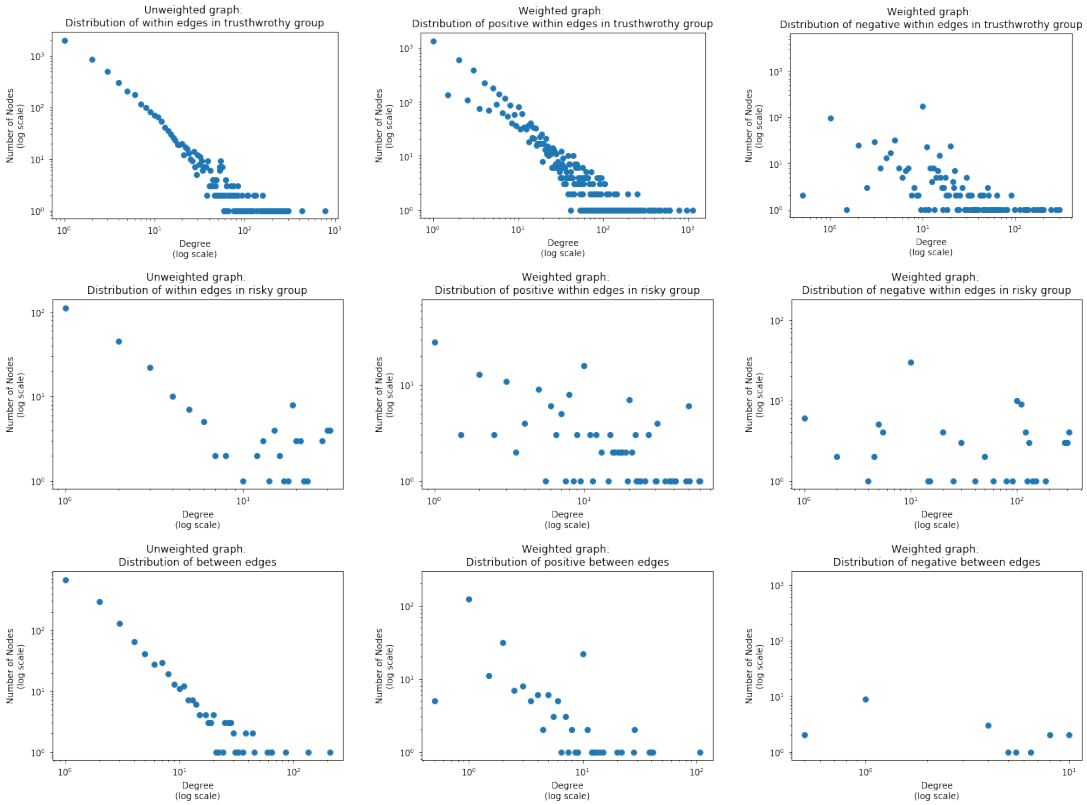

Figure 3: Degree distribution of the Bitcoin network

B.4 Evaluation measure

We employ accuracy as the evaluation metric for both synthetic and real-world data.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Number of total predictions}} \tag{26}$$

Considering the presence of class imbalance in real-world data, we also utilize the F1-macro score as an additional metric. The F1-macro score calculates the unweighted mean of the F1-scores computed

for each class. F1-macro metrics can be expressed in terms of True Positive (TP), True Negative (TN) , False Positive (FP) and False Negative (FN) as follows:

$$P_t = \frac{TP_t}{TP_t + FP_t} \tag{27}$$

$$R_t = \frac{TP_t}{TP_t + FN_t} \tag{28}$$

$$F1 - macro = \frac{1}{T} \sum_{t=1}^{T} \frac{2P_t R_t}{P_t + R_t} \tag{29}$$

where t is the class index, and T is the number of classes in the data set.

For the Bitcoin data, where we encounter binary classification with imbalanced data classes, we incorporate an additional metric called the F1-binary score to evaluate the models' ability to predict the minority class, i.e. to identify risky users.

$$F1 - binary = \frac{2P_m R_m}{P_m + R_m} \tag{30}$$

where m is the minority or risky class.

C Additional results

C.1 Results on synthetic networks

The following figures display the results for both scenarios in synthetic data, regarding different feature scaling techniques. A consistent pattern observed in both scenarios is the improved performance of all models due to feature scaling. Notably, Nlinear-GCN is particularly sensitive to feature scaling, as it affects the relative scale of positive and negative edge updates.

C.2 Results on real citation networks

The two following tables show the mean accuracy and F1-macro score for all selected models in real-world graphs. Our models demonstrate competitive performance on real datasets that have a low between-community degree and a higher number of target classes. However, the strong performance of Pos-GCN raises concerns about the way of introducing negative links (by using cross-community citation) which could be the reason why negative edges provide limited value.

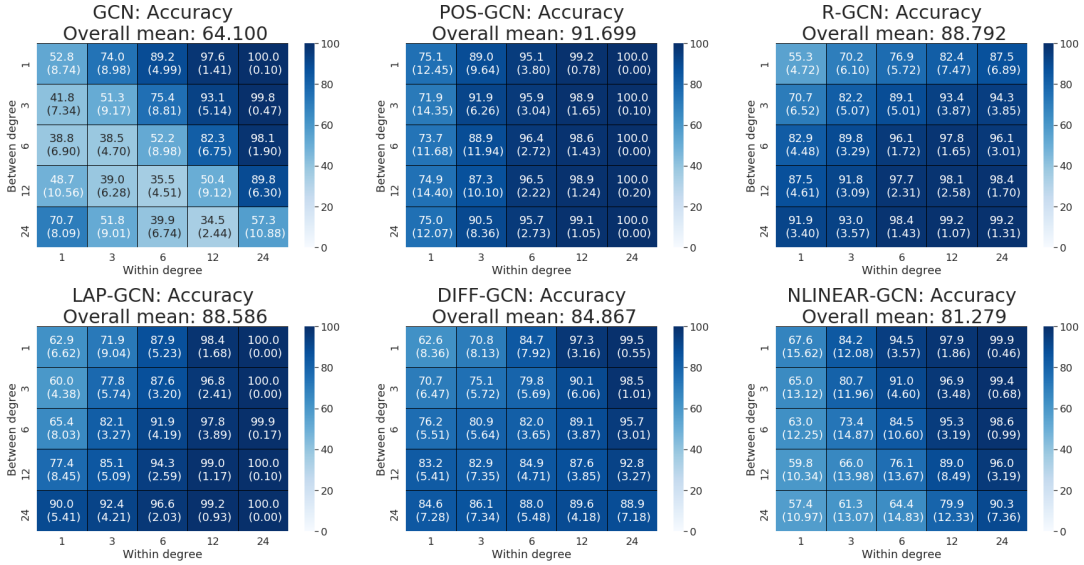

Figure 4: SMALL OVERLAP: Results using unscaled features

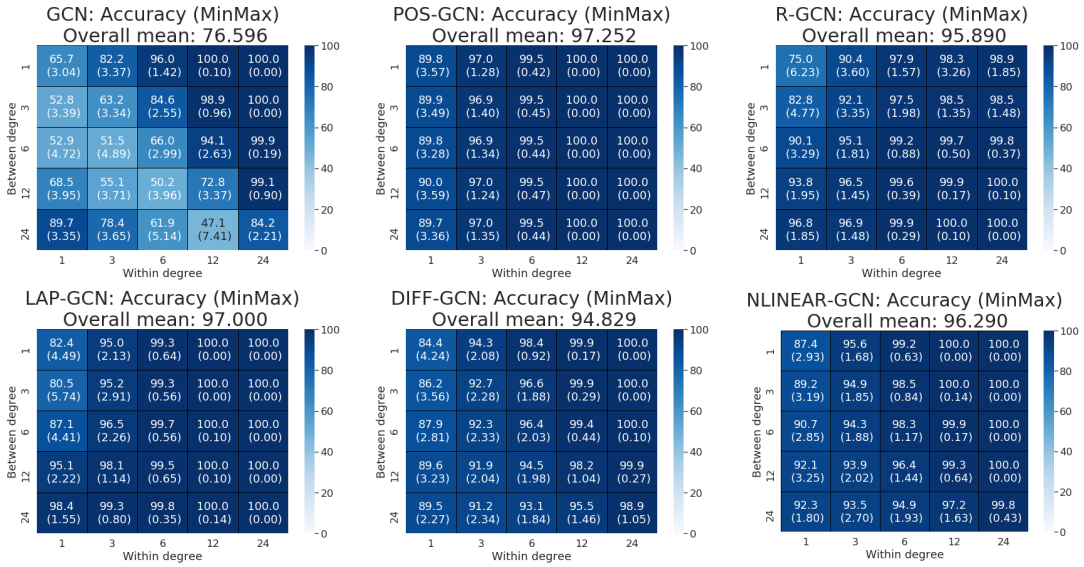

Figure 5: SMALL OVERLAP: Results using Min-Max normalized features

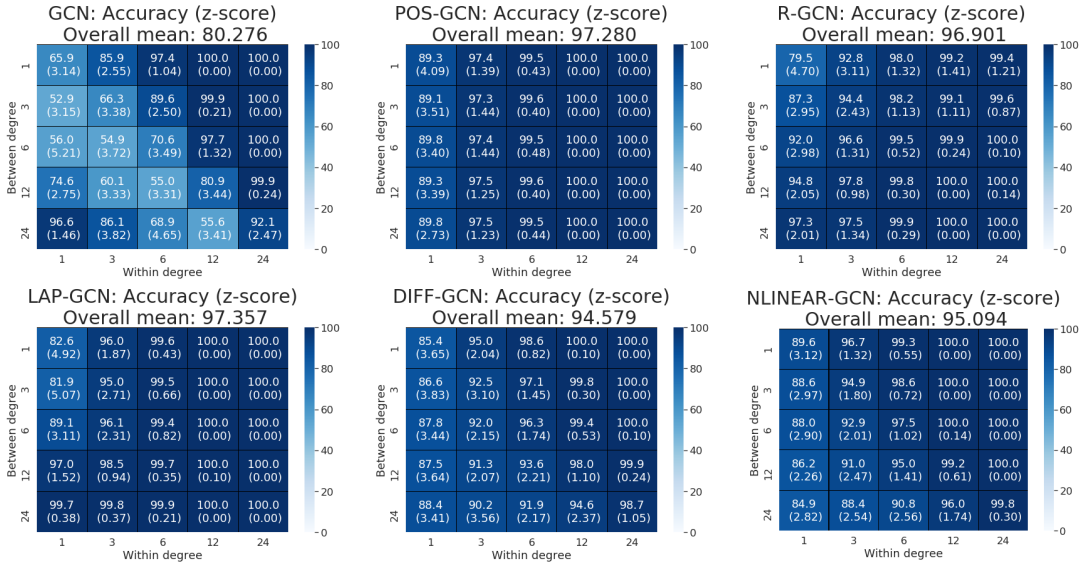

Figure 6: SMALL OVERLAP: Results using z-score normalized features

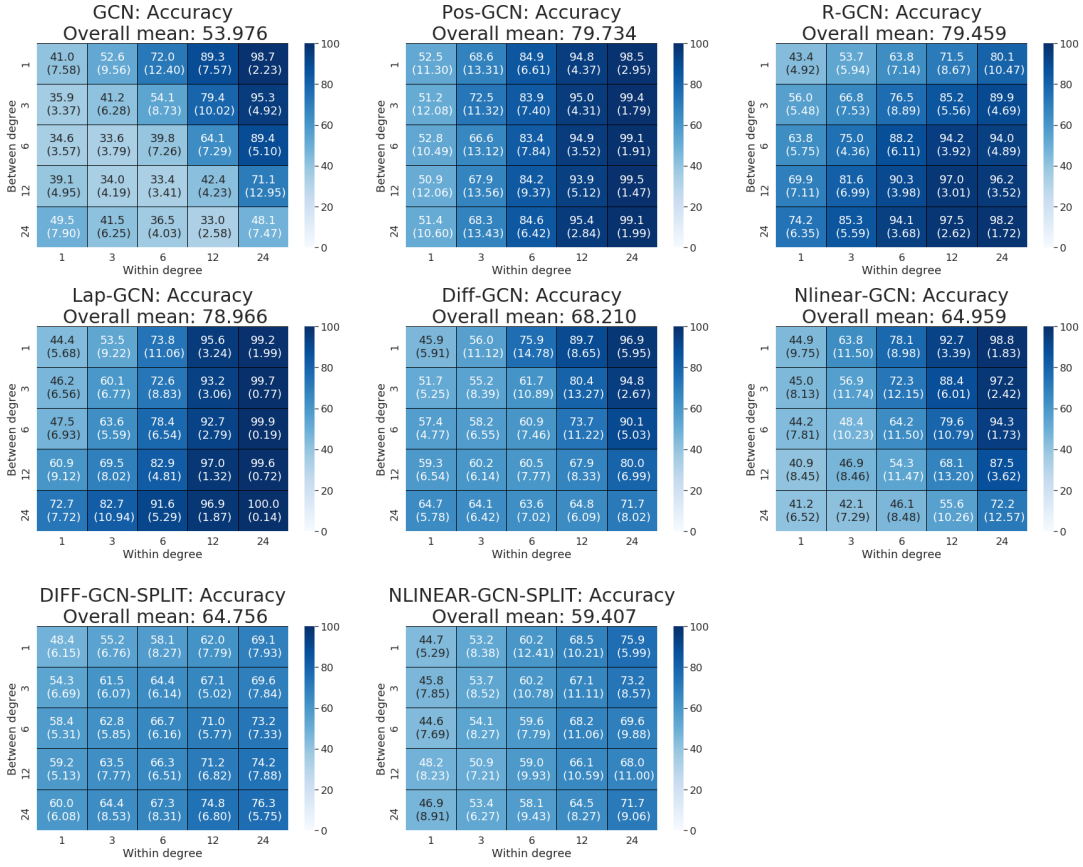

Figure 7: LARGE OVERLAP: Results using unscaled features

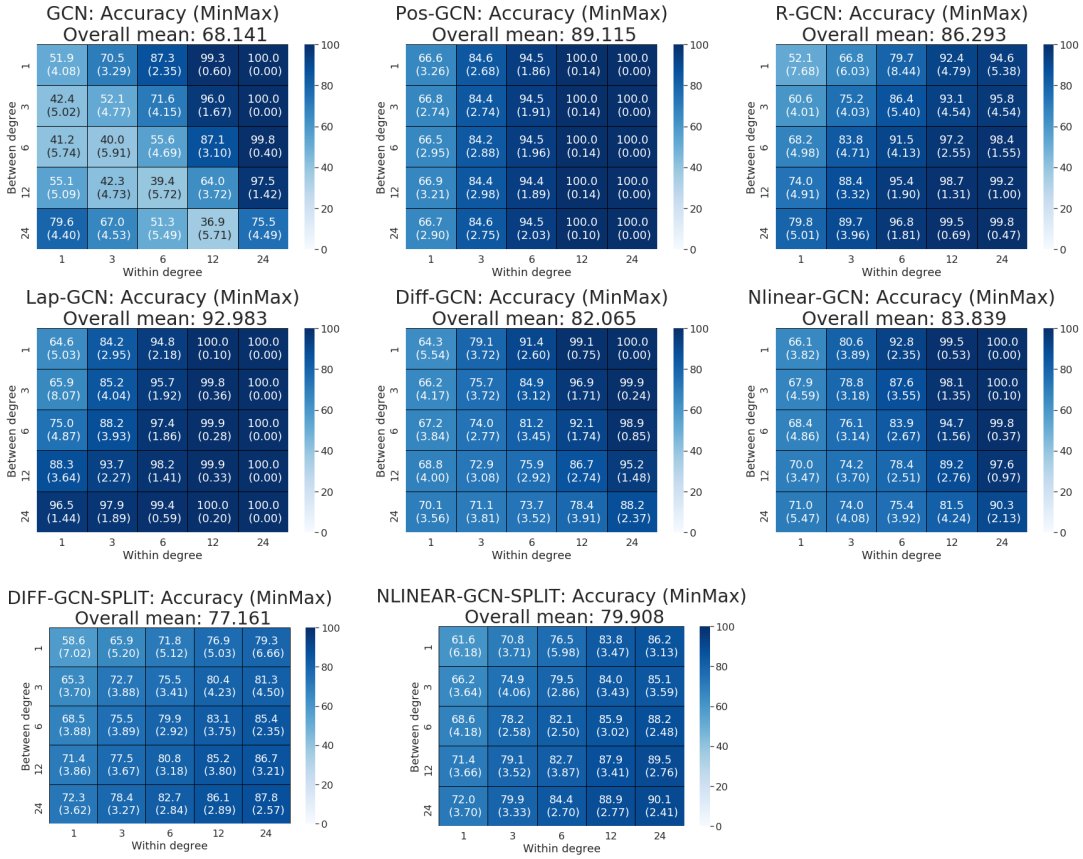

Figure 8: LARGE OVERLAP: Results using Min-Max normalized features

Table 5: Accuracy of the different models on the real-world citation networks for various noise settings. Averages over 30 weight initialization. The best-performing model is highlighted.

Data	Settings	GCN	Pos-GCN	R-GCN	Lap-GCN	Diff-GCN	Nlinear-GCN	Diff-GCN-split	Nlinear-GCN-split
Cora	No noise	80.72 ± 0.55	**87.19 ± 0.31**	85.68 ± 0.29	83.27 ± 0.77	84.36 ± 0.47	84.09 ± 0.34	80.29 ± 1.03	80.76 ± 0.94
	10% neg.	81.68 ± 0.53	**87.07 ± 0.31**	84.98 ± 0.69	81.79 ± 0.83	82.67 ± 0.52	82.36 ± 0.42	76.21 ± 1.86	74.18 ± 2.37
	30% neg.	82.74 ± 0.45	**87.19 ± 0.32**	84.80 ± 0.49	78.27 ± 1.21	80.03 ± 0.75	79.04 ± 0.96	67.80 ± 2.24	66.11 ± 1.43
	10% pos.	78.02 ± 0.63	**83.92 ± 0.57**	81.79 ± 0.74	80.40 ± 0.95	81.43 ± 0.85	80.91 ± 0.61	76.14 ± 1.52	75.68 ± 1.79
	30% pos.	73.71 ± 0.75	**78.00 ± 1.06**	77.12 ± 0.95	75.35 ± 0.78	76.17 ± 0.99	75.77 ± 0.83	68.08 ± 2.36	66.48 ± 1.70
	10% both	78.88 ± 0.58	**83.52 ± 0.87**	81.69 ± 1.13	78.76 ± 0.57	79.93 ± 0.77	79.16 ± 0.70	70.67 ± 2.24	68.35 ± 2.08
	30% both	76.60 ± 0.60	**78.01 ± 0.76**	76.10 ± 1.56	70.41 ± 1.14	72.27 ± 0.65	71.17 ± 0.95	47.50 ± 8.25	55.58 ± 1.80
	10% feat.	52.78 ± 2.38	58.93 ± 2.63	33.12 ± 7.11	**59.60 ± 1.93**	52.90 ± 2.34	57.90 ± 1.85	34.46 ± 7.50	57.05 ± 2.33
	30% feat.	47.81 ± 5.54	54.76 ± 6.20	25.99 ± 6.45	58.23 ± 2.33	51.96 ± 2.42	55.42 ± 3.22	26.69 ± 8.37	**56.57 ± 2.72**
Citeseer	No noise	71.00 ± 0.54	**76.70 ± 0.36**	72.60 ± 0.54	69.30 ± 0.59	73.94 ± 0.32	72.40 ± 0.37	68.37 ± 1.12	56.59 ± 4.79
	10% neg.	71.83 ± 0.66	**76.91 ± 0.46**	72.17 ± 0.49	67.37 ± 1.33	72.08 ± 0.68	69.36 ± 1.02	61.47 ± 4.97	50.38 ± 4.52
	30% neg.	73.19 ± 0.85	**76.70 ± 0.33**	72.26 ± 1.10	63.35 ± 1.57	68.70 ± 0.97	64.59 ± 1.94	55.17 ± 2.74	46.23 ± 3.66
	10% pos.	67.93 ± 0.83	**73.20 ± 0.90**	69.85 ± 0.74	67.35 ± 0.75	71.79 ± 0.58	69.85 ± 0.56	64.22 ± 1.65	53.75 ± 4.39
	30% pos.	64.21 ± 1.28	67.40 ± 1.31	66.00 ± 1.58	62.72 ± 1.43	**67.65 ± 1.26**	65.03 ± 0.98	54.82 ± 5.61	47.03 ± 5.40
	10% both	69.47 ± 1.15	73.26 ± 1.15	**69.83 ± 0.87**	65.10 ± 1.47	69.98 ± 0.98	67.25 ± 0.98	57.69 ± 5.61	47.03 ± 5.40
	30% both	67.41 ± 1.12	**67.98 ± 1.01**	65.68 ± 1.54	57.25 ± 2.44	62.78 ± 1.38	57.85 ± 2.05	43.38 ± 9.32	34.58 ± 8.93
	10% feat.	30.91 ± 3.84	32.90 ± 3.59	22.14 ± 2.10	**34.70 ± 2.67**	29.13 ± 5.89	32.78 ± 1.94	22.31 ± 4.49	29.94 ± 4.61
	30% feat.	24.45 ± 4.82	27.12 ± 6.23	21.86 ± 3.57	**33.08 ± 3.42**	24.83 ± 5.12	31.69 ± 3.64	20.85 ± 4.74	28.68 ± 5.43

Table 6: F1-macro score of the different models on the real-world citation networks for various noise settings. Averages over 30 weight initialization. The best-performing model is highlighted.

Data	Settings	GCN	Pos-GCN	RGCN	Lap-GCN	Diff-GCN	Nlinear-GCN	Diff-GCN-split	Nlinear-GCN-split
Cora	No noise	79.70 ± 0.52	**86.52 ± 0.42**	84.57 ± 0.79	82.29 ± 0.79	83.07 ± 0.47	82.90 ± 0.39	79.59 ± 0.88	80.03 ± 1.11
	10% neg.	80.48 ± 0.50	**86.49 ± 0.32**	83.86 ± 0.80	80.67 ± 0.96	81.45 ± 0.60	81.17 ± 0.49	75.84 ± 1.60	74.12 ± 1.97
	30% neg.	81.56 ± 0.50	**86.59 ± 0.27**	83.76 ± 0.65	77.25 ± 1.13	78.92 ± 0.56	78.08 ± 0.98	67.74 ± 2.01	66.86 ± 1.47
	10% pos.	77.22 ± 0.74	**83.26 ± 0.65**	80.62 ± 0.81	79.22 ± 0.79	80.05 ± 0.87	79.73 ± 0.62	75.44 ± 1.48	75.11 ± 1.79
	30% pos.	72.80 ± 0.74	**77.15 ± 1.09**	75.92 ± 1.16	74.23 ± 0.86	74.83 ± 1.04	74.38 ± 0.97	67.34 ± 2.19	66.25 ± 1.28
	10% both	77.75 ± 0.62	**82.81 ± 1.07**	80.84 ± 1.33	77.68 ± 0.59	78.66 ± 0.83	78.06 ± 0.79	70.63 ± 1.62	68.65 ± 1.90
	30% both	75.53 ± 0.73	**77.21 ± 0.79**	75.14 ± 1.43	69.53 ± 1.27	71.22 ± 0.83	70.36 ± 1.07	44.89 ± 14.64	56.15 ± 1.41
	10% feat.	30.60 ± 8.80	36.59 ± 7.46	20.80 ± 6.80	38.60 ± 1.22	30.12 ± 6.81	36.94 ± 2.86	13.27 ± 4.41	**40.36 ± 5.32**
	30% feat.	16.36 ± 11.86	19.94 ± 13.67	13.33 ± 5.44	**44.44 ± 4.58**	39.80 ± 5.50	39.44 ± 9.49	19.38 ± 8.42	40.35 ± 11.17
Citeseer	No noise	68.21 ± 0.55	**73.57 ± 0.32**	71.14 ± 0.48	65.40 ± 0.49	68.47 ± 0.42	67.16 ± 0.36	63.39 ± 1.25	53.76 ± 4.41
	10% neg.	68.88 ± 0.67	**73.66 ± 0.48**	70.74 ± 0.45	63.72 ± 1.28	67.17 ± 0.59	64.77 ± 0.87	57.67 ± 4.36	48.52 ± 4.10
	30% neg.	70.25 ± 0.79	**73.52 ± 0.31**	70.78 ± 0.99	60.33 ± 1.27	64.19 ± 1.02	60.64 ± 1.58	52.26 ± 2.60	44.55 ± 3.32
	10% pos.	65.17 ± 0.79	**69.81 ± 1.04**	68.21 ± 0.67	63.40 ± 0.95	66.91 ± 0.69	64.87 ± 0.66	59.48 ± 1.62	51.37 ± 3.40
	30% pos.	61.60 ± 1.18	64.22 ± 1.10	**64.28 ± 1.44**	58.73 ± 1.65	63.35 ± 1.23	60.81 ± 1.02	51.38 ± 2.31	43.58 ± 3.58
	10% both	66.53 ± 1.04	**70.02 ± 1.34**	68.05 ± 0.74	61.54 ± 1.39	65.31 ± 0.96	62.83 ± 1.29	53.67 ± 5.27	45.06 ± 4.73
	30% both	64.68 ± 1.00	**64.93 ± 0.90**	63.97 ± 1.26	54.24 ± 2.28	58.81 ± 1.14	54.48 ± 2.01	39.32 ± 12.31	32.79 ± 8.78
	10% feat.	27.55 ± 5.39	30.18 ± 4.65	17.72 ± 3.40	**33.08 ± 2.72**	25.97 ± 7.17	30.17 ± 2.20	18.74 ± 3.97	26.20 ± 6.20
	30% feat.	17.93 ± 6.77	22.25 ± 8.07	18.13 ± 4.40	**31.56 ± 3.64**	19.70 ± 7.00	28.54 ± 4.62	16.61 ± 4.25	24.22 ± 7.44