# SDGNN: Learning Node Representation for Signed Directed Networks

**Junjie Huang, Huawei Shen*, Liang Hou, Xueqi Cheng**
CAS Key Laboratory of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
{huangjunjie17s, shenhuawei, houliang17z, cxq}@ict.ac.cn

## Abstract

Network embedding is aimed at mapping nodes in a network into low-dimensional vector representations. Graph Neural Networks (GNNs) have received widespread attention and lead to state-of-the-art performance in learning node representations. However, most GNNs only work in unsigned networks, where only positive links exist. It is not trivial to transfer these models to signed directed networks, which are widely observed in the real world yet less studied. In this paper, we first review two fundamental sociological theories (i.e., status theory and balance theory) and conduct empirical studies on real-world datasets to analyze the social mechanism in signed directed networks. Guided by related sociological theories, we propose a novel Signed Directed Graph Neural Networks model named SDGNN to learn node embeddings for signed directed networks. The proposed model simultaneously reconstructs link signs, link directions, and signed directed triangles. We validate our model's effectiveness on five real-world datasets, which are commonly used as the benchmark for signed network embeddings. Experiments demonstrate the proposed model outperforms existing models, including feature-based methods, network embedding methods, and several GNN methods.

## Introduction

With the growing popularity of online social media, many interactions between people are generated and recorded on the web. Modeling and understanding these interactions is a useful perspective on a range of social computing studies. Most of these interactions are positive relationships, such as friend, trust, like, support, and approval. Meanwhile, the conflicts on the web (Kumar et al. 2018) are everywhere, forming negative links that indicate hostility, distrust, hate, disagree, and disapproval. Since the interactions are generated by people and reflect people's attitude and psychology, related researches on social psychology (e.g., status theory and balance theory) shed light on the social mechanism of signed networks. These two theories are widely used in signed network analysis (Leskovec, Huttenlocher, and Kleinberg 2010a). In signed network analysis, link sign prediction is to predict the sign of a given link. Leskovec, Huttenlocher, and Kleinberg carefully extract features based on social theories and achieve good performances. Link sign prediction is the main downstream machine learning task for evaluating signed network embedding. Signed network embedding learning, which aims to map nodes in signed network to low-dimensional vector representations, also relies on these sociological theories (Wang et al. 2017; Kim et al. 2018; Chen et al. 2018a; Islam, Prakash, and Ramakrishnan 2018; Chen et al. 2018b). For example, Chen et al. mathematically model "bridge" edges based on balance and status theory and achieve state-of-the-art performances.

Recently, Graph Neural Networks (GNNs) have been receiving more and more attention. Modern GNNs mostly follow Message Passing Neural Networks (MPNNs) (Gilmer et al. 2017) manner, which consist of message aggregators and update functions. The common aggregators include mean aggregator (Hamilton, Ying, and Leskovec 2017), attention aggregator (Veličković et al. 2018), and max-pooling aggregator (Hamilton, Ying, and Leskovec 2017). GNNs have achieved good results in many machine learning tasks (e.g., semi-supervised node classification, learning low-dimensional representations, and link prediction) (Kipf and Welling 2019, 2016; Wang, Cui, and Zhu 2016). A number of recent researches have pivoted to learn the node embeddings using GNNs, which aim to aggregate information from neighbors for node embeddings (Kipf and Welling 2016; Duran and Niepert 2017; Pan et al. 2018). These GNN-based network embedding methods have revolutionized the field of network embedding and achieved state-of-the-art performances. Specifically, Graph Auto-Encoder (GAE) uses GCNs to process node features jointly with the graph structure to produce a set of hidden representations (Kipf and Welling 2016). It uses a scoring function to reconstruct the adjacency matrix of the graph from hidden representations. GAE is designed for the unsigned networks and can't directly be applied in the signed networks.

For signed network, SGCN generalizes GCN to signed networks and designs a new information aggregator based on balance theory (Derr, Ma, and Tang 2018). SNEA leverages the self-attention mechanism to enhance the performance of signed network embeddings (Li et al. 2020). Although signed GNNs discussed above are proposed to model signed networks with balance theory, they don't take direction (i.e., status theory) into consideration, which is important for signed graph modeling (Cui et al. 2020). In this pa-


*Corresponding Author

per, we try to model signed directed networks with a new Signed Directed Graph Neural Networks model (SDGNN). In comparison with traditional GNNs, we take related social theories into account and redesign our aggregators and loss functions. For signed directed networks, we define four different signed directed relations. And we propose a layer-by-layer signed relation GNN to aggregate and propagate the information of nodes in signed networks. For training our model, we reconstruct three important parts of signed directed network: signs, directions, and triangles.

The major contributions of this paper are as follows:

- After reviewing the two fundamental social psychology theories of signed network analysis, we conduct an empirical analysis of these theories on five real-world datasets.
- We introduce a new layer-by-layer Signed Directed Relation GNN model. It aggregates and propagates information between nodes under different signed directed relation definitions. Guided by two sociological theories, our loss functions consist of reconstructing signs, directions, and triangles.
- We conduct link sign prediction experiments on five real-world signed social network datasets to demonstrate the effectiveness of our proposed model.

## Related Work

### Signed Network Embedding

Signed social networks are such social networks in signed social relations having both positive and negative signs (Easley and Kleinberg 2010). To mine signed networks, many algorithms have been developed for lots of tasks, such as community detection (Traag and Bruggeman 2009), node classification (Tang, Aggarwal, and Liu 2016), link prediction (Leskovec, Huttenlocher, and Kleinberg 2010b), spectral graph analysis (Li et al. 2018), and group partition (Huang and Luo 2018). Recently, with the development of network representation learning (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Tang et al. 2015), researchers begin to learn low-dimensional representations for signed networks. For signed network embedding methods, SNE (Yuan, Wu, and Xiang 2017) adopts the log-bilinear model and incorporates two signed-type vectors to capture the positive or negative relationship of each edge along the path. SiNE (Wang et al. 2017) designs an objective function guided by social theories to learn signed network embeddings. SiNE proposes that social theories can provide a fundamental understanding of signed social networks. For directed signed network, SIDE (Kim et al. 2018) provides a linearly scalable method that leverages balance theory along with random walks. SIGNet (Islam, Prakash, and Ramakrishnan 2018) combines balance theory with specialized random and new sampling techniques in directed signed networks. BESIDE (Chen et al. 2018a) mathematically models "bridge" edges based on balance and status theory and achieves state-of-the-art performances.

These methods are devoted to defining an objective function that incorporates sociological theory and then using machine learning techniques (e.g., sampling and random walks) to optimize look-up embeddings.

### Graph Neural Networks

Today's GNNs can be summarized as Message Passing Neural Networks (MPNNs), including message functions and vertex update functions (Gilmer et al. 2017). Because of the non-Euclidean data structure of the graph, traditional RNNs and CNNs are not easy to be used in the graph domains. By using GNNs, researchers have successfully applied convolution (Kipf and Welling 2019), attention (Veličković et al. 2018), LSTM (Hamilton, Ying, and Leskovec 2017), and other mechanisms into the graph data. Specifically, Graph Auto-Encoders (Kipf and Welling 2016; Pan et al. 2018) are a family of models aimed at mapping (encoding) each node to low-dimensional vectors which reconstructing (decoding) the graph. Compared to the network embedding methods, GNNs have a partial intersection but use the deep learning methods instead of matrix factorization and random walk and can better describe the network structure and node characteristics (Wu et al. 2020). A lot of GNN models show a better performance than the shadow look-up embeddings (Kipf and Welling 2019; Veličković et al. 2018; Hamilton, Ying, and Leskovec 2017).

Most Graph Neural Networks (Kipf and Welling 2019; Veličković et al. 2018; Hamilton, Ying, and Leskovec 2017; Xu et al. 2019, 2020) are designed for unsigned social networks whose links are only positive. How to apply graph neural networks to signed directed networks faces some challenges (e.g., how to model the negative links and the directions). SGCN (Derr, Ma, and Tang 2018) designs a new information aggregation and propagation mechanism for the undirected signed networks according to balance theory. SGCN applies a mean-pooling strategy that is close to GraphSAGE (Hamilton, Ying, and Leskovec 2017) to learning node embeddings. SiGAT (Huang et al. 2019) introduces GAT (Huang et al. 2019) to directed signed networks and designs a motif-based graph neural network model based on social theories. However, SiGAT uses 38 motifs, which is expensive for large graphs, and its objective functions can only model the sign, ignoring other vital features (e.g., directions and triangles).

## Problem Definition

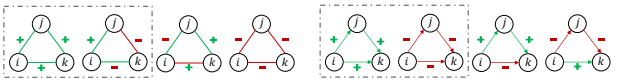
We define a signed directed network as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, s)$, where $\mathcal{V}$ is the set of nodes in a graph $\mathcal{G}$, and $\mathcal{E}$ is the edge list with signs $s$ and directions. $\mathcal{E}$ consist of $\mathcal{E}^+$ and $\mathcal{E}^-$ while $\mathcal{E}^+ \bigcap \mathcal{E}^- = \emptyset$; $\mathcal{E}^+$ and $\mathcal{E}^-$ denote the sets of positive and negative links, respectively. It can be denoted as the adjacency matrix of the signed network $A$, where $A_{ij} = 1$ means there exists a positive link from $u_i$ to $u_j$, $A_{ij} = -1$ denotes a negative link from $u_i$ to $u_j$, and $A_{ij} = 0$ means there is no link from $u_i$ to $u_j$. Given a signed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, s)$, our purpose is to map the nodes $u \in \mathcal{V}$ to low-dimensional vectors $z_u \in \mathbb{R}^d$ as:

$$f(A) \rightarrow Z, \tag{1}$$

where $Z \in \mathbb{R}^{|\mathcal{V}| \times d}$ is $d$-dimensional representations for the $|\mathcal{V}|$ nodes of the signed network, $f$ is a learned transformation function.

## Sociological Theory

Two sociological theories (i.e., Balance Theory and Status Theory) play an essential role in analyzing and modeling signed directed networks (Leskovec, Huttenlocher, and Kleinberg 2010b). In this section, we will briefly introduce these two theories and compare them in five real-world datasets.



(a) Balance theory (b) Status theory

Figure 1: Illustrations of two sociological theories.

### Balance Theory

Balance theory originated in the 1950s (Heider 1946) is initially intended as a model for undirected signed networks. Triads with an even number of negative edges as balanced. In Figure 1(a), the triangles with three positive signs and those with one positive sign (i.e., the first two triads in Figure 1(a)) are balanced. Balance theory posits that balanced triads are more plausible — and hence should be more prevalent in real-world networks — than unbalanced triads. It exemplifies the principle that *the friend of my friend is my friend* and *the enemy of my enemy is my friend*. Balance theory is widely used in the field of signed networks (Leskovec, Huttenlocher, and Kleinberg 2010a).

### Status Theory

Status theory is another critical sociological theory in signed network analysis, which provides a different organizing principle for directed networks of signed links. It supposes that a positive directed link "+" indicates that the creator of the link views the recipient as having higher status; and a negative directed link "-" indicates that the recipient is viewed as having lower status (Leskovec, Huttenlocher, and Kleinberg 2010b). The status may denote the relative prestige, ranking, or reputation. For example, a positive link from A to B means not only *B is A's friend* but also *B has a higher status than A*. For the triangles in Figure 1(b), the first two triads satisfy the status order, but the last two do not satisfy it. For the first triads, when $\text{Status(j)} > \text{Status(i)}$ and $\text{Status(k)} > \text{Status(j)}$, we have $\text{Status(k)} > \text{Status(i)}$.

### Comparison of Balance Theory and Status Theory

Balance theory focuses on undirected signed networks, although it has been applied to directed networks by simply disregarding the directions (Wasserman, Faust et al. 1994). While, status theory normally reflects relations between two users, which is based on directions. In some cases, these two theories can be consistent with predictions. This phenomenon has received a lot of attention from researchers (Leskovec, Huttenlocher, and Kleinberg 2010b; Chen et al. 2018a).

Table 1: The ratio of triads satisfying balance and/or status theory.

| Dataset | Both | Only Balance | Only Status | Neither |
|---|---|---|---|---|
| Bitcoin-Alpha | 0.673 | 0.208 | 0.094 | 0.025 |
| Bitcoin-OTC | 0.686 | 0.208 | 0.083 | 0.023 |
| Wikirfa | 0.686 | 0.059 | 0.189 | 0.066 |
| Slashdot | 0.751 | 0.167 | 0.066 | 0.016 |
| Epinions | 0.769 | 0.156 | 0.066 | 0.009 |

We follow the related works and examine the percentage of triads satisfying balance and/or status theory on five real-world datasets used in this paper.

From Table 1, we can find that only a tiny fraction of triangles satisfies neither of two theories. About 70% of triads can be consistent with both theories. We think these triangles that satisfy neither theory maybe the noise for the signed network embedding representation. In addition, balance theory models the relationship among three vertices, and status theory capture the relationship between two vertices with the transitivity property (Chen et al. 2018a). The complementation of both theories can be the key point for signed directed network representation learning.

## Proposed Method

Based on the previous discussion on sociological theory in signed directed networks, we will introduce how to design our new SDGNN model in this section.

### Signed Directed GNNs

For unsigned networks, neighborhood nodes have the same semantic relations. GraphSAGE (Hamilton, Ying, and Leskovec 2017) uses a mean aggregator to aggregate information from neighborhoods. As the weights in GraphSAGE only determined by network structure, GAT (Veličković et al. 2018) learns the weights by structure masked self-attention mechanism. We adapt the above GNNs aggregators to signed directed networks.

Figure 2 depicts the overall architecture of the proposed SDGNN model. For a signed directed graph, the directions and signs between nodes reflect different relations and semantics. Different neighborhoods with different relations should be distinguished. We first define 4 different signed directed relation (i.e., $u \rightarrow^{+} v, u \rightarrow^{-} v, u \leftarrow^{+} v, u \leftarrow^{-} v$). Based on a signed directed relation (SDR) $r_i$, we can get the neighborhoods $\mathcal{N}_{r_i}$. After that, we use different SDR GNN aggregators to aggregate the information from different neighborhoods and use an MLP to encode these messages into node embeddings. GNN aggregators can be mean aggregators or attention aggregators. For a mean aggregator, we get the information $X^l_{r_i}(u)$ and concatenate $z^l_u$ with $X^l(u)$ by:

$$\begin{aligned} X^l_{r_i}(u) &= \sigma(\mathbf{W}^l_{r_i} \cdot \text{MEAN}(\{z^l_u\} \cup \{z^l_v, \forall v \in \mathcal{N}_{r_i}(u)\})) \\ z^{l+1}_u &= \text{MLP}(\text{CONCAT}(X^l(u), X^l_{r_1}(u), ..., X^l_{r_i}(u))), \end{aligned} \tag{2}$$

where $z_u$ is the embedding of node $u$, $\mathcal{N}_{r_i}(u)$ is the neighborhoods of $u$ under the definition $r_i$, $\mathbf{W}$ is the parameter,

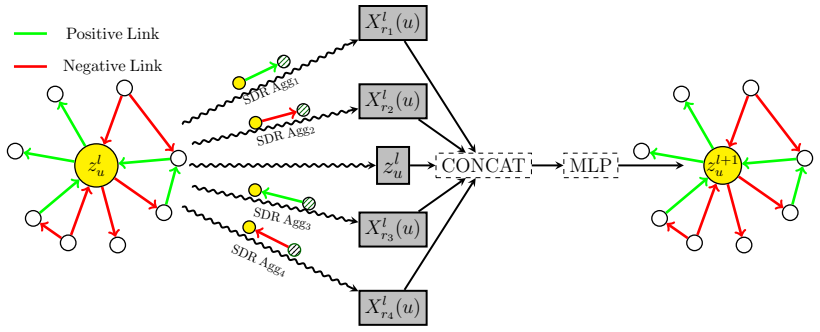



Figure 2: An Illustration of SDGNN model. We use Signed Directed Relation (SDR) aggregators to aggregate and propagate the information of nodes.

and $\sigma$ is the activation function.

For attention aggregators, we first compute $\alpha_{uv}^{r_i}$ for the node $u$ and node $v$ by the attention mechanism $\vec{\mathbf{a}}_{r_i}$ and LeakyReLU nonlinearity activation function (with negative input slope $\alpha = 0.2$) as:

$$\alpha_{uv}^{r_i} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}_{r_i}^{\top}[\mathbf{W}_{r_i}^{l} z_u^{l} \| \mathbf{W}_{r_i}^{l} z_v^{l}]\right)\right)}{\sum_{k \in \mathcal{N}_{r_i}(u)} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}_{r_i}^{\top}[\mathbf{W}_{r_i}^{l} z_u^{l} \| \mathbf{W}_{r_i}^{l} z_k^{l}]\right)\right)}, \tag{3}$$

where $\|$ is the concatenation operation, $\mathcal{N}_{r_i}(u)$ is the neighborhoods of node $u$ under the definition of $r_i$, $\mathbf{W}_{r_i}$ is the weight matrix parameter. Then we get the information $X_{r_i}^{l}(u)$ and concatenate $X_{r_i}^{l}(u)$ with $z_u$ to get new embeddings:

$$\begin{aligned} X_{r_i}^{l}(u) &= \sum_{v \in \mathcal{N}_{r_i}(u)} \alpha_{uv}^{r_i} \mathbf{W}_{r_i} z_v^{l}, \\ z_u^{l+1} &= \text{MLP}(\text{CONCAT}(X^{l}(u), X_{r_1}^{l}(u), ..., X_{r_i}^{l}(u))). \end{aligned} \tag{4}$$

In Figure 2, we use a layer-by-layer Signed Directed GNN to cover the direction and the topology of the triangle when we use multi-layers instead of one layer.

## Loss Function

For training our SDGNN model, we design three loss functions to reconstruct three critical features of signed networks: sign, direction, and triangle.

For sign, we use the following cross entropy loss function to model the sign between two nodes:

$$\begin{aligned} \mathcal{L}_{sign}(u,v) &= -y_{u,v} \log\left(\sigma(z_u^{\top} z_v)\right) - (1 - y_{u,v}) \log\left(1 - \sigma(z_u^{\top} z_v)\right), \\ \mathcal{L}_{sign} &= \sum_{e_{u,v} \in \mathcal{E}} \mathcal{L}_{sign}(u,v), \end{aligned} \tag{5}$$

where $\sigma$ is the sigmoid function, $y_{u,v}$ is the sign ground truth, and $\mathcal{E}$ is the edge list with signs.

As we discussed before, status theory relies on the directions in signed networks. We denote the status ranking score of node $u$, $v$ as $s(z_u)$, $s(z_v)$. We use the following square loss function to measure the difference between the predicted status relationship value $s(z_u) - s(z_v)$ from the edge $e_{uv}$ and the "ground truth" value $q_{uv}$:

$$\begin{aligned} \mathcal{L}_{direction}(u \to v) &= (q_{uv} - (s(z_u) - s(z_v)))^2 \\ q_{uv} &= \begin{cases} \max(s(z_u) - s(z_v),\ \gamma) & u \to v : - \\ \min(s(z_u) - s(z_v), \gamma) & u \to v : + \end{cases} \\ \mathcal{L}_{direction} &= \sum_{e_{u,v} \in \mathcal{E}} \mathcal{L}_{direction}(u \to v), \end{aligned} \tag{6}$$

where $s(z) = \text{sigmoid}(W \cdot z + b)$ is a score function for mapping embedding $z$ to a score, $\gamma$ is a threshold of status relationship value ($\gamma = 0.5$ in this paper), and $\mathcal{E}$ is the edge list with signs.

For triangles, we hope our model can learn from the true triangles distribution. For a triangle (e.g., $\triangle_{i,j,k}, i \to^{+} j, i \to^{+} k, k \to^{+} j$), we maximize the likelihood by:

$$J_{\triangle_{i,j,k}} = P(+|e_{ij}) * P(+|e_{ik}) * P(+|e_{kj}). \tag{7}$$

Further, triangles objective function $J_{tri}$ as follows:

$$\begin{aligned} J_{tri} &= \prod_{\triangle \in T} J_{\triangle}, \\ \mathcal{L}_{triangle} &= -\log J_{tri} = \sum_{\triangle \in T} -\log J_{\triangle} = \sum_{\triangle \in T} \mathcal{L}_{\triangle}, \end{aligned} \tag{8}$$

where $T$ is the set of triangles based on balanced and status theories. We can construct a triangle by three vertices $(i, j, k)$ by:

$$\begin{aligned} \mathcal{L}_{\triangle_{i,j,k}} &= \mathcal{L}_{ij} + \mathcal{L}_{ik} + \mathcal{L}_{kj}, \\ \mathcal{L}_{ij} &= -y_{i,j} \log P(+|e_{ij}) - (1 - y_{i,j}) \log(1 - P(+|e_{ij})) \\ &= -y_{i,j} \log\left(\sigma(z_i^{\top} z_j)\right) - (1 - y_{i,j}) \log(1 - \sigma(z_i^{\top} z_j)), \end{aligned} \tag{9}$$

where $y_{i,j}$ is the sign ground truth for edge $e_{ij}$. With Equation 8 and Equation 9, we can reconstruct triangles by edge binary cross entropy loss function. The weight of the edge loss function is the number of edges in triangles.

Based on sign, direction and triangle loss function, the overall objective function is written as:

$$\mathcal{L}_{loss} = \mathcal{L}_{sign} + \lambda_1 \mathcal{L}_{direction} + \lambda_2 \mathcal{L}_{triangle}, \tag{10}$$

where $\lambda_1$ and $\lambda_2$ are the weight of different loss functions. Equation 10 shows that our loss functions are designed to reconstruct the various properties of signed networks.

## Experiments

In this section, we conduct link sign prediction to check whether our model improves the performance of signed network embeddings. Link sign prediction is the task of predicting the unobserved sign of existing edges in the test dataset given training dataset (Leskovec, Huttenlocher, and Kleinberg 2010a; Chen et al. 2018a). We follow their experimental settings and compare our method against some state-of-the-art embedding methods.

### Experimental Settings

**Datasets** We do experiments on five real-world signed social network datasets (i.e., Bitcoin-Alpha[1], Bitcoin-otc[2], Wikirfa[3], Slashdot[4] and Epinions[5]). Bitcoin-Alpha and Bitcoin-OTC (Kumar et al. 2016) are the who-trust-whom networks of people who trade using Bitcoin on platforms. In these datasets, members rate other members on a scale of -10 (total distrust) to +10 (total trust) in steps of 1. We treat the scores greater than 0 as positive and others as negative. Wikirfa (Kumar et al. 2016) is a signed network in which nodes represent Wikipedia members and edges represent votes. It records the voting results for "request for adminship (RfA)", where the community member can cast a supporting, neutral, or opposing vote for a Wikipedia adminship election. We remove the neural votes and construct a signed network (Chen et al. 2018a). Slashdot (Leskovec, Huttenlocher, and Kleinberg 2010b) is from a technology-related news website with user communities. The website introduced Slashdot Zoo features that allow users to tag each other as friends or foes. The dataset is a common signed social network with friends and enemies labels. Epinions (Leskovec, Huttenlocher, and Kleinberg 2010b) is a who-trust-whom online social network of a consumer review site Epinions.com. Members of the site can indicate their trust or distrust of the reviews of others. The network reflects people's opinions on others.

Table 2: Statistics of five datasets.

| Dataset | # nodes | # pos links | # neg links | % pos ratio |
|---|---|---|---|---|
| Bitcoin-Alpha | 3,783 | 22,650 | 1,536 | 93.65 |
| Bitcoin-OTC | 5,881 | 32,029 | 3,563 | 89.99 |
| Wikirfa | 11,259 | 138,813 | 39,283 | 77.94 |
| Slashdot | 82,140 | 425,072 | 124,130 | 77.40 |
| Epinions | 131,828 | 717, 667 | 123,705 | 85.30 |

The statistics of five datasets are summarized in Table 2. For these five datasets, positive and negative links are imbalanced (i.e., nearly 80% links are positive edges).

**Baselines** To validate the effectiveness of SDGNN, we compare it with a number of baselines including unsigned network embedding methods, signed embedding methods and sigend graph neural networks.

- Random: It generates $d$ dimensional random values, $z = (x_1, x_2, ..., x_d), x_i \in [0.0, 1.0)$. It can be used to show the ability of the downstream logistic regression.
- Unsigned Network Embedding: We use several classical unsigned network embedding methods to validate the effectiveness of the structure: DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), Node2vec (Grover and Leskovec 2016), and LINE (Tang et al. 2015). Since these methods cannot distinguish positive and negative edges, we remove the negative links in the training stage.
- Signed Network Embedding: In order to show the effectiveness of modeling signed directed edges, we use some signed network embedding methods (e.g., , SiNE (Wang et al. 2017), SIGNet (Islam, Prakash, and Ramakrishnan 2018), BESIDE (Chen et al. 2018a)). Specifically, SiNE is designed for undirected signed networks. SIGNet and BESIDE can model signed directed networks.
- FeExtra (Leskovec, Huttenlocher, and Kleinberg 2010a): This method extracts two parts, a total of 23 features from signed directed social network. For each pair ($v_i$,$v_j$), the first one is the degree based, such as the number of incoming positive and negative links of $v_i$, the number of outgoing positive and negative links of $v_j$ and so on. The other one is the structure information of the triad that contains $v_i$ and $v_j$.
- Graph Neural Network: For the baselines of GNNs, we choose SGCN (Derr, Ma, and Tang 2018) and SiGAT (Huang et al. 2019). SGCN makes a dedicated and principled effort that utilizes balance theory to correctly aggregate and propagate the information across layers of a undirected signed GCN model. SiGAT uses 38 motifs and GAT aggregators to model signed directed networks.

For a fair comparison, the embedding dimension $d$ is set to 20 for all methods except FeExtra. It is a common setting used in SiNE (Wang et al. 2017), BESIDE (Chen et al. 2018a) and SiGAT (Huang et al. 2019). We use the authors' released code for DeepWalk[6], Node2vec[7], LINE[8], SiNE[9], SIGNet[10], BESIDE[11], and SiGAT[12]. For SGCN, we use the code from github[13]. We follow the authors' suggested hyperparameter settings. Like previous works (Kim et al. 2018; Wang et al. 2017; Derr, Ma, and Tang 2018), we first use these methods to get node representations. For edge $e_{ij}$, we concatenate these two learned representation $z_i$ and $z_j$ to compose an edge representation $z_{ij}$. After that, we train a logistic regression classifier on the training set and use it to predict the edge sign in the test set. We randomly select 80% edges as training set and the remaining 20% as the test set. We run with different train-test splits for 5 times to get the average scores. Each training set is used to train both

[1] http://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html
[2] http://snap.stanford.edu/data/soc-sign-bitcoin-otc.html
[3] http://snap.stanford.edu/data/wiki-RfA.html
[4] http://snap.stanford.edu/data/soc-sign-Slashdot090221.html
[5] http://snap.stanford.edu/data/soc-sign-epinions.html
[6] https://github.com/phanein/deepwalk
[7] https://github.com/aditya-grover/node2vec
[8] https://github.com/tangjianpku/LINE
[9] http://www.public.asu.edu/~swang187/codes/SiNE.zip
[10] https://github.com/raihan2108/signet
[11] https://github.com/yqc01/BESIDE
[12] https://github.com/huangjunjie95/SiGAT
[13] https://github.com/benedekrozemberczki/SGCN

Table 3: The results of link sign prediction on five datasets.

| | | Random | Unsigned Network Embedding | | | Signed Network Embedding | | | Feature Engineering | Graph Neural Network | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Metric | Random | Deepwalk | Node2vec | LINE | SiNE | SIGNet | BESIDE | FeExtra | SGCN | SiGAT | SDGNN |
| Bitcoin-Alpha | Micro-F1 | 0.9367 | 0.9367 | 0.9355 | 0.9352 | 0.9458 | 0.9422 | 0.9489 | 0.9486 | 0.9256 | 0.9456 | **0.9491** |
| | Binary-F1 | 0.9673 | 0.9673 | 0.9663 | 0.9664 | 0.9716 | 0.9696 | **0.9732** | 0.9730 | 0.9607 | 0.9714 | 0.9729 |
| | Macro-F1 | 0.4837 | 0.4848 | 0.6004 | 0.5220 | 0.6869 | 0.6965 | 0.7300 | 0.7167 | 0.6367 | 0.7026 | **0.7390** |
| | AUC | 0.6146 | 0.6409 | 0.7576 | 0.7114 | 0.8728 | 0.8908 | 0.8981 | 0.8882 | 0.8469 | 0.8872 | **0.8988** |
| Bitcoin-OTC | Micro-F1 | 0.9000 | 0.8937 | 0.9089 | 0.8911 | 0.9095 | 0.9229 | 0.9320 | **0.9361** | 0.9078 | 0.9268 | 0.9357 |
| | Binary-F1 | 0.9473 | 0.9434 | 0.9507 | 0.9413 | 0.9510 | 0.9581 | 0.9628 | **0.9653** | 0.9491 | 0.9602 | 0.9647 |
| | Macro-F1 | 0.4737 | 0.5281 | 0.6793 | 0.5968 | 0.6805 | 0.7386 | 0.7843 | 0.7826 | 0.7306 | 0.7533 | **0.8017** |
| | AUC | 0.6145 | 0.6596 | 0.7643 | 0.7248 | 0.8571 | 0.8935 | **0.9152** | 0.9121 | 0.8755 | 0.9055 | 0.9124 |
| Wikirfa | Micro-F1 | 0.7797 | 0.7837 | 0.7814 | 0.7977 | 0.8338 | 0.8384 | 0.8589 | 0.8346 | 0.8489 | 0.8457 | **0.8627** |
| | Binary-F1 | 0.8762 | 0.8779 | 0.8719 | 0.8827 | 0.8972 | 0.9001 | 0.9117 | 0.8987 | 0.9069 | 0.9042 | **0.9142** |
| | Macro-F1 | 0.4381 | 0.4666 | 0.5626 | 0.5738 | 0.7319 | 0.7384 | 0.7803 | 0.7235 | 0.7527 | 0.7535 | **0.7849** |
| | AUC | 0.5423 | 0.5876 | 0.6930 | 0.6772 | 0.8602 | 0.8682 | **0.8981** | 0.8604 | 0.8563 | 0.8829 | 0.8898 |
| Slashdot | Micro-F1 | 0.7742 | 0.7738 | 0.7526 | 0.7489 | 0.8265 | 0.8389 | 0.8590 | 0.8472 | 0.8296 | 0.8494 | **0.8616** |
| | Binary-F1 | 0.8728 | 0.8724 | 0.8528 | 0.8525 | 0.8918 | 0.8983 | 0.9105 | 0.9070 | 0.8926 | 0.9055 | **0.9128** |
| | Macro-F1 | 0.4364 | 0.4384 | 0.5390 | 0.5052 | 0.7273 | 0.7554 | **0.7892** | 0.7399 | 0.7403 | 0.7671 | **0.7892** |
| | AUC | 0.5370 | 0.5408 | 0.6709 | 0.6145 | 0.8409 | 0.8752 | **0.9017** | 0.8880 | 0.8534 | 0.8874 | 0.8977 |
| Epinions | Micro-F1 | 0.8525 | 0.8214 | 0.8563 | 0.8535 | 0.9173 | 0.9113 | 0.9336 | 0.9226 | 0.9112 | 0.9293 | **0.9355** |
| | Binary-F1 | 0.9204 | 0.9005 | 0.9170 | 0.9175 | 0.9525 | 0.9489 | 0.9615 | 0.9561 | 0.9486 | 0.9593 | **0.9628** |
| | Macro-F1 | 0.4602 | 0.5131 | 0.6862 | 0.6305 | 0.8160 | 0.8060 | 0.8601 | 0.8130 | 0.8105 | 0.8454 | **0.8610** |
| | AUC | 0.5589 | 0.6702 | 0.8081 | 0.6835 | 0.8872 | 0.9095 | 0.9351 | **0.9444** | 0.8745 | 0.9333 | 0.9411 |

embedding vectors and logistic regression classifiers. Our models are implemented by PyTorch with the Adam optimizer ($\mathrm{LearningRate} = 0.001$, $\mathrm{WeightDecay} = 0.001$). We use the 2-layer-GAT aggregators to build our model and set $\lambda_1 = 1$ and $\lambda_2 = 1$. All experiments run on a computer with Intel Xeon E5-2640 CPU and 128GB RAM, which installs Linux CentOS 7.1.

## Experiment Results

We report the average Micro-F1, Binary-F1, Macro-F1, and AUC in Table 3 (Pedregosa et al. 2011). We have bolded the highest value of each row and underlined the second value. From Table 3, we can find that:

- For signed networks, link sign prediction is a positive and negative imbalance classificiation problem. Even given random embedding, logistic regression can be used as the downstream machine learning methods.
- After using unsigned network embedding methods, even the only positive links are used, the metrics have been increased. It means that the structural information matters. Node2vec has made the best results in the unsigned network embeddings methods.
- Signed network embedding methods (i.e., SiNE, SIGNet, and BESIDE) are designed for signed network. The results are significantly higher than other unsigned network embedding methods. These algorithms model related sociological theories and had achieved good results for signed network analysis. SiNE is designed for undirected signed networks based on balance theory. Its results are not as good as SIGNet and BESIDE. Although SIGNet can be applied to signed directed networks, it does not model status theory. This makes it is less effective than BESIDE. BESIDE shows some good results in our experiments, which demonstrates that modeling two theories is the key for the problem.
- FeExtra method is a very classic method in the early days and performs well because of its successful use of relevant sociological theories. However, it should be pointed out that this method relies on feature engineering to extract features manually and only model edge features without node embeddings, so its generalization ability is weak.
- SGCN shows a performance close to SiNE, but it cannot effectively model signed directed networks because the algorithm did not consider the direction. Besides, it uses the $\mathrm{MEAN}$ aggregators which can not model the effects of different neighborhoods. SiGAT is designed for directed networks and uses the attention aggregators. Its experimental results are better than SGCN. But its decoder design limits its expressiveness.
- SDGNN outperforms all baseline methods in most terms of metrics. It demonstrates the ability of learning node embeddings using our SDGNN model. It is worth mentioning that the experimental results depend on the downstream classifier, and there may exist some inconsistent between F1 and AUC.

## Parameter Analysis and Ablation Study

In this section, we investigate the effects of hyperparameters and do some ablation studies to analyze our model architecture design. We choose the Bitcoin-Alpha as our dataset and select 80% training edges and 20% test edges as the previous subsection does. We also use logistic regression function as our downstream machine learning classifier.

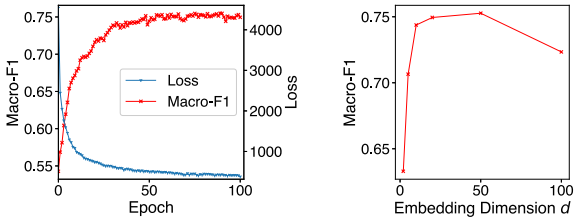
**Parameter Analysis** In this subsection, we analyze the hyperparameters about the number of epoch and the embedding dimension $d$.

Table 4: Ablation study on different aggregators.

| Metric | 2-Layer-GAT-AGG | 1-Layer-GAT-AGG | 2-Layer-MEAN-AGG | 1-Layer-MEAN-AGG |
|---|---|---|---|---|
| Micro-F1 | 0.9446 | 0.9442 | 0.9415 | 0.9399 |
| Binary-F1 | 0.9706 | 0.9703 | 0.9689 | 0.9679 |
| Macro-F1 | 0.7510 | 0.7516 | 0.7417 | 0.7411 |
| AUC | 0.9154 | 0.9095 | 0.9041 | 0.9000 |

Table 5: Ablation study on loss functions.

| Metric | $\mathcal{L}_{sign}$ | $\mathcal{L}_{sign} + \mathcal{L}_{direction}$ | $\mathcal{L}_{sign} + \mathcal{L}_{triangle}$ | $\mathcal{L}_{sign} + \mathcal{L}_{direction} + \mathcal{L}_{triangle}$ |
|---|---|---|---|---|
| Micro-F1 | 0.9386 | 0.9438 | 0.9415 | 0.9475 |
| Binary-F1 | 0.9677 | 0.9702 | 0.9690 | 0.9721 |
| Macro-F1 | 0.6738 | 0.7414 | 0.7210 | 0.7585 |
| AUC | 0.8883 | 0.9082 | 0.9030 | 0.9109 |




(a) Epoch (b) $d$

Figure 3: Parameter analysis for sign prediction.

When analyzing epoch, we set $d = 20$, and record the Macro-F1 performance of different epoch generated representations and the loss value during training. Figure 3(a) shows that SDGNN converges quickly and keeps relatively stable performances. When discussing the dimension $d$, we set the corresponding Epoch to 100 to discuss the robustness of different dimensions. Figure 3(b) shows that the performance increases first and decreases in 100 (We decrease the learning rate for the large $d$). Even with a small embedding dimension like $d = 5$, our model has already achieved pretty good performance which is close to BESIDE (Chen et al. 2018a). We find that too many parameters will cause the difficulties of training embeddings, which may be the reason for the decrease in large dimensions.

**Ablation Study** In this subsection, we do some ablation studies to discuss the design of aggregators and loss functions. As the same as previous subsection, we choose Bitcoin-Alpha as the dataset to analyze different architecture designs. For the aggregators, as we mentioned, mean aggregator and attention aggregator are common GNNs models. We analyze whether the attention mechanism can perform better than mean strategy. In addition, we also discusse the number of GNN layers to verify our layer-by-layer design. Table 4 demonstrates that the attention mechanism performs better than the mean strategy. And the number of layers can capture higher-order information, which can increase the performance.

For our loss functions, we can control $\lambda_1$ and $\lambda_2$ in Equation 10 to discuss the effects of different objectives on the results. When $\lambda_1 = 0$ and $\lambda_2 = 0$, it degenerates to just reconstruct the sign of signed networks, which is close to SiGAT. In Table 5, we can find that only reconstructing signs using $\mathcal{L}_{sign}$ performs poorly. When considering direction and triangle, the results are improved. It demonstrates that our training objectives should take both directions and triangles into consideration. Signs, directions, and triangles are vital features for signed directed networks.

## Conclusion

In this paper, we investigate the signed directed network representations learning. We firstly analyze two fundamental social theories (i.e., Balance Theory and Status Theory) in the signed directed network. Guided by sociological theories, we propose SDGNN to encode a signed network into network embeddings. SDGNN aggregates messages from different signed directed relation definitions. It can apply multi-layers to capture high order structure information. To train our SDGNN, we introduce combined loss functions to reconstruct not only the signs but also other important features for signed directed networks (i.e., directions and triangles). We perform experiments on five real-world signed networks and demonstrate that our proposed SDGNN performs better than other state-of-art baselines. We analyze the hyperparameters and do some ablation studies to analyze our design for aggregators and loss functions. In future work, we will further generalize this method to heterogeneous networks to incorporate more complex semantic information (Schlichtkrull et al. 2018).

## Acknowledgments

The authors would like to thank the AAAI reviewers for their insightful suggestions to improve the manuscript. This work is funded by the the National Key R&D Program of China 2020AAA0105200 and National Natural Science Foundation of China under Grant No. 91746301 and 61802371. This work is supported by Beijing Academy of Artificial Intelligence (BAAI) under Grant No. BAAI2019QN0304. Huawei Shen is also funded by K.C. Wong Education Foundation.

## References


Chen, Y.; Qian, T.; Liu, H.; and Sun, K. 2018a. " Bridge" Enhanced Signed Directed Network Embedding. In *CIKM*, 773–782.

Chen, Y.; Qian, T.; Zhong, M.; and Li, X. 2018b. BASSI: Balance and Status Combined Signed Network Embedding. In *DASFAA*, 55–63. Springer.

Cui, J.; Zhuang, H.; Liu, T.; and Wang, H. 2020. Semi-Supervised Gated Spectral Convolution on a Directed Signed Network. *IEEE Access* 8: 49705–49716.

Derr, T.; Ma, Y.; and Tang, J. 2018. Signed graph convolutional networks. In *ICDM*, 929–934. IEEE.

Duran, A. G.; and Niepert, M. 2017. Learning graph representations with embedding propagation. In *NIPS*, 5119–5130.

Easley, D.; and Kleinberg, J. 2010. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for Quantum chemistry. In *ICML*, 1263–1272.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *KDD*, 855–864. ACM.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, 1025–1035.

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

Huang, J.; and Luo, T. 2018. Computing Len for Exploring the Historical People's Social Network. In *FiCloudW*, 95–101. IEEE.

Huang, J.; Shen, H.; Hou, L.; and Cheng, X. 2019. Signed graph attention networks. In *ICANN*, 566–577. Springer.

Islam, M. R.; Prakash, B. A.; and Ramakrishnan, N. 2018. Signet: Scalable embeddings for signed networks. In *PAKDD*, 157–169. Springer.

Kim, J.; Park, H.; Lee, J.-E.; and Kang, U. 2018. Side: representation learning in signed directed networks. In *WWW*, 509–518.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*.

Kipf, T. N.; and Welling, M. 2019. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kumar, S.; Hamilton, W. L.; Leskovec, J.; and Jurafsky, D. 2018. Community interaction and conflict on the web. In *WWW*, 933–943.

Kumar, S.; Spezzano, F.; Subrahmanian, V.; and Faloutsos, C. 2016. Edge weight prediction in weighted signed networks. In *ICDM*, 221–230. IEEE.

Leskovec, J.; Huttenlocher, D.; and Kleinberg, J. 2010a. Predicting positive and negative links in online social networks. In *WWW*, 641–650. ACM.

Leskovec, J.; Huttenlocher, D.; and Kleinberg, J. 2010b. Signed networks in social media. In *CHI*, 1361–1370. ACM.

Li, Y.; Tian, Y.; Zhang, J.; and Chang, Y. 2020. Learning Signed Network Embedding via Graph Attention. In *AAAI*, 4772–4779.

Li, Y.; Yuan, S.; Wu, X.; and Lu, A. 2018. On spectral analysis of directed signed graphs. *International Journal of Data Science and Analytics* 6(2): 147–162.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *IJCAI*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *JMLR* 12: 2825–2830.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710. ACM.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Tang, J.; Aggarwal, C.; and Liu, H. 2016. Node classification in signed social networks. In *SIAM*, 54–62. SIAM.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*, 1067–1077.

Traag, V. A.; and Bruggeman, J. 2009. Community detection in networks with positive and negative links. *Physical Review E* 80(3): 036115.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. *ICLR* URL https://openreview.net/forum?id=rJXMpikCZ.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *KDD*, 1225–1234. ACM.

Wang, S.; Tang, J.; Aggarwal, C.; Chang, Y.; and Liu, H. 2017. Signed network embedding in social media. In *SIAM*, 327–335. SIAM.

Wasserman, S.; Faust, K.; et al. 1994. *Social network analysis: Methods and applications*, volume 8. Cambridge university press.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* .

Xu, B.; Huang, J.; Hou, L.; Shen, H.; Gao, J.; and Cheng, X. 2020. Label-Consistency based Graph Neural Networks for Semi-supervised Node Classification. In *SIGIR*, 1897–1900.

Xu, B.; Shen, H.; Cao, Q.; Qiu, Y.; and Cheng, X. 2019. Graph wavelet neural network. In *ICLR*.

Yuan, S.; Wu, X.; and Xiang, Y. 2017. Sne: signed network embedding. In *PAKDD*, 183–195. Springer.

# Appendix

## Comparison of Different Model Architectures

In general, our model is a layer-by-layer signed relation GNN model with social theories guided loss functions. We list the difference between the proposed method with existing baselines in Table 6.

Table 6: Summary of selected methods.

| Method | SDGNN | BESIDE | SiGAT |
|---|---|---|---|
| GNN Aggregator | ✓ | ✗ | ✓ |
| Layer by Layer | ✓ | ✗ | ✗ |
| Sign Loss | ✓ | ✗ | ✓ |
| Direction Loss | ✓ | ✓ | ✗ |
| Triangle Loss | ✓ | ✓ | ✗ |

## Algorithm Detail

When our model work on a large signed directed graph whose number of nodes is more than 100,000, we could not put the whole graph into our memory with $L$-layer GNN aggregators ($L \geq 1$).

To train our SGDNN model in such large networks, we use mini-batch stochastic gradient descent to update the parameters SDGNN. It needs to recombine the neighborhoods of the nodes in the batch to achieves batch calculation. In this paper, the batch size is 500. The training procedure is summarized in Algorithm 1.

**Algorithm 1:** SDGNN Algorithm

**Input:** Signed Directed Graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, s)$;
Encoder Aggregators $Enc$;
GNN Layer Number $L$;
Epoch $T$;
**Output:** Node representation $Z$

Prepare Original node embeddings $z_u^0, \forall u \in V$.
Initialize the parameters of neural networks.
**for** $epoch = 1, ..., T$ **do**
  **for** each mini-batch node-set $B$ from $\mathcal{V}$ **do**
    Get neighborhoods $\mathcal{N}(v), \forall v \in B$
    **for** $u \in B$ **do**
      **for** $l = 1...L$ **do**
        Get neighborhoods embeddings $Z^l_{\mathcal{N}(v)}, \forall v \in B$
        $z_u^{l+1} \leftarrow Enc^l(z_u^l, Z^l_{\mathcal{N}(u)})$
      **end for**
    **end for**
    $Z_B \leftarrow z_v^L, \forall v \in B$
    Compute loss $\sum_{v \in B} \mathcal{L}_{loss}(v)$ with $Z_B$
    Back propagation, update parameters.
  **end for**
**end for**
**return** $Z = Enc^L(V)$

From Algorithm 1, we can find that the number of computed neighbors increases exponentially, which is consistent with GraphSAGE (Hamilton, Ying, and Leskovec 2017). In this paper, we set $L = 2$, which can capture triangles structure information.

## Experiment Detail

As we said in Experimental Settings section, the dataset used in this paper is public and can be downloaded from the corresponding links. Data preprocessing is also discussed in Experiment Setting section, including how to binarize the networks. In wikirfa dataset, there exits duplicate links (i.e., A $\rightarrow^+$ B and A $\rightarrow^-$ B ), we use the last record in the edge list. That's why it is slightly different from Chen et al.. Besides, we number the nodes to make string id into a number id, which is eaiser to be vectorized. These numbers are completed in the data preprocessing and are same to all embedding methods.

## Complexity Analysis

We give complexity analysis in this section. Firstly, in the dataset given in Dataset section, no node attribute information is provided. For graph neural network methods (i.e., SGCN, SiGAT and SDGNN), there are usually two types of methods: using id embedding or using spectral analysis methods. For our SDGNN, we use a learnable id embeddings for the initial node representation. It's $O(|\mathcal{V}| * d)$, where $d$ is the embedding dimensions. Then, when we have $L$-layer GNN aggregators, we have the $\mathbf{W}$ whose complexity is $O(L*d*d)$. Besides, we have a two-layer MLP to encoder the memssage for neighborhoods $\mathcal{N}(v)$ and node representation $z_v$. The complexity of this part is $O(L*10*d*d)$. In total, the complexity of SGNN is $O(|\mathcal{V}|*d+11*L*d*d)$. For attention aggregators, it will include an additional attention vector, whose complexity is $O(d)$.

## Various Values of $\lambda_1$ and $\lambda_2$

We show the results of various value of $\lambda_1$ and $\lambda_2$ in Table 7. From Table 7, we can find that $\lambda_1$ and $\lambda_2$ are not zero, showing better performance. Since reconstructing the sign is the most fundamental task, we don't set it to zero.

Table 7: Various Values of $\lambda_1$ and $\lambda_2$.

| | Metric | $\lambda_2 = 0$ | $\lambda_2 = 1$ | $\lambda_2 = 5$ | $\lambda_2 = 10$ |
|---|---|---|---|---|---|
| $\lambda_1 = 0$ | Micro-F1 | 0.9386 | 0.9415 | 0.9411 | 0.9413 |
| | Binary-F1 | 0.9677 | 0.9690 | 0.9688 | 0.9689 |
| | Macro-F1 | 0.6738 | 0.7210 | 0.7267 | 0.7272 |
| | AUC | 0.8883 | 0.9030 | 0.8991 | 0.9008 |
| $\lambda_1 = 1$ | Micro-F1 | 0.9438 | 0.9475 | 0.9452 | 0.9440 |
| | Binary-F1 | 0.9702 | 0.9721 | 0.9709 | 0.9703 |
| | Macro-F1 | 0.7414 | 0.7585 | 0.7484 | 0.7392 |
| | AUC | 0.9082 | 0.9109 | 0.9070 | 0.9014 |
| $\lambda_1 = 5$ | Micro-F1 | 0.9456 | 0.9440 | 0.9461 | 0.9436 |
| | Binary-F1 | 0.9711 | 0.9702 | 0.9714 | 0.9700 |
| | Macro-F1 | 0.7553 | 0.7495 | 0.7531 | 0.7476 |
| | AUC | 0.9139 | 0.9152 | 0.9123 | 0.9088 |
| $\lambda_1 = 10$ | Micro-F1 | 0.9471 | 0.9465 | 0.9473 | 0.9452 |
| | Binary-F1 | 0.9719 | 0.9715 | 0.9720 | 0.9709 |
| | Macro-F1 | 0.7598 | 0.7598 | 0.7627 | 0.7526 |
| | AUC | 0.9154 | 0.9162 | 0.9118 | 0.9127 |