ASiNE: Adversarial Signed Network Embedding

Yeon-Chang Lee
Hanyang University
Seoul, Korea
lyc0324@hanyang.ac.kr

Nayoun Seo
Hanyang University
Seoul, Korea
nayounseo@hanyang.ac.kr

Kyungsik Han
Ajou University
Suwon, Korea
kyungsikhan@ajou.ac.kr

Sang-Wook Kim*
Hanyang University
Seoul, Korea
wook@hanyang.ac.kr

ABSTRACT

Motivated by a success of *generative adversarial networks* (GAN) in various domains including information retrieval, we propose a novel signed network embedding framework, ASiNE, which represents each node of a given signed network as a low-dimensional vector based on the *adversarial learning*. To do this, we first design a generator G^+ and a discriminator D^+ that consider positive edges, as well as a generator G^- and a discriminator D^- that consider negative edges: (1) G^+/G^- aim to generate the most indistinguishable *fake* positive/negative edges, respectively; (2) D^+/D^- aim to discriminate between *real* positive/negative edges and *fake* positive/negative edges, respectively. Furthermore, under ASiNE, we propose two new strategies for effective signed network embedding: (1) an *embedding space sharing* strategy for learning both positive and negative edges; (2) a *fake edge generation* strategy based on the *balance theory*. Through extensive experiments using five real-life signed networks, we verify the effectiveness of each of the strategies employed in ASiNE. We also show that ASiNE consistently and significantly outperforms all the state-of-the-art signed network embedding methods in all datasets and with all metrics in terms of accuracy of sign prediction.

CCS CONCEPTS

• **Information systems → Data mining**.

KEYWORDS

signed network embedding; adversarial learning; balance theory

ACM Reference Format:
Yeon-Chang Lee, Nayoun Seo, Kyungsik Han, and Sang-Wook Kim. 2020. ASiNE: Adversarial Signed Network Embedding. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401079

*Corresponding author.

1 INTRODUCTION

Network embedding (NE) aims to represent nodes of a given network by vectors preserving structural properties [17], such as proximity in the network space, in a single low-dimensional embedding space [7]. Literature has shown that the low-dimensional vectors can be used as effective features for nodes in solving various machine learning tasks related to information retrieval (IR) including edge prediction [25], node clustering [38], node classification [33], and recommendation [6, 14, 22]. Examples of NE methods include DeepWalk [31], Node2vec [11], LINE [34], GCN [12], and GraphGAN [36]. Intuitively, each example attempts to represent the nodes in the proximity located closely each other in the embedding space.

With the emergence of *signed networks* with both positive and negative edges between nodes, we can better understand the complex relationships between the nodes based on rich information obtained from the edge signs [15, 20, 21, 24]. For example, on a product review website, Epinions[1], a user can decide whether to trust (*i.e.*, positive edges) or distrust (*i.e.*, negative edges) others. However, since existing *unsigned* NE methods [11, 12, 31, 34, 36] assume that there is only one type of edges (*i.e.*, positive edges) in the network, they have inherent limitations in effectively utilizing the rich information provided by such signed networks.

To address these limitations, many researchers have extended existing unsigned NE methods to signed networks, named as *signed NE* [9, 18, 23, 37, 39, 40]. Examples include SNE [40], SIDE [18], SGCN [9], and SLF [39]. Basically, they attempt to represent the nodes with the positive edges to be close and those with the negative edges to be distant in the embedding space. Furthermore, they employ a *balance theory* [3, 13], a well-known theory in social sciences, to exploit more complex relationships between nodes by combining positive and negative edges. The balance theory states that social relationships of the real world follow four rules: “a friend of my friend is my friend,” “a friend of my enemy is my enemy,” “an enemy of my friend is my enemy,” and “an enemy of my enemy is my friend.” Most signed NE methods [9, 18] regard the friends and enemies newly identified by the balance theory as inherent positive and negative edges, respectively.

However, they have focused only on building either a generative [18, 39, 40] or discriminative model [9]. Recently, Goodfellow et al. [10] proposed *generative adversarial networks* (GAN) that combine generative and discriminative models to play a minimax game. The *adversarial learning* approaches, including GAN and its

[1] www.epinions.com

variants, have already been successful in many applications such as IR [26], image generation [8], and recommendation [4, 5]. Motivated by such a success, Wang et al. [36] proposed an excellent idea of GraphGAN which is a variant of GAN *for the unsigned NE*. In [36], they showed that the adversarial learning helps to improve the performance of NE by making the generator learn an *underlying connectivity distribution* of an unsigned network under the guidance provided by the discriminator. Nevertheless, GraphGAN does not take into account the edge signs, thereby being applicable only to unsigned networks.

Therefore, we propose a novel signed NE method, named as ASiNE, which can represent nodes of a signed network as low-dimensional vectors *based on adversarial learning*. To do this, we first divide the given signed network $\mathcal{G}$ into a subgraph $\mathcal{G}^+$ consisting of all nodes and only positive edges and a subgraph $\mathcal{G}^-$ consisting of all nodes and only negative edges. Then, we design two generator models G^+/G^- and two discriminator models D^+/D^- that can play a *minimax game* in $\mathcal{G}^+/\mathcal{G}^-$, respectively. The final goal of this minimax game is to make G^+/G^- learn the *underlying positive/negative connectivity distribution* of $\mathcal{G}^+/\mathcal{G}^-$ under the guidance provided by D^+/D^-. Towards this goal, ASiNE learns each of the four models as follows:

- G^+/G^- aim to generate the most indistinguishable *fake* positive/negative edges by considering *both* the structure of $\mathcal{G}^+/\mathcal{G}^-$ and the balance theory, respectively.
- D^+/D^- aim to discriminate between *real* positive/negative edges and *fake* positive/negative edges, respectively.

In the above learning process, G^+ tries to represent the two nodes incident to a *fake* positive edge to be close in the embedding space; G^- tries to represent the two nodes incident to a *fake* negative edge to be distant. On the other hand, D^+ tries to represent the two nodes incident to *real* and *fake* positive edges to be close and distant, respectively; D^- tries to represent the two nodes incident to *real* and *fake* negative edges to be distant and close, respectively. To reflect positive and negative edges together, we propose an *embedding space sharing strategy* in which G^+ and G^- share *one embedding space for generation* and D^+ and D^- share *another embedding space for discrimination*.

Note that G^+ and G^- generate fake positive and negative edges from $\mathcal{G}^+$ and $\mathcal{G}^-$, respectively. However, we can let G^- also generate *fake positive edges from* $\mathcal{G}^-$ by exploiting "an enemy of my enemy is my friend" of the balance theory. Thus, in this paper, we propose a strategy in which G^- generates not only fake negative edges but also fake positive edges. The fake positive edges generated by G^- help us fully learn rich information of $\mathcal{G}^-$.

Through extensive experiments using five real-life datasets, we validate the effectiveness of ASiNE. The experimental results show that (1) our strategies used in ASiNE for signed NE are all effective for more-accurate embedding, (2) ASiNE consistently and significantly outperforms all four state-of-the-art signed NE methods, SNE [40], SIDE [18], SGCN [9], and SLF [39], in terms of the accuracy of edge sign prediction, and (3) ASiNE has a linear scalability with the increasing number of edges.

Our contributions are summarized as follows:

- We propose a novel signed NE method, ASiNE, based on *adversarial learning*. To the best of our knowledge, this paper is the first attempt that utilizes adversarial learning for signed NE.
- We design the objective functions of our generators G^+/G^- and discriminators D^+/D^-, which allow to play a *minimax game* in $\mathcal{G}^+/\mathcal{G}^-$.
- We propose a strategy in which G^+ and G^- learn *one shared embedding space* and D^+ and D^- learn *another shared embedding space* to reflect positive and negative edges together.
- We propose a strategy in which G^- generates not only *fake negative edges* but also *fake positive edges* from $\mathcal{G}^-$.
- Through extensive experiments using five real-life datasets, we validate the effectiveness of ASiNE, showing ASiNE improves the accuracies of the competing methods significantly in all cases.

The rest of this paper is organized as follows: Section 2 reviews existing work for NE. Section 3 presents our proposed approach in detail. Section 4 validates the effectiveness of the proposed approach through extensive experiments. Finally, Section 5 concludes the paper and discusses future research directions.

2 RELATED WORK

The learning process of most NE methods can be summarized in the following two steps: (1) *edge sampling* that determines node pairs with strong relationships and (2) *likelihood optimization* that tries to place them preserving the pair-wise relationships in the embedding space. In this section, we briefly review the representative unsigned and signed NE methods by focusing on the above two steps.

Unsigned network embedding. DeepWalk [31] obtains node sequences for each target node v_c with a truncated random walk, and samples the *context nodes* for each node within the sequences. Then, it optimizes the likelihood of observing the context nodes for each node. Node2vec [11] extends the idea of DeepWalk. It obtains node sequences for each target node v_c with a biased random walk that considers *breadth first search* (BFS) and *depth first search* (DFS) together. Then, it performs likelihood optimization in the same way as DeepWalk.

Unlike DeepWalk and Node2vec, LINE [34] randomly samples directly connected nodes for each node v_c. Then, it designs two objective functions that preserve first- and second-order proximities using the sampled edges. It obtains two kinds of node embeddings by minimizing these two objectives, and take their concatenation as the final node embedding.

Recently, *graph convolutional network* (GCN) [12] exploits all edges of the network without edge sampling. It iteratively aggregates the embeddings of direct neighbors (*i.e.*, local neighborhood) for each target node v_c by defining a convolution operator on the network. Multiple iterations allow the learned embedding of v_c to consider both global and local neighborhoods.

GraphGAN [36] leverages GAN [10] to facilitate NE. It works through two components: (1) generator $G(v|v_c)$, which tries to generate the likely connected nodes obtained from random walks, and (2) discriminator $D(v, v_c)$, which tries to differentiate the node pairs generated by $G(v|v_c)$ from the ground truth. Under the GraphGAN framework, the discriminator maximizes the likelihood for real edges, while the generator maximizes the likelihood for sampled fake edges. In [36], the authors showed that the adversarial

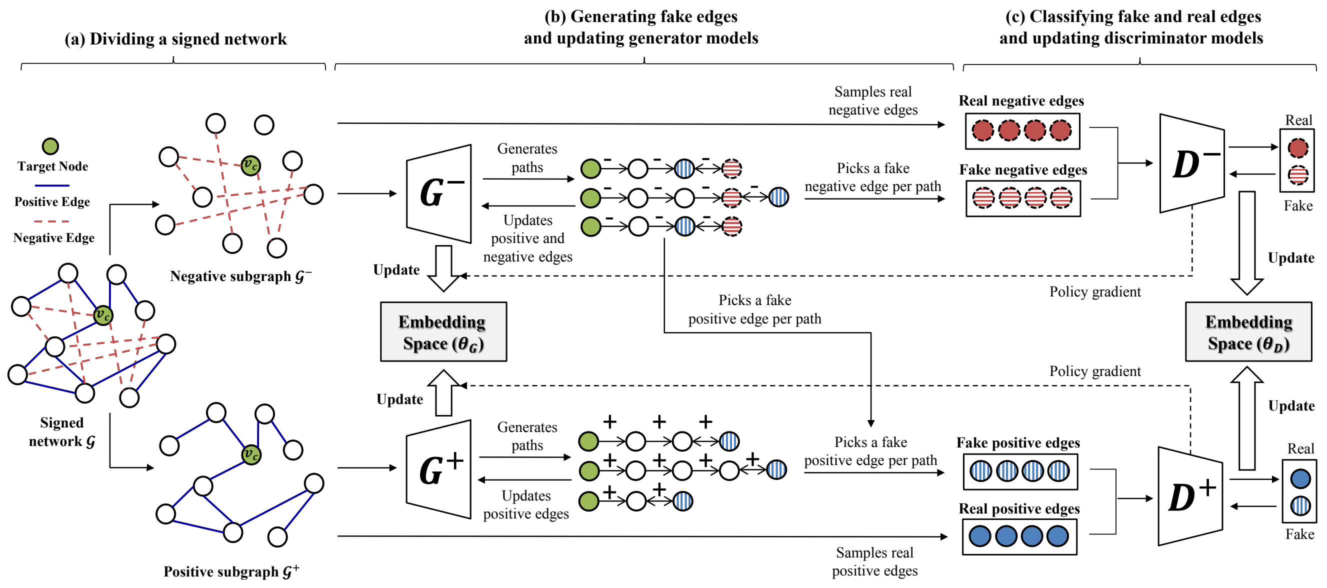
(a) Dividing a signed network
(b) Generating fake edges and updating generator models
(c) Classifying fake and real edges and updating discriminator models
Target Node
Positive Edge
Negative Edge
Negative subgraph $\mathcal{G}^-$
Signed network $\mathcal{G}$
Positive subgraph $\mathcal{G}^+$
G^-
G^+
Generates paths
Updates positive and negative edges
Updates positive edges
Update
Embedding Space (θ_G)
Embedding Space (θ_D)
Picks a fake positive edge per path
Picks a fake negative edge per path
Samples real negative edges
Samples real positive edges
Real negative edges
Fake negative edges
Fake positive edges
Real positive edges
D^-
D^+
Real
Fake
Policy gradient

Figure 1: Overview of the ASiNE framework.

learning helps to improve the performance of NE by optimizing the generator under the guidance provided by the discriminator.

Although the aforementioned studies provided important insights on NE, their approaches are applicable only to unsigned networks.

Signed network embedding. SNE [40] and SIDE [18] extend the ideas of DeepWalk and Node2vec. First, SNE [40] samples context nodes for each target node v_c based on a truncated random walk. Then, it optimizes the likelihood of observing the context nodes for each node based on the extended log-bilinear model [29]. However, it does not consider the balance theory.

SIDE [18] obtains node sequences for each target node v_c with a truncated random walk. Then, it samples node pairs with a positive sign and those with a negative sign in the sequences by considering the balance theory. It optimizes the likelihood for node pairs with positive and negative signs.

SGCN [9] extends GCN. Similar to GCN, SGCN does not perform node sampling, but utilizes all the direct neighbors of each target node v_c. Then, to consider multi-hop neighbors for v_c, SGCN utilizes the balance theory to correctly aggregate and propagate the sign information across layers of their GCN model.

Lastly, SLF [39] extends latent-factor models [19]. It performs two types (*i.e.*, positive and negative) of node embeddings (*i.e.*, not a single embedding) for each target node v_c. First, it samples node pairs between v_c and all nodes directly connected to v_c, and optimizes embeddings of the two nodes incident to the sampled node pair based on its sign. Finally, it takes a combination of two node embeddings as the final node embedding.

In summary, most signed NE methods have been successfully applied to signed networks by extending the ideas of unsigned NE methods. However, although the adversarial learning has been verified by GraphGAN to be effective for unsigned NE, none of the above methods employ the adversarial learning for signed NE.

3 PROPOSED APPROACH

3.1 Overview

The signed NE problem is formulated as follows: Let $\mathcal{G}= (V, E^+, E^-)$ be a given signed network, where $V = \{v_1, v_2, \cdots, v_n\}$ represents the set of n nodes and E^+ and E^- represent the sets of positive and negative edges, respectively. Note that $E^+ \cap E^- = \emptyset$, a node pair cannot have both positive and negative edges simultaneously. Signed NE methods aim to learn a function $f : V \longrightarrow \mathbb{R}^k$ which maps each node $v \in V$ to a k-dimensional vector. Table 1 summarizes a list of notations used in this paper.

In this section, we propose a novel signed NE method, ASiNE, based on *adversarial learning*. Figure 1 shows the overview of an ASiNE framework. We first divide a given signed network $\mathcal{G}$ into a subgraph $\mathcal{G}^+$, consisting of V^+ and E^+, and a subgraph $\mathcal{G}^-$, consisting of V^- and E^- (Figure 1-(a)). Then, we train two generators G^+/G^- and two discriminators D^+/D^- using $\mathcal{G}^+/\mathcal{G}^-$: (G^+, D^+) and (G^-, D^-) act as *two pairs of opponents in the four-player minimax game.*

Under the guidance provided by D^+/D^-, G^+/G^- generate *fake positive/negative edges*, respectively, through the biased random walk that considers the proximity by positive and negative edges (Figure 1-(b)). Further, G^- additionally generates fake positive edges, by *considering "an enemy of my enemy is my friend" of the balance theory*. In this generation process, G^+/G^- maximize the likelihood for fake positive/negative edges in their *shared embedding space*, respectively.

As shown in Figure 1-(c), D^+/D^- discriminate *fake* positive/negative edges generated from G^+/G^-, and *real* positive/negative edges sampled directly from $\mathcal{G}^+/\mathcal{G}^-$, respectively. In this discrimination process, D^+/D^- maximize the likelihood for real positive/negative edges, respectively, while minimizing that for fake positive/negative edges in their *shared embedding space*, respectively.

Table 1: Notations used in this paper

Notation	Description
$\mathcal{G}$	Signed network
V, V^+, V^-	Sets of nodes, nodes with at least one positive edge, and nodes with at least one negative edge
E, E^+, E^-	Sets of edges, positive edges, and negative edges
$\mathcal{G}^+, \mathcal{G}^-$	Subgraphs consisting of v and only E^+/E^-
$T^+_{v_c}, T^-_{v_c}$	BFS-trees rooted at v_c in $\mathcal{G}^+/\mathcal{G}^-$
$N^+_{v_c}(v), N^-_{v_c}(v)$	Sets of nodes directly connected to v in $T^+_{v_c}/T^-_{v_c}$
$Path^+_{v_c \to v}, Path^-_{v_c \to v}$	Unique paths from v_c to v in $T^+_{v_c}/T^-_{v_c}$
$D^+(v, v_c; \theta_D)$, $G^+(v\|v_c; \theta_G)$	Discriminator and generator for $\mathcal{G}^+$
$D^-(v, v_c; \theta_D)$, $G^-(v\|v_c; \theta_G)$	Discriminator and generator for $\mathcal{G}^-$
$d_v, g_v \in \mathbb{R}^k$	k-dimensional representation vectors of v in discriminators D^+/D^- and generators G^+/G^-
$\theta_D, \theta_G \in \mathbb{R}^{\|V\| \times k}$	Union of all d_v and g_v
$p^+_{true}(v\|v_c), p^-_{true}(v\|v_c)$	Underlying true positive and negative connectivity distribution of v_c over all other nodes v. $N^+_{v_c}(v_c)/N^-_{v_c}(v_c)$ can be seen as a set of observed nodes sampled from $p^+_{true}(v\|v_c)/p^-_{true}(v\|v_c)$.
$p_{T^+_{v_c}}(v_j\|v_i), p_{T^-_{v_c}}(v_j\|v_i)$	Positive and negative relevance probability of v_j for given v_i in $T^+_{v_c}/T^-_{v_c}$

The intuition of employing the adversarial learning for signed NE is as follows. The competition in this minimax game derives both the generators and the discriminators to help each other to achieve their own goals *until the generators produce plausible fake edges* that can deceive the discriminators eventually. In other words, the nodes in the *shared* embedding space for G^+ and G^- are located in such a way to preserve the proximity in both $\mathcal{G}^+$ and $\mathcal{G}^-$, which is the eventual goal of signed NE. Therefore, we use the vectors of the nodes represented in the *shared* embedding space as their final representation vectors.

3.2 ASiNE framework

Overall objective. Given the subgraphs $\mathcal{G}^+$ and $\mathcal{G}^-$, we aim to learn the following four models (*i.e.*, two generators and two discriminators):

- Generator $G^+(v|v_c; \theta_G)$ for $\mathcal{G}^+$: This model aims to *generate* the most likely nodes to be *positively* connected with a given v_c from V^+ by approximating $p^+_{true}(v|v_c)$.
- Generator $G^-(v|v_c; \theta_G)$ for $\mathcal{G}^-$: This model aims to *generate* the most likely nodes to be *negatively* connected with a given v_c from V^- by approximating $p^-_{true}(v|v_c)$.
- Discriminator $D^+(v, v_c; \theta_D)$ for $\mathcal{G}^+$: This model aims to *discriminate* labels (*i.e.*, real or fake) for the node pairs (v, v_c) by estimating the *positive connectivity* for the pairs.
- Discriminator $D^-(v, v_c; \theta_D)$ for $\mathcal{G}^-$: This model aims to *discriminate* labels (*i.e.*, real or fake) for the node pairs (v, v_c) by estimating the *negative connectivity* for the pairs.

The generators $G^+/G^-$and discriminators D^+/D^- are combined with playing a *minimax game*. In *a shared embedding space*, G^+ embeds two nodes incident to a fake positive edge to be close, and G^- embeds those incident to a fake negative edge to be distant. On the other hand, in *another shared embedding space*, D^+ embeds two nodes incident to a real positive edge to be close and those incident to a fake positive edge to be distant; D^- embeds two nodes incident to a real negative edge to be distant and those incident to a fake negative edge to be close.

Formally, (G^+, D^+) and (G^-, D^-) act as *two pairs of opponents in the following four-player minimax game* with the joint loss function $L(G^+, G^-, D^+, D^-)$:

$$\begin{aligned}\min_{\theta_G}\max_{\theta_D}\ & L(G^+, G^-, D^+, D^-)\\ &= \sum_{v_c \in V^+} \big((\mathbb{E}_{v \sim p^+_{true}(\cdot|v_c)})\left[log D^+(v, v_c; \theta_D)\right]\\ &\qquad + (\mathbb{E}_{v \sim G^+(\cdot|v_c;\theta_G)})\left[log(1 - D^+(v, v_c; \theta_D))\right]\big)\\ &+ \sum_{v_c \in V^-} \big((\mathbb{E}_{v \sim p^-_{true}(\cdot|v_c)})\left[log D^-(v, v_c; \theta_D)\right]\\ &\qquad + (\mathbb{E}_{v \sim G^-(\cdot|v_c;\theta_G)})\left[log(1 - D^-(v, v_c; \theta_D))\right]\big).\end{aligned} \tag{1}$$

Note that, to learn both positive and negative edges, G^+ and G^- update the parameters (*i.e.*, θ_G) in a *shared* embedding space, and D^+ and D^- update the parameters (*i.e.*, θ_D) in another *shared* embedding space.[2] Here, we could design four models, G^+, G^-, D^+, and D^-, in four embedding spaces to update parameters in their own embedding space, individually. Then, we can get the final representation vectors of nodes by combining (*e.g.*, concatenation, average) the two vectors learned by G^+ and G^- for each node. However, we believe that *learning the proximity of both positive and negative edges together in a shared embedding space* helps to consider complex relations involving both positive and negative signs between nodes. We will verify the effectiveness of the embedding space sharing strategy in RQ1 of Section 4.2.

Discriminator optimization. Based on Eq. (1), we need to learn parameters θ_D and θ_G for discriminators and generators, respectively. To do this, we discuss the implementation and optimization of discriminators. Given real and fake positive edges, D^+ aims to maximize the log-probability of correctly classifying these edges. Similarly, given real and fake negative edges, D^- aims to maximize the log-probability of correctly classifying these edges. We can define D^+ and D^- as any functions such as sigmoid and SDNE [35]. Following [16, 32, 36], we present implementations of D^+ and D^- for a given node pair (v, v_c) with the sigmoid function as follows[3]:

$$D^+(v, v_c) = \sigma(d_v^\top d_{v_c}) = \frac{1}{1 + exp(-d_v^\top d_{v_c})}, \tag{2}$$

$$D^-(v, v_c) = 1 - \sigma(d_v^\top d_{v_c}) = 1 - \frac{1}{1 + exp(-d_v^\top d_{v_c})}. \tag{3}$$

Note that $D^+(v, v_c)$ and $D^-(v, v_c)$ indicate the *predicted positive and negative connectivity* for (v, v_c), respectively. Finally, given real edges (*i.e.*, E^+/E^- existing in $\mathcal{G}^+/\mathcal{G}^-$, respectively) and fake edges (*i.e.*, node pairs unconnected in $\mathcal{G}^+/\mathcal{G}^-$) (v, v_c) for D^+ and D^-, respectively, ASiNE updates d_v and d_{v_c} by ascending the gradient with respect to them:

[2]One can be worried that a node pair is generated as a fake *positive* edge by G^+ and as a fake *negative* edge by G^- at the same time. In this case, our sharing strategy makes G^+/G^- embed the two nodes in the pair to be close/distant in the *shared embedding space*. This could be *repeatedly* performed in the training process, causing meaningless learning. However, we observed that such a contradictory case very rarely occurred (*e.g.*, 0.08% among all fake edges in the case of the Bitcoin-Alpha dataset) in fake positive/negative edges generated by G^+/G^-. This is because G^+/G^- have *different* underlying connectivity distributions p^+_{true}/p^-_{true} to generate the fake positive/negative edges.

[3]Although we can define D^+ and D^- with other functions, examining the superiority between these functions is not the scope of this paper.

$$\nabla_{\theta_D} L(G^+, G^-, D^+, D^-) = \begin{cases} \nabla_{\theta_D} log D^+(v, v_c), & \text{if } (v, v_c) \text{ is a real positive edge from } \mathcal{G}^+, \\ \nabla_{\theta_D} (1 - log D^+(v, v_c)), & \text{if } (v, v_c) \text{ is a fake positive edge from } G^+, \\ \nabla_{\theta_D} log D^-(v, v_c), & \text{if } (v, v_c) \text{ is a real negative edge from } \mathcal{G}^-, \\ \nabla_{\theta_D} (1 - log D^-(v, v_c)), & \text{if } (v, v_c) \text{ is a fake negative edge from } G^-. \end{cases} \tag{4}$$

Generator optimization. We discuss the implementation and optimization of generators. Given fake positive edges, G^+ aims to minimize the log-probability that D^+ correctly assigns fake labels to the edges. Similarly, given fake negative edges, G^- aims to minimize the log-probability that D^- correctly assigns fake labels to the edges. For the implementation of G^+, we can employ any functions such as *hierarchical softmax* [30], *negative sampling* [28], and *graph softmax* [36]. In this paper, we present an implementation of G^+ with the *graph softmax* [36] which is known to satisfy useful properties such as *graph-structure-awareness* and *computational efficiency*.[4]

To do the graph softmax, we first make a *breadth-first-search* (BFS) tree $T^+_{v_c}$ rooted at a target node v_c in $\mathcal{G}^+$. Given $T^+_{v_c}$, we find a set $N^+_{v_c}(v_c)$ of the nodes directly connected to v_c. Then, given a node v_c and its neighbor $v_i \in N^+_{v_c}(v_c)$, we define the *positive relevance probability* $p^+_{T^+_{v_c}}(v_i|v_c)$ of v_i for given v_c in $T^+_{v_c}$ as follows:

$$p^+_{T^+_{v_c}}(v_i|v_c) = \frac{exp(g_{v_i}^\top g_{v_c})}{\sum_{v_j \in N^+_{v_c}(v_c)} exp(g_{v_j}^\top g_{v_c})}. \tag{5}$$

which is actually a softmax function over $N^+_{v_c}(v_c)$. Note that $p^+_{T^+_{v_c}}$ can be defined only for the node pairs directly connected in $T^+_{v_c}$. However, we need to generate the node pairs *unconnected* in $T^+_{v_c}$ as *fake positive edges*. To compute the graph softmax $G^+(v|v_c)$ for the unconnected node pairs $(v|v_c)$, we denote a unique path from v_c to v at $T^+_{v_c}$ as $Path^+_{v_c \to v} = (v_{r_0}, v_{r_1}, \cdots, v_{r_m})$ where $v_{r_0} = v_c$ and $v_{r_m} = v$. Based on $Path^+_{v_c \to v}$, we define $G^+(v|v_c)$ as follows [36]:

$$G^+(v|v_c) = \prod_{j=1}^{m} p^+_{T^+_{v_c}}(v_{r_j}|v_{r_{j-1}}). \tag{6}$$

Here, $G^+(v|v_c)$ decreases with the increase of the distance between v and v_c in $\mathcal{G}^+$ [36]. In other words, we can obtain an approximated positive connectivity probability that preserves the proximity in $\mathcal{G}^+$.

Next, for the implementation of G^-, we extend the graph softmax to exploit the *balance theory*. Similar to G^+, we first make a BFS tree $T^-_{v_c}$ rooted at a target node v_c in $\mathcal{G}^-$. Also, we find a set $N^-_{v_c}(v_c)$ of nodes directly connected to v_c in $T^-_{v_c}$. Using $N^-_{v_c}(v_c)$, we define the *positive relevance probability* $p^+_{T^-_{v_c}}(v_i|v_c)$ of v_i for given v_c in $T^-_{v_c}$ as follows:

$$p^+_{T^-_{v_c}}(v_i|v_c) = \frac{exp(g_{v_i}^\top g_{v_c})}{\sum_{v_j \in N^-_{v_c}(v_c)} exp(g_{v_j}^\top g_{v_c})}. \tag{7}$$

Based on $p^+_{T^-_{v_c}}(v_i|v_c)$, we define the *negative relevance probability* $p^-_{T^-_{v_c}}(v_i|v_c)$ of v_i for given v_c in $T^-_{v_c}$ as follows:

$$p^-_{T^-_{v_c}}(v_i|v_c) = \frac{1 - p^+_{T^-_{v_c}}(v_i|v_c)}{\sum_{v_j \in N^-_{v_c}(v_c)}(1 - p^+_{T^-_{v_c}}(v_j|v_c))}. \tag{8}$$

[4]Although we can define G^+ with other functions, examining the superiority between these functions is not the scope of this paper.

As in G^+, we need to generate the node pairs *unconnected* in $T^-_{v_c}$ as *fake negative edges*. To do this, we denote a unique path from v_c to v at $T^-_{v_c}$ as $Path^-_{v_c \to v} = (v_{r_0}, v_{r_1}, \cdots, v_{r_m})$ where $v_{r_0} = v_c$ and $v_{r_m} = v$. Based on $Path^-_{v_c \to v}$, we define $G^-(v|v_c)$ as follows:

$$G^-(v|v_c) = \prod_{j=1}^{m} p^-_{T^-_{v_c}}(v_{rj}|v_{r_{j-1}}). \tag{9}$$

Note that, based on the balance theory, the signs for relationships of unconnected node pairs (v, v_c) in $T^-_{v_c}$ can be determined by *the number of edges* in $Path^-_{v_c \to v}$. More specifically, the sign is given *negative* if there are an *odd number of edges* along $Path^-_{v_c \to v}$ while it is given *positive* if there are an *even number of edges*. Likewise, our extended graph softmax $G^-(v|v_c)$ is designed so that its output can be either a positive connectivity probability (*i.e.*, if m is an even number) or a negative connectivity probability (*i.e.*, if m is an odd number) depending on the number of edges (=m) between v and v_c. In addition, $G^-(v|v_c)$ decreases with the increase of the distance between v and v_c in $\mathcal{G}^-$. In other words, we can obtain an approximated positive or negative connectivity probability that preserves the proximity in $\mathcal{G}^-$. Thus, based on $G^-(v|v_c)$, we can generate fake positive edges as well as fake negative edges. We will explain more specifically how to generate fake edges in Section 3.3.

Finally, given fake positive/negative edges (v, v_c) for G^+/G^-, respectively, we compute the gradient of $L(G^+, G^-, D^+, D^-)$ with respect to θ_G by *policy gradient* [36] as follows:

$$\nabla_{\theta_G} L(G^+, G^-, D^+, D^-) = \begin{cases} \nabla_{\theta_G} log G^+(v|v_c) log(1 - D^+(v, v_c)), \\ \qquad \text{if } (v, v_c) \text{ is a fake positive edge from } G^+, \\ \nabla_{\theta_G} log G^-(v|v_c) log(1 - D^-(v, v_c)), \\ \qquad \text{if } (v, v_c) \text{ is a fake negative edge from } G^-. \end{cases} \tag{10}$$

3.3 Fake edge generation

Now, we need to decide the edges that G^+, G^-, D^+, and D^- will learn based on the above optimization functions. In the case of *real* positive and negative edges, we simply select the node pairs (v, v_c) where v is *directly connected* to a target node v_c in $\mathcal{G}^+$ and $\mathcal{G}^-$, respectively, whereas in the case of *fake* positive and negative edges, we should newly generate the *unconnected* node pairs based on $G^+(v|v_c)$ and $G^-(v|v_c)$, respectively. However, it is a very time-consuming task to calculate $G^+(v|v_c)$ and $G^-(v|v_c)$ of *all* unconnected node pairs and to find the node pairs with the highest probability among them.

Therefore, in this section, we discuss how to efficiently generate fake positive and negative edges. Inspired by GraphGAN [36], we propose a *random-walk-based generation strategy* that considers the balance theory as well as the positive and negative relevance probabilities of node pairs.

Generation of fake positive edges from $\mathcal{G}^+$. First, the steps to generate fake positive edges in $\mathcal{G}^+$ are as follows:

- Given a target node v_c, we perform a biased random walk that starts from v_c at $T^+_{v_c}$ and considers the *positive relevance probability* defined in Eq. (5).
- The random walk repeats until the walker *revisits* the same node that the walker has visited before.

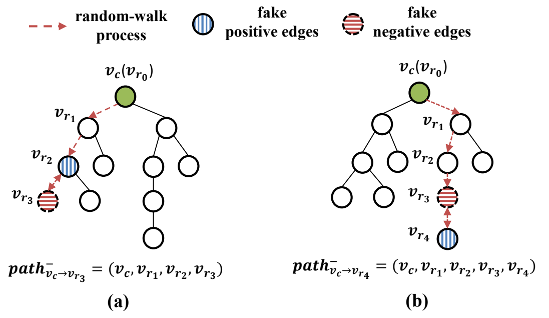
random-walk process
fake positive edges
fake negative edges
$v_c(v_{r_0})$
v_{r_1}
v_{r_2}
v_{r_3}
v_{r_4}
$path^-_{v_c \to v_{r_3}} = (v_c, v_{r_1}, v_{r_2}, v_{r_3})$
$path^-_{v_c \to v_{r_4}} = (v_c, v_{r_1}, v_{r_2}, v_{r_3}, v_{r_4})$
(a)
(b)

Figure 2: Generation of fake edges from $\mathcal{G}^-$.

- We define a random-walk path $Path^+_{v_c \to v}$ from v_c to the node v (the previous node of the revisited node) and select the node pair (v_c, v) for v_c and the last node v of $Path^+_{v_c \to v}$ for a *fake positive edge.*

For example, suppose there is a random-walk path $Path^+_{v_c \to v} = (v_{r_0}, v_{r_1}, v_{r_2}, v_{r_3})$ where $v_{r_0} = v_c$ and $v_{r_3} = v$ that satisfies the above termination condition. In $Path^+_{v_c \to v}$, G^+ generates a node pair (v_{r_0}, v_{r_3}) as a fake positive edge. Note that G^+ can generate fake positive edges only since $\mathcal{G}^+$ only has positive edges inside.[5]

Generation of fake negative edges from $\mathcal{G}^-$. The steps to generate fake negative edges in $\mathcal{G}^-$ are as follows:

- Given a target node v_c, we perform a biased random walk that starts from v_c at $T^-_{v_c}$ and considers the *negative relevance probability* defined in Eq. (8).
- The random walk repeats until the walker *revisits* the same node that the walker has visited before.
- We define a random-walk path $Path^-_{v_c \to v}$ from v_c to the node v (the previous node of the revisited node) and select a *fake negative edge* in $Path^-_{v_c \to v}$ based on the following two aspects of the balance theory:
  - If there are an odd number of edges in $Path^-_{v_c \to v}$, a node pair (v, v_c) for v_c and the last node v is selected as a fake negative edge (Figure 2-(a)).
  - If there are an even number of edges in $Path^-_{v_c \to v}$, a node pair (v, v_c) for v_c and the second last node v is selected as a fake negative edge (Figure 2-(b)).

Here, we discuss the termination condition of our random-walk-based generation strategy. Unlike existing random-walk-based signed NE methods [18, 40] that generate paths of a *fixed* length, our strategy can generate paths of *variable* lengths according to the termination condition. To address this issue, we examine the distribution of the path lengths generated by our strategy. Figure 3 shows the results from the Bitcoin-Alpha and WikiRfA datasets.[6] The x-axis represents the path length and the y-axis does the ratio of the paths having a length. We see that most paths (*i.e.*, 97% for Bitcoin-Alpha and 99% for WikiRfA) have the lengths less than 6. The results indicate that the paths generated by our strategy have the lengths that are short and are similar to one another. Furthermore, we tried to generate the paths of a fixed length as in [18, 40]. However, in spite

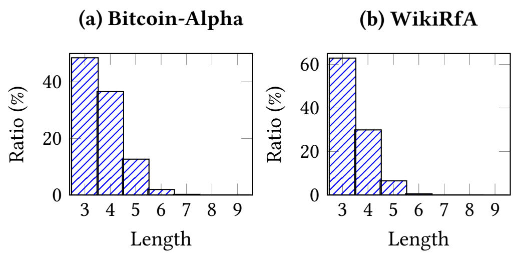

Figure 3: Distribution of the path lengths.

of extensive experiments, we could not obtain any positive results verifying that this generation strategy contributes to improve the accuracy of ASiNE.

Generation of fake positive edges from $\mathcal{G}^-$. We have presented how to generate fake positive/negative edges in $\mathcal{G}^+/\mathcal{G}^-$, respectively. In this section, we propose a strategy of generating fake positive edges from $\mathcal{G}^-$ by using "an enemy of my enemy is my friend" of the balance theory. To do this, in $Path^-_{v_c \to v}$, we select a *fake positive edge* according to the following two aspects of the balance theory:

- If there are an odd number of edges in $Path^-_{v_c \to v}$, a node pair (v, v_c) for v_c and the second last node v is selected as a fake positive edge (Figure 2-(a)).
- If there are an even number of edges in $Path^-_{v_c \to v}$, a node pair (v, v_c) for v_c and the last node v is selected as a fake positive edge (Figure 2-(b)).

The fake positive edges generated by G^- enable us fully learn rich information of $\mathcal{G}^-$. To update the parameters θ_D and θ_G for the node pairs (v, v_c) incident to the fake positive edges, we compute the gradients for them as follows. If (v, v_c) is a fake positive edge from G^-:

$$\nabla_{\theta_D} L(G^+, G^-, D^+, D^-) = \nabla_{\theta_D}(1 - logD^+(v, v_c)). \tag{11}$$

$$\nabla_{\theta_G} L(G^+, G^-, D^+, D^-) = \nabla_{\theta_G} logG^-(v|v_c) log(1 - D^+(v, v_c)). \tag{12}$$

There are an even number of negative edges in $Path^-_{v_c \to v}$ between the two nodes (v, v_c) incident to the fake positive edge. Therefore, we update the parameters θ_G based on $G^-(v, v_c)$ (*i.e.*, rather than $1 - G^-(v, v_c)$) because $G^-(v, v_c)$ already indicates a positive connectivity probability by Eq. (9). Here, we note that $G^+(v, v_c)$ cannot be used because it can be defined only when a path between v_c and v exists in $T^+_{v_c}$. Since we generate the fake positive edge (v, v_c) from $T^-_{v_c}$, there may not be a path between them in $T^+_{v_c}$. In RQ3 of Section 4.2, we will validate that the additional generation of fake positive edges from $\mathcal{G}^-$ helps to improve the performance of the signed NE.

Finally, Algorithm 1 shows the overall process of our ASiNE framework. We summarize a single adversarial training procedure for target node v_c in the ASiNE framework as follows: (1) fake edge generation: ASiNE generates *two* fake positive edges (from $\mathcal{G}^+$ and $\mathcal{G}^-$) and *a* fake negative edge (from $\mathcal{G}^-$) by performing a single random walk process in $T^+_{v_c}$ and $T^-_{v_c}$, respectively[7]; (2) discriminator

[5]To generate a fake negative edge, the balance theory requires negative edges in the path (*i.e.*, "a friend of my enemy is my enemy," "an enemy of my friend is my enemy").

[6]Due to space limitations, we omit the results for other datasets, which showed a tendency similar to those in Figure 3.

[7]Since the number of positive edges is greater than that of negative edges in real-life signed networks, we generate more fake positive edges than fake negative edges. We also conducted additional experiments in a setting that generates equal numbers of fake positive and negative edges; however, there was no difference in accuracies between this setting and our original setting. To reduce the training time of ASiNE without sacrificing the accuracy, we recommend to set the ratio of fake positive and negative edges as 2:1.

Algorithm 1 ASiNE framework

Input: embedding dimensionality k, size of generating edges g, size of discriminating edges d, number of epochs e, number of iterations for generators and discriminators per epoch i, window size w

Output: $G^+(v|v_c;\theta_G)$, $G^-(v|v_c;\theta_G)$, $D^+(v, v_c;\theta_D)$, $D^-(v, v_c;\theta_D)$

```
1:  Initialize $\theta_D$ and $\theta_G$
2:  Divide $\mathcal{G}$ into $\mathcal{G}^+$ and $\mathcal{G}^-$
3:  Construct BFS trees $T^+_{v_c}$ and $T^-_{v_c}$ for all $v_c \in V$
4:  for $epoch \leftarrow 1$ to $e$ do
5:      for $v_c \in V$ do
6:          for $iter \leftarrow 1$ to $i$ do                                    ▷ $D^+$-step
7:              Sample $d$ real positive edges from $\mathcal{G}^+$
8:              Generate $d$ fake positive edges from $G^+(v|v_c;\theta_G)$
9:              Update $\theta_D$ via Eqs. (2), (3), and (4)
10:         end for
11:         for $iter \leftarrow 1$ to $i$ do                                    ▷ $G^+$-step
12:             Generate $g$ fake positive edges from $G^+(v|v_c;\theta_G)$
13:             Sample all node pairs exsiting within $w$ in paths
14:             Update $\theta_G$ via Eqs. (5), (6), and (10)
15:         end for
16:         for $iter \leftarrow 1$ to $i$ do                                    ▷ $D^-$-step
17:             Sample $d$ real negative and $d$ real positive edges from $\mathcal{G}^-$ and $\mathcal{G}^+$
18:             Generate $d$ fake negative and $d$ fake positive edges from $G^-(v|v_c;\theta_G)$
19:             Update $\theta_D$ via Eqs. (3) and (4) (i.e., negative edges), and Eqs. (2) and (11)
                (i.e., positive edges)
20:         end for
21:         for $iter \leftarrow 1$ to $i$ do                                    ▷ $G^-$-step
22:             Generate $g$ fake negative and $g$ fake positive edges from $G^-(v|v_c;\theta_G)$
23:             Sample all node pairs exsiting within $w$ in paths
24:             Update $\theta_G$ via Eqs. (8), (9), (10) (i.e., negative edges), and (12) (i.e.,
                positive edges)
25:         end for
26:     end for
27: end for
```

optimization: ASiNE provides fake and real edges to D^+ and D^- to classify them. Then, it updates their parameters θ_D based on the classification results; (3) generator optimization: ASiNE updates the parameters θ_G for node pairs corresponding to fake positive and negative edges. In addition, it updates the parameters θ_G for all node pairs existing within a window of size w in the random-walk paths. Apparently, we update θ_G for the node pairs existing within w in the paths obtained from $\mathcal{G}^-$ by considering their sign.

3.4 Discussions

We discuss the characteristics of ASiNE in comparisons with GraphGAN [36]. ASiNE is based on the excellent philosophy of GraphGAN. However, we would like to highlight that ASiNE is not just a straightforward extension of GraphGAN but incorporates three of our novel ideas that allow the *adversarial mechanism* to effectively leverage the properties of signed networks: (1) extension of graph softmax; (2) fake edge generation; (3) embedding space sharing. In this paper, we carefully incorporated them to ASiNE, thereby providing more-accurate embeddings that preserve the structural characteristics of signed networks.

More specifically, the extension of graph softmax assigns a positive/negative connectivity probability for a pair of nodes in $\mathcal{G}^-$ if there are an even/odd number of edges between them, making the balance theory satisfied. Also, the probability decreases with the increase of the distance between the two nodes, which helps to preserve the proximity in $\mathcal{G}^-$. Thanks to the idea of the extension of graph softmax, our fake edge generation idea generates both fake positive and negative edges in $\mathcal{G}^-$ satisfying two conditions: (a) their edge sign is determined by the balance theory; (b) they have a high absolute value of positive/negative connectivity probability. As a result, we can generate both fake negative and positive edges in a single random-walk-path with G^- alone (*i.e.*, not requiring two separate generators). Lastly, the embedding space sharing idea considers effectively various combinations of positive and negative signs on the path between nodes. This idea is more effective than the ideas of aggregating (separate) positive and negative embeddings, widely used in existing NE methods. In Section 4, we demonstrate that each of our ideas is fairly effective and our final ASiNE integrating all ideas is the most effective.

Table 2: Summary of the datasets used in the experiments

Datasets	Nodes	Edges	Positive Edges	Negative Edges
Bitcoin-Alpha	3,784	14,145	12,729 (89.9%)	1,416 (10.1%)
Bitcoin-OTC	5,901	21,522	18,390 (85.4%)	3,132 (14.6%)
Slashdot	13,182	36,338	30,914 (85.1%)	5,424 (14.9%)
WikiRfA	11,258	185,629	144,451 (77.8%)	41,178 (22.2%)
Epinions	25,148	105,061	74,060 (70.5%)	31,001 (29.5%)

4 EVALUATION

In this section, we validate the effectiveness of our approach via extensive experiments. We designed our experiments, aiming at answering the following key questions:

- RQ1: Does *sharing* a single embedding space by generators (G^+, G^-) and discriminators (D^+, D^-) help to signed NE?
- RQ2: Does our *adversarial learning* with edge signs help to signed NE?
- RQ3: Do *fake positive edges generated from* $\mathcal{G}^-$ help to signed NE?
- RQ4: Does *our ASiNE framework* provide more-effective representation vectors than state-of-the-art signed NE methods?
- RQ5: How do *different values of parameters* influence the accuracy of ASiNE?
- RQ6: Is the training of ASiNE *scalable*?

4.1 Experimental Settings

Datasets. In this paper, we used five real-life signed network datasets *without any preprocessing*: (1) Bitcoin-Alpha, Bitcoin-OTC, and WikiRfA in SNAP (https://snap.stanford.edu/data); (2) Slashdot and Epinions in AMiner (https://www.aminer.cn/data-sna). We regard all the networks as undirected networks. Table 2 shows the detailed statistics of the five datasets.

- **Bitcoin-Alpha** and **Bitcoin-OTC** are trust networks between bitcoin users designed to prevent transactions with fraudulent and risky users. These networks include trust (*i.e.*, positive) and distrust (*i.e.*, negative) edges between users.
- **Slashdot** is a friend network between users on a technology news site. This network contains friend (*i.e.*, positive) and enemy (*i.e.*, negative) edges between users.
- **WikiRfA** is a voting network for electing managers in Wikipedia. This network contains users' supporting vote (*i.e.*, positive) and opposing vote (*i.e.*, negative) edges.
- **Epinions** is a trust network between users on a product review site. This network includes trust (*i.e.*, positive) and distrust (*i.e.*, negative) edges between users based on their reviews.

Competing methods. We compare ASiNE with four state-of-the-art signed NE methods:

- **SNE** [40]: This method utilizes a *log-bilinear* model [29] with random walk sampling. However, it does not consider the balance theory for signed networks.
- **SIDE** [18]: This method utilizes a *skip-gram* model [27] with random walk sampling. It aggregates signs and directions along the path according to the balance theory. In addition, we employ a variant of SIDE, denoted as $\mathbf{SIDE}_{opt}$, using the optimization techniques proposed in [18]. It subsamples high-degree nodes and deletes one-degree nodes.
- **SGCN** [9]: This method utilizes a GCN model [12]. It aggregates and propagates the information across the layers of the signed GCN model.
- **SLF** [39]: This method utilizes a *latent-factor* model [19]. It exploits two types (*i.e.*, positive and negative) of latent factors for each node and combines the two factors to represent the node.

For evaluation, we used the source codes provided by the authors [9, 18, 39, 40]. For a fair comparison, we set node embeddings for all methods to the same dimensionality, 128, following [18]. For other parameters for each method, we used the best settings found via extensive grid search in the following ranges suggested in their respective papers: learning rate $\in \{0.01, 0.025, 0.001, 0.0025\}$; window size $\in \{1, 2, 3, 4, 5\}$ (for SNE, SIDE); maximum length of random walk path $\in \{10, 20, 30, 40, 50\}$ (for SNE, SIDE); number of layers $\in \{1, 2\}$ (for SGCN); sample size of the null relationships $\in \{10, 20, 50, 100\}$ (for SLF); $p_0 \in \{0.001, 0.01, 0.1\}$ (for SLF). The best setting found in ASiNE is as follows: learning rate = 0.01, $e = 20$, $g = 20$, $d = 20$, $i = 10$, $w = 2$.

Evaluation metrics. Following [9, 18, 39, 40], we employ an *edge sign prediction* task to evaluate the accuracy of ASiNE and the competing methods: we first output the embedding vectors of all nodes in the training set by using each method; then, we train a logistic regression classifier with the embedding vectors of each method as features; finally, we predict the edge signs in the test set based on the learned classifier for each method. We note that the proportion of positive and negative edges is unbalanced in most signed networks. To address the class imbalance problem, we use the following two metrics popularly employed in other signed NE research [9, 18, 39, 40]: *area under curve* (AUC) and F1-micro. The final accuracy of each metric was determined by averaging the five accuracies obtained from five-fold cross-validation. Due to space limitations, we omit some experimental results in this paper. The details for *all* experiments are available at https://bit.ly/2YYxIMP.

4.2 Results

RQ1: Effectiveness of shared embedding spaces. Note that, to learn both positive and negative edges, we let G^+ and G^- share one embedding space and D^+ and D^- share another embedding space. In order to verify the effectiveness of this *sharing strategy*, we made variants of ASiNE that do not employ the sharing strategy: they update the parameters in each of four independent embedding spaces by G^+, G^-, D^+, and D^- and get the final embedding space by combining two embeddings learned by G^+ and G^-. For combining the two embeddings, we employed the following four widely used operators [39, 40]: L1_weight (L1): $g_v = |g_v^+ - g_v^-|$, L2_weight (L2): $g_v = |g_v^+ - g_v^-|^2$, Average (Avg): $g_v = \frac{1}{2}(g_v^+ + g_v^-)$, and Concatenation

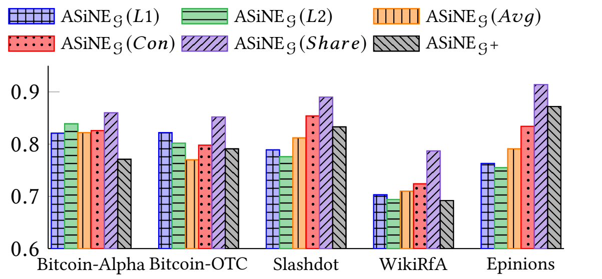

Figure 4: Comparison between the ASiNE methods with/without the embedding space sharing strategy.

(Con): $g_v = g_v^+ \oplus g_v^-$. We denote the four variants of ASiNE according to the operators as ASiNE$_\mathcal{G}(L1)$, ASiNE$_\mathcal{G}(L2)$, ASiNE$_\mathcal{G}(Avg)$, and ASiNE$_\mathcal{G}(Con)$. For comparison, we denote a variant using our sharing strategy as ASiNE$_\mathcal{G}(Share)$.

Figure 4 shows the accuracy of the variants of ASiNE in terms of the AUC. The x-axis represents each method in each dataset, and the y-axis does the AUC. First, among four variants that do not share the embedding spaces, we found that ASiNE$_\mathcal{G}(L1)$ and ASiNE$_\mathcal{G}(L2)$ showed the best accuracy in the small datasets (*i.e.*, Bitcoin-Alpha, Bitcoin-OTC), whereas ASiNE$_\mathcal{G}(Con)$ showed the best accuracy in the large datasets (*i.e.*, Slashdot, WikiRfA, Epinions). However, we observed that ASiNE$_\mathcal{G}(Share)$ *consistently* outperforms *all* other variants in *all* datasets. Specifically, ASiNE$_\mathcal{G}(Share)$ improves the AUC of ASiNE$_\mathcal{G}(Con)$ by 4.09%, 6.78%, 4.21%, 8.69%, and 9.64% for Bitcoin-Alpha, Bitcoin-OTC, Slashdot, WikiRfA, and Epinions, respectively. This result indicates that learning the proximity of both positive and negative edges in a *shared* embedding space helps to consider complex relations of both positive and negative signs between nodes.

RQ2: Effectiveness of adversarial learning with edge signs. Next, we verify whether our adversarial learning with edge signs helps in signed NE. To do this, we compare ASiNE$_\mathcal{G}(Share)$ and a variant, denoted as ASiNE$_{\mathcal{G}^+}$, of ASiNE that uses only the positive subgraph $\mathcal{G}^+$. Note that ASiNE$_{\mathcal{G}^+}$ is equivalent to GraphGAN [36].

Figure 4 shows the accuracy of ASiNE$_\mathcal{G}(Share)$ and ASiNE$_{\mathcal{G}^+}$. We found that ASiNE$_\mathcal{G}(Share)$ *significantly* outperforms ASiNE$_{\mathcal{G}^+}$ in all datasets. Specifically, ASiNE$_\mathcal{G}(Share)$ improves the AUC of ASiNE$_{\mathcal{G}^+}$ by 11.05%, 7.69%, 6.88%, 13.83%, and 4.86% for Bitcoin-Alpha, Bitcoin-OTC, Slashdot, WikiRfA, and Epinions, respectively. This result shows that our adversarial learning with edge signs is more effective in representing the signed network compared to the one without them. In addition, the result validates our objective functions designed to consider edge signs.

RQ3: Effectiveness of fake positive edges from $\mathcal{G}^-$. As mentioned in Section 3.3, we generate fake positive and negative edges from $\mathcal{G}^+$ and $\mathcal{G}^-$, respectively. Furthermore, we additionally generate fake positive edges from $\mathcal{G}^-$ by exploiting "an enemy of my enemy is my friend" of the balance theory. To verify the effectiveness of this strategy of fake positive edge generation, we compare ASiNE$_\mathcal{G}(Share)$ and a variant, denoted as ASiNE$_\mathcal{G}(Share + FP_{\mathcal{G}^-})$, of ASiNE that performs the additional learning of the fake positive edges generated from $\mathcal{G}^-$.

Table 3 shows the accuracies of ASiNE$_\mathcal{G}(Share)$ and ASiNE$_\mathcal{G}(Share + FP_{\mathcal{G}^-})$ in terms of the AUC and F1-micro. We found that

Table 3: Comparison between the ASiNE methods with/without fake positive edges generated from $\mathcal{G}^-$

Methods	ASiNE$_\mathcal{G}$(*Share*)		ASiNE$_\mathcal{G}$(*Share* + *FP*$_{\mathcal{G}^-}$)	
Metrics	AUC	F1-micro	AUC	F1-micro
Bitcoin-Alpha	0.860	0.931	0.866	0.935
Bitcoin-OTC	0.853	0.908	0.869	0.908
Slashdot	0.890	0.885	0.893	0.883
WikiRfA	0.788	0.792	0.806	0.800
Epinions	0.915	0.869	0.918	0.876

Table 4: Comparison of five competing methods and ASiNE

		SNE	SIDE	SIDE$_{opt}$	SGCN	SLF	ASiNE
Bitcoin -Alpha	**AUC**	0.822	0.744	0.747	0.783	*0.844*	**0.866**
	F1-micro	0.918	*0.919*	0.917	0.851	0.915	**0.935**
Bitcoin -OTC	**AUC**	0.816	0.835	0.841	0.798	*0.860*	**0.869**
	F1-micro	0.878	*0.896*	0.894	0.839	0.895	**0.908**
Slashdot	**AUC**	0.853	0.842	0.671	0.767	*0.866*	**0.893**
	F1-micro	0.868	0.841	0.861	0.851	*0.879*	**0.883**
WikiRfA	**AUC**	0.611	0.740	0.737	0.620	*0.790*	**0.806**
	F1-micro	0.772	0.778	0.790	0.543	*0.797*	**0.800**
Epinions	**AUC**	0.758	0.867	0.765	-	*0.896*	**0.918**
	F1-micro	0.771	0.818	0.797	-	*0.852*	**0.876**

-: out-of-time

ASiNE$_\mathcal{G}$(*Share* + *FP*$_{\mathcal{G}^-}$) *universally* outperforms ASiNE$_\mathcal{G}$(*Share*) in all datasets and with all metrics. This result shows that additional learning of fake positive edges generated from $\mathcal{G}^-$ is helpful for accurate signed NE. Thus, we will use ASiNE$_\mathcal{G}$(*Share* + *FP*$_{\mathcal{G}^-}$) as our final approach in the following experiments, simply denoting ASiNE$_\mathcal{G}$(*Share* + *FP*$_{\mathcal{G}^-}$) as ASiNE.

RQ4: Comparison with state-of-the-art signed NE methods. We conducted comparative experiments to show greater accuracy of ASiNE than that of the following five competing methods: SNE [40], SIDE [18], SIDE$_{opt}$ [18], SGCN [9], and SLF [39]. Table 4 illustrates the result.[8] The value in boldface indicates the *best accuracy in each row*, and the value in italic does the *accuracy obtained from the best performer among "all competing methods"*.

We summarize the results shown in Table 4 as follows. First, the best performer among the competing methods depends on the datasets and the metrics: SLF (AUC) and SIDE (F1-micro) for Bitcoin-Alpha and Bitcoin-OTC; SLF for Slashdot, WikiRfA, and Epinions. Second, our ASiNE *consistently and significantly* outperforms all competing methods in all datasets and with all metrics. More specifically, our ASiNE improves the accuracies of SIDE/SLF, the best performers among the competing methods, up to 16.37%/3.18% in terms of AUC, respectively.

The experimental results can be summarized as follows: (1) our *embedding space sharing* strategy enables to represent nodes in the embedding space accurately, in such a way to preserve the proximity by both positive and negative edges; (2) our *fake edge generation* strategy based on the balance theory helps to provide more-sophisticated node embeddings by learning rich relationships between nodes; (3) our *adversarial learning* with edge signs enables to learn accurately the underlying positive/negative connectivity distribution of a signed network under the guidance provided by the discriminators.

[8]Note that, in Epinions, the accuracies of SGCN could not be obtained; SGCN has not finished its training in a week.

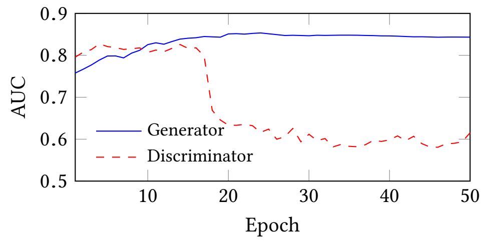

Figure 5: Accuracy changes according to the number of epochs.

RQ5: Parameter analysis for ASiNE.[9] To understand the learning stability of ASiNE, we analyze carefully the changes of accuracy according to different epochs. Figure 5 shows the changes of the accuracy by the generator and the discriminator of ASiNE over the increasing number of epochs. Note that two generators share one embedding space, and two discriminators share another embedding space; thus, two generators (resp. two discriminators) have only one accuracy. The x-axis represents the number of epochs, and the y-axis does the AUC of our generator and discriminator in each epoch in Bitcoin-Alpha.

In Figure 5, the minimax game in our ASiNE shows two trends: (1) the accuracy of the generator steadily increases up to a certain epoch (=20 for Bitcoin-Alpha) and then reaches equilibrium; (2) the accuracy of the discriminator increases only up to a certain epoch (=5 for Bitcoin-Alpha) and then gradually decreases. This result indicates that, after a certain epoch, our generator produces *realistic fake edges that can deceive discriminator*, thereby degrading the accuracy of the discriminator. We note that this trend coincides with those reported in [36].

RQ6: Scalability analysis for ASiNE. To test the scalablity of ASiNE, we examine the changes of its execution time as the number of edges increases. Towards this end, we measured the execution time of ASiNE by adding duplicate edges to WikiRfA.[10] The experiments were conducted in Windows 10 running on Intel Core i9 processor (3.60 GHZ) with 64GB RAM. The result shows that the execution time was around 60, 135, 210, 285, 360 minutes when we have about 0.2M (185,629), 0.4M (371,258), 0.6M (556,887), 0.8M (742,516), 1M (928,145) edges. This result indicates that ASiNE has a *linear scalability* with the increasing number of edges.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we designed a novel signed NE framework, ASiNE, based on the *adversarial learning*. Under our ASiNE framework, we proposed three ideas for effective signed NE: (1) (G^+, D^+) and (G^-, D^-) play two pairs of minimax games that consider edge signs; (2) (G^+, G^-) shares one embedding space for generation, and (D^+, D^-) shares another embedding space for discrimination to learn positive and negative edges together; (3) G^- generates not only fake negative edges but also fake positive edges based on the balance

[9]Due to space limitations, we omit the results in other parameters (*i.e.*, dimensionality of embedding, size of generating and discriminating edges, window size); please, refer to https://bit.ly/2YYxIMP.

[10]WikiRfA has the largest one among the five datasets used in this paper.

theory. Through comprehensive experiments using five real-life signed networks, we demonstrated that each of our ideas for signed NE is effective and our final ASiNE framework integrating all ideas is the most effective in achieving the high accuracy of edge sign prediction tasks. In addition, our experimental results showed that ASiNE consistently and significantly outperforms all the state-of-the-art signed NE methods in all datasets and with all metrics.

Our ASiNE framework is the first attempt that employs adversarial learning for signed NE, suggesting a promising direction for signed NE. As future work, we plan to extend our ASiNE framework by integrating the recent state-of-the-art GANs (*e.g.*, Wasserstein GAN [1], BEGAN [2]). In addition, we note that the directions and weights of edges provide additional information, useful in understanding real-life networks [11, 18, 34]. Thus, we plan to design a new strategy to consider them together on top of our ASiNE framework.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2020R1A2B5B03001960).

REFERENCES

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR* abs/1701.07875 (2017). http://arxiv.org/abs/1701.07875

[2] David Berthelot, Tom Schumm, and Luke Metz. 2017. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *CoRR* abs/1703.10717 (2017). http://arxiv.org/abs/1703.10717

[3] Dorwin Cartwright and Frank Harary. 1956. Structural balance: a generalization of Heider's theory. *Psychological review* 63, 5 (1956), 277.

[4] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. 2019. Rating Augmentation with Generative Adversarial Networks towards Accurate Collaborative Filtering. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 2616–2622.

[5] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks. In *Proc. of the ACM Int'l Conf. on Information and Knowledge Management (ACM CIKM).* 137–146.

[6] Kyung-Jae Cho, Yeon-Chang Lee, Kyungsik Han, Jaeho Choi, and Sang-Wook Kim. 2019. No, That's Not My Feedback: TV Show Recommendation Using Watchable Interval. In *Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE).* 316–327.

[7] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2019. A Survey on Network Embedding. *IEEE Trans. Knowl. Data Eng.* 31, 5 (2019), 833–852.

[8] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. 2015. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Proc. of the Annual Conf. on Neural Information Processing Systems (NIPS).* 1486–1494.

[9] Tyler Derr, Yao Ma, and Jiliang Tang. 2018. Signed Graph Convolutional Networks. In *Proc. of the IEEE Int'l Conf. on Data Mining (IEEE ICDM).* 929–934.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proc. of the Annual Conf. on Neural Information Processing Systems (NIPS).* 2672–2680.

[11] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proc. of the ACM Int'l Conf. on Knowledge Discovery and Data Mining (ACM KDD).* 855–864.

[12] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. of the Annual Conf. on Neural Information Processing Systems (NIPS).* 1024–1034.

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

[14] Won-Seok Hwang, Juan Parc, Sang-Wook Kim, Jongwuk Lee, and Dongwon Lee. 2016. "Told You I didn't Like It": Exploiting uninteresting items for effective collaborative filtering. In *Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE).* 349–360.

[15] Min-Hee Jang, Christos Faloutsos, Sang-Wook Kim, U Kang, and Jiwoon Ha. 2016. PIN-TRUST: Fast Trust Propagation Exploiting Positive, Implicit, and Negative Information. In *Proc. of the ACM Int'l Conf. on Information and Knowledge Management (CIKM).* 629–638.

[16] Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. 2019. CommunityGAN: Community Detection with Generative Adversarial Nets. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 784–794.

[17] Yong-Yeon Jo, Myung-Hwan Jang, Sang-Wook Kim, and Sunju Park. 2019. RealGraph: A Graph Engine Leveraging the Power-Law Distribution of Real-World Graphs. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 807–817.

[18] Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U. Kang. 2018. SIDE: Representation Learning in Signed Directed Networks. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 509–518.

[19] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[20] Jérôme Kunegis, Julia Preusse, and Felix Schwagereit. 2013. What is the added value of negative links in online social networks?. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 727–736.

[21] Yeon-Chang Lee, Sang-Wook Kim, and Dongwon Lee. 2018. gOCCF: Graph-Theoretic One-Class Collaborative Filtering Based on Uninteresting Items. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI).* 3448–3456.

[22] Youngnam Lee, Sang-Wook Kim, Sunju Park, and Xing Xie. 2018. How to Impute Missing Ratings?: Claims, Solution, and Its Application to Collaborative Filtering. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 783–792.

[23] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. 2010. Predicting positive and negative links in online social networks. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 641–650.

[24] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. 2010. Signed networks in social media. In *Proc. of the Int'l Conf. on Human Factors in Computing Systems (CHI).* 1361–1370.

[25] David Liben-Nowell and Jon M. Kleinberg. 2003. The link prediction problem for social networks. In *Proc. of the ACM Int'l Conf. on Information and Knowledge Management (ACM CIKM).* 556–559.

[26] Shuqi Lu, Zhicheng Dou, Xu Jun, Jian-Yun Nie, and Ji-Rong Wen. 2019. PSGAN: A Minimax Game for Personalized Search with Limited and Noisy Click Data. In *Proc. of the ACM Int'l Conf. on Research & Development in Information Retrieval (ACM SIGIR).* 555–564.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. of the Int'l Conf. on Learning Representations (ICLR).*

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proc. of the Annual Conf. on Neural Information Processing Systems (NIPS).* 3111–3119.

[29] Andriy Mnih and Geoffrey E. Hinton. 2008. A Scalable Hierarchical Distributed Language Model. In *Proc. of the Annual Conf. on Neural Information Processing Systems (NIPS).* 1081–1088.

[30] Frederic Morin and Yoshua Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model. In *Proc. of the Int'l Workshop on Artificial Intelligence and Statistics (AISTATS).*

[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proc. of the ACM Int'l Conf. on Knowledge Discovery and Data Mining (ACM KDD).* 701–710.

[32] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proc. of the Int'l Conf. on Learning Representations (ICLR).*

[33] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29, 3 (2008), 93–106.

[34] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proc. of the Int'l Conf. on World Wide Web (WWW).* 1067–1077.

[35] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proc. of the ACM Int'l Conf. on Knowledge Discovery and Data Mining (ACM KDD).* 1225–1234.

[36] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning With Generative Adversarial Nets. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI).* 2508–2515.

[37] Suhang Wang, Jiliang Tang, Charu C. Aggarwal, Yi Chang, and Huan Liu. 2017. Signed Network Embedding in Social Media. In *Proc. of the SIAM Int'l Conf. on Data Mining (SDM).* 327–335.

[38] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI).* 203–209.

[39] Pinghua Xu, Wenbin Hu, Jia Wu, and Bo Du. 2019. Link Prediction with Signed Latent Factors in Signed Social Networks. In *Proc. of the ACM Int'l Conf. on Knowledge Discovery and Data Mining (ACM KDD).* 1046–1054.

[40] Shuhan Yuan, Xintao Wu, and Yang Xiang. 2017. SNE: Signed Network Embedding. In *Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD).* 183–195.