
LATENT NETWORK EMBEDDING VIA ADVERSARIAL AUTO-ENCODERS

Minglong Lei
Beijing University of Technology
Beijing, 100124, China

Yong Shi
University of Chinese Academy of Sciences
Beijing, 100190, China

Lingfeng Niu*
University of Chinese Academy of Sciences
Beijing, 100190, China

ABSTRACT

Graph auto-encoders have proved to be useful in network embedding task. However, current models only consider explicit structures and fail to explore the informative latent structures cohered in networks. To address this issue, we propose a latent network embedding model based on adversarial graph auto-encoders. Under this framework, the problem of discovering latent structures is formulated as inferring the latent ties from partial observations. A latent transmission matrix that describes the strengths of existing edges and latent ties is derived based on influence cascades sampled by simulating diffusion processes over networks. Besides, since the inference process may bring extra noises, we introduce an adversarial training that works as regularization to dislodge noises and improve the model robustness. Extensive experiments on link prediction and node classification tasks show that the proposed model achieves superior results compared with baseline models.

Keywords Graph Auto-encoders · Diffusion Process · Network Inference · Adversarial Regularization

1 Introduction

Learning network representations via deep neural networks achieves promising results in recent years [1, 2]. Compared with shallow network embedding models (i.e., [3, 4]), deep network embedding models exhibit superior representation ability and can extract high-level structure features. The learned embeddings can be used for downstream network analysis tasks such as node classification [5, 6] and link prediction [2, 7].

Graph auto-encoders are a series of deep neural networks which apply an auto-encoder paradigm [8, 9] to network embedding. Such framework benefits from its flexibility since the encoders and decoders can be replaced with various deep models (e.g., MLP [2] and LSTM [10]) to accomplish certain tasks. Generally, the encoder aims at mapping the global or local structure information to a low-dimensional space while the decoder serves as a generator that recovers information to approximate the input. Inspired by the encoder-decoder paradigm, substantial works devote efforts to developing diverse variants of encoders and decoders. For example, DNGR [11] builds a positive pointwise mutual information (PPMI) matrix and proposes using stacked denoising auto-encoders to learn node embeddings. NetRA [10] first samples node sequences by random walks and then uses LSTM as the encoder and decoder. Above examples demonstrate that the graph auto-encoders can be designed to handle either global structures (e.g., global adjacent matrix) or local structures (e.g., short random walk sequences).

Although current graph auto-encoder models can extract useful structural patterns in networks, they assume that networks are static and all network data can be readily fed into the learning models. However, the assumption is often violated since the real network structures are not always observable and are required to be discovered [12, 13]. Specifically, there exist *latent ties* in networks that capture the uncertainty or possible node dependencies which have not

*Corresponding author

been observed in current edges [14]. The latent ties can be described as the joint likelihood of transmission probabilities and transmission times [15]. For example, consider the epidemic spreading in a network, if an infectious source infects a node multi-hops away, there is supposed to have a latent tie between those two nodes and the strength between them describes the probability of infections and also how long it would take before the infection happens. We notice that most graph auto-encoder models overlook the latent ties or simply approximate them by transmission probabilities within random walk paths [11], which fails to explore the rich information in latent networks.

To address the shortcomings, we present in this paper a latent auto-encoder with adversarial regularization (LAAE), which is designed as a general framework for complex network structures with sparse observations. Our core assumption here is that network structures should also be learnable. In addition to the general *feature learning* phase accomplished by auto-encoders, we pursue an extra *structure learning* phase to explore the latent network structures. Instead of approximating the latent ties by linear random walk sequences, the proposed model uses cascades to capture these complex structures. The cascades are related to the diffusion process over networks and can capture both transmission probabilities and duration times during the propagation of information. We seek to model the uncertainty and durations in latent networks and resort to network inference based on above cascades to endow latent ties and existing edges with exact strengths. Concretely, we attempt to build a *latent transmission matrix* that captures the weighted dependencies between nodes. Based on the latent transmission matrix, it is natural to define a latent global unit and a latent local unit to preserve the latent information. The global unit is to directly recover the connections from a global view, while the local unit is to make sure the nodes connected via latent ties and existing edges yield similar representations in the feature space.

Besides, although the auto-encoder paradigm can learn salient patterns from network data, the auto-encoders are generally regularized to preserve specific properties [2, 10]. We notice that forcing the network structures to be learnable brings extra noises and instability, which is unfavorable for the quality of embeddings. To this end, it is desirable to further regularize the feature learning phase to improve the model robustness and flexibility [16]. To embed the latent structures, we use an adversarial auto-encoder [17] to learn the node representations. The adversarial training introduces generative idea into the embedding step while avoiding explicitly defining a prior distribution. The generator is trained to approximate the distribution in the embedding space of the auto-encoder. Consequently, the joint optimization of the adversarial auto-encoder can improve the robustness of leaned features and filter noises from the structure learning phase.

To summarize, the contributions of this paper is as follows,

- We propose a latent network embedding method based on adversarial auto-encoders which can learn robust embeddings from informative latent structures.
- To discover latent structures, we propose a new strategy to learn the strengths of latent ties based on diffusion cascades which capture both transmission probabilities and times.
- We jointly learn from latent global information and latent local information and develop an adversarial term to avoid the influences of extra noises in the latent transmission matrix.
- We evaluate the proposed model in several graph analysis tasks. The results demonstrate the effectiveness of our model.

The reminder of this paper is organized as follows: Section 2 reviews the line of graph auto-encoders and also the network inference techniques related to our model. Section 3 gives the model implementation from a top-down fashion and introduces the optimization process of the model. The numerical results are reported in Section 4. We conclude our paper in Section 5.

2 Related Work

2.1 Graph Auto-encoders

The graph auto-encoders are considered as a basic framework for network analysis, especially the network embedding or network representation learning. The success of graph auto-encoders mainly lies in two aspects. First, they can be designed to assemble information from different perspectives (e.g., structures and attributes [18, 19], global view and local view [2, 20]) by imposing certain regularization terms. Second, they are flexible to incorporate various forms of encoder and decoder models that consist of multiple levels of non-linearity. Intuitively, if the reconstructions generated from the embedding space reveal desired information (e.g., the distribution of node degrees) in networks, the encoders are supposed to be reasonable feature extractors. With the encoder-decoder scheme in mind, early shallow models based on matrix factorization [21, 22] and random walks [3, 4] can also be understood within this framework

[23]. Unsurprisingly, the encoders in shallow models are usually formulated as look-up functions which have limited representation ability.

The extensions of auto-encoder framework to deep neural networks improve the model capacity and boost the performance in network embedding. A natural choice for encoders and decoders is the multi-layer perceptions [2, 11]. For example, SDNE [2] combines the deep auto-encoder with a Laplacian regularizer and jointly captures the first-order structures and second-order structures. Since local node sequences are also critical for representing network structures, advanced models mainly focus on modeling the contextual dependencies between nodes which reveal local proximities. In general, these models use sequences-to-sequence auto-encoders that can model the node sequences sampled by random walks, where the LSTMs are used as encoders and decoders [10]. When considering attributes associated with nodes in networks, the graph convolutional networks (GCNs) are adopted as feature extractors to learn from both structural information and attribute information [7, 24, 25].

There are also considerable graph auto-encoders that fall into the category of generative models, such as variational auto-encoders (VAEs) [26] and adversarial auto-encoders (AAEs) [17]. The variational auto-encoders attempt to embed the nodes to random vectors with a prior distribution (e.g., Gaussian) in the latent space [27, 28, 29]. The parameters are learned by optimizing the evidence lower bound [27]. In contrast, the AAEs are built under a general auto-encoder framework but with an adversarial regularization phase [16, 30]. For example, ANE [16] seeks to learn robust node features by approximating the posterior distribution defined by encoders to a given prior distribution. Our model is also built under the adversarial regularized framework but is designed for latent networks.

2.2 Network Inference

Network inference problem aims at inferring network edges from the observations of cascades which describe certain diffusion behaviors over graphs [31], e.g., information propagation [32] and epidemic spreading [33]. The studies of cascades relate to the influence maximization problem [34] and also network evolution problem [35]. Most network inference methods are built under the time-aware assumption where the cascades are truncated by a time constraint that terminates the diffusion process [36]. Estimating network structures boils down to discovering presence and transmission rates of edges via time-stamps and can be further built as an optimization problem by maximum likelihood estimation [31]. The transmission rates that adhere to edges can be denoted as conditional probabilities of infection from infected nodes to uninfected nodes [12]. Under the homogeneous setting, the information propagates in a statistically same manner along all edges [37]. NetInf [38] considers cascades as directed trees and proposes to infer the conditional transmission rates between nodes. To facilitate the optimization process, NetInf only considers the trees that are more likely to propagate in graphs. On the contrary, heterogeneous models relax the assumption and endow different transmission rates to edges. NetRate [39] models the conditional likelihood between nodes as a parametric form and proposes to optimize a convex inference problem based on observed cascades.

The general network inference problem can be easily extended to various settings. For example, the Bayesian inference models assume that both exact activation times and edges are unknown and required to be inferred [40, 41]. Besides, the network inference can be integrated into other frameworks to accomplish certain tasks, such as clustering [42] and recommendation system [43]. There are also a bunch of methods focus on representation learning under information cascades, which can also be used for inferring networks [44, 45]. Unlike those methods, we simulate the information diffusion process under a given network, which can explore rich information of latent structures.

3 Our Model

3.1 The Framework

We first introduce the notations and symbols that are used in this paper. Let $G(V, E)$ be a graph with a node set $V = \{v_i\}_{i=1}^N$ and an edge set $E \subseteq \{(v_i, v_j)\}_{i,j=1}^N$, where N is the number of nodes. The adjacent matrix of graph G is denoted as A , where $A_{i,j} = 1$ represents that there is an edge between v_i and v_j and otherwise $A_{i,j} = 0$. The Laplacian matrix is $L = D - A$ where $D \in \mathbb{R}^{N \times N}$ is a diagonal matrix $D_{i,i} = \sum_j A_{i,j}$. The neighbors of node v is denoted as $\mathcal{N}(v) = \{u \in V | (u, v) \in E\}$. The network embedding task maps nodes in a graph to a low-dimensional space where the representation vectors preserve the node proximities.

The graph auto-encoders typically use an encoder to map the proximity information in the node domain to a hidden space, from where a decoder can recover information to approximate the input. Formally, a basic graph auto-encoder attempts to minimize the reconstruction error,

$$\min_{\mathbf{x}} \sum dist(\mathbf{x}, g_{\psi}(f_{\phi}(\mathbf{x}))), \quad (1)$$

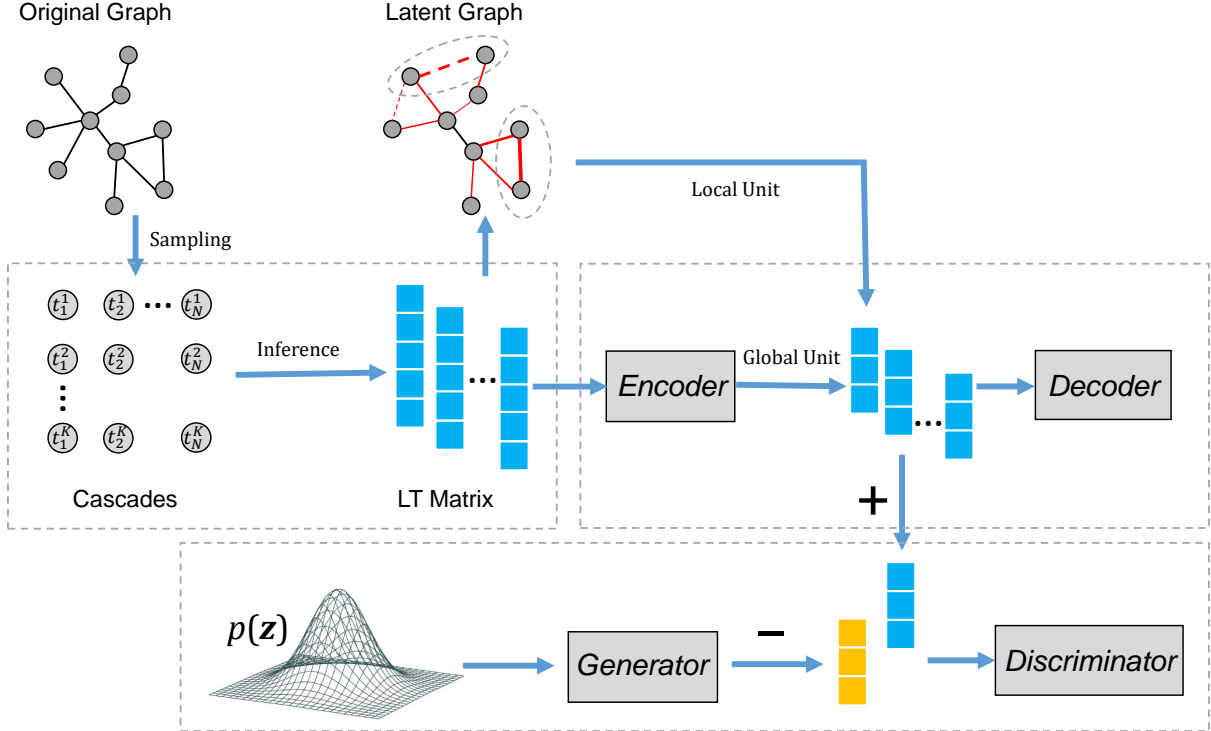


Figure 1: Overall framework of the proposed model. This framework consists of a structure learning phase (middle-left of the figure) and a feature learning phase (middle-right and bottom of the figure). The structure learning phase contains a diffusion sampling process and a network inference process. The cascades are generated to capture the high-order information. LT matrix denotes the latent transmission matrix inferred from cascades. The latent graph is built based on the LT matrix and the line widths denote the transmission rates or weights in edges. The dashed circles in the latent graph point out two types of edges. The dashed edges indicate newly inferred latent ties and the solid edges are existing edges in the original graph. The feature learning phase contains a graph auto-encoder and an adversarial regularization term. The global unit directly learns from the latent transmission matrix and keeps all connected and unconnected information via the encoder. The local unit learns from the pair-wise similarities drawn from the weighted latent graph.

where $dist(\cdot)$ is a metric that measures the distance between the input and reconstruction, $f(\cdot)$ and $g(\cdot)$ are the encoder and decoder parameterized by ϕ and ψ , respectively, \mathbf{x} is the input of the auto-encoder that is directly obtained from graph G . For example, \mathbf{x} can be defined as the global similarity matrix A .

The proposed model is designed under the framework of graph auto-encoders but implemented for latent networks. We first present the overall framework of our model in this subsection and then give the detailed illustrations of the model components in the following subsections. Our key concerns are two-fold: First, how to discover latent structures from already existed observations; Second, how to learn robust features from latent networks while avoiding the disturbance of noises. In general, network embedding via graph auto-encoders can be formulated as the following optimization problem,

$$\mathcal{L} = \mathcal{L}_{gae} + \lambda \mathcal{R}, \quad (2)$$

where \mathcal{L}_{gae} is the graph auto-encoder loss which learns embeddings from desired information drawn from graphs, \mathcal{R} is a regularization term that preserves certain properties, e.g., robustness, λ is a positive hyper-parameter. Under this framework, our first concern boils down to formulating inputs with latent information, which can be directly fed into the graph auto-encoders, and our second concern turns to exploring the model robustness by leveraging adversarial training as a regularization term.

An illustration of the proposed model is drawn in Figure 1. To address the above concerns, we combine the diffusion process with adversarial auto-encoders to solve the latent network embedding problem. Conventional network embedding methods directly learn from similarity matrices (e.g., the adjacent matrix A) drawn from graphs and approximate the static information. Consequently, it is desirable to unfold the undiscovered structures before feeding them into the embedding model. We refer to a structure learning phase to discover the latent structures. As shown in Figure

1, the diffusion sampling is used to generate cascades to capture high-order proximities in the graph. Then, a latent transmission matrix is derived by network inference from the cascades. To further learn node embeddings, we refer to a feature learning phase and use the adversarial auto-encoder to capture both global latent information and local latent information. The basic graph auto-encoder is regularized by an adversarial term which improves the model robustness.

3.2 Structure Learning Phase

In this subsection, we mainly discuss the structure learning phase and formulate the objective function. The goal of structure learning is to capture the latent information given the current observations. We define a weighted *latent transmission matrix* W where the elements represent the transmission rates between nodes. However, directly defining a learning model that infers W from A encounters difficulties, especially when the dimensions of A and W are large. To this end, we use the graph sampling method and simulate dynamic processes to capture the structural information in given static networks. We hence break the structure learning problem into two separate components. The diffusion sampling generates sequences that maximally keep the structural information while the following inference step reconstructs the connections via network inference.

Most networks embedding methods based on graph sampling use random walks to preserve local information, which are limited by their linear modeling and unstable performance. In this paper, we borrow the idea of diffusion process that has been widely used to model epidemic spreading and information propagation in networks. The diffusion process can reveal how many neighbors a node can influence in a network. The region of influence provides nice property to capture local structural information. Consequently, we can simulate diffusion processes and sample local structures from the original network. Recall that the random walks assume that a path is generated by iteratively moving from one node to another with a certain probability. This process omits the resistances of moving decided by the properties of networks. By contrast, diffusion process considers the easiness of transitions between nodes in addition to the transition probabilities. For simplicity, we follow the general diffusion setting and denote the easiness of transition as the time passing from one node to another [15]. At the beginning, an initially activated root node is randomly selected from the node set V with time-stamp 0. Then, the root node randomly activates a neighbor and the time interval of moving is also sampled. Then, all activated nodes in the current iteration can be used to activate their neighbors in the next iteration. After the diffusion sampling, we can obtain a set of activated nodes with time-stamps.

To facilitate the inference of latent ties, we consider all activated nodes and inactivated nodes simultaneously and use *cascades* to denote the results of diffusion samplings. Each cascade corresponds to a diffusion sampling started at a random node. Formally, a cascade \mathbf{c}^i is represented as an N -dimensional vector (t_1^i, \dots, t_N^i) where each element represents the activated time of a given node and N is the number of nodes in a graph. Only observations within a fixed duration time are considered and all time-stamps are clipped by a time window T . The time-stamps for inactivated nodes are denoted as ∞ . A set of cascades is represented as $\mathbf{C} : (\mathbf{c}^1, \dots, \mathbf{c}^K) \in \mathbb{R}^{K \times N}$ where K is the number of cascades.

We can then formulate our objective function to learn the latent transmission matrix based on the sampled cascades [39]. Instead of directly inferring $\phi(A; W)$, we use a bunch of observed cascades to infer the latent transmission matrix by maximizing the likelihood $\phi(\mathbf{C}; W)$. We begin the discussion of the inference problem considering a single cascade $\mathbf{c} \in \mathbf{C}$ and infer its likelihood $\phi(\mathbf{c}; W)$. Generally, given the time window $[0, T]$, $\phi(\mathbf{c}; W)$ can be seen as a joint likelihood of observing activated nodes (i.e., $t_i \leq T$) and inactivated nodes (i.e., $t_i > T$). The pair-wise transmission from v_i to v_j can be modeled by an exponential distribution in the following form,

$$f(t_j|t_i; W_{i,j}) = \begin{cases} W_{i,j} \cdot e^{-W_{i,j}(t_j-t_i)}, & \text{if } t_i < t_j, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Then, the probability that v_j is activated by v_i can be represented as the joint probability that v_j is activated by v_i and is not activated by other already activated nodes in the cascade. t_i can be interpreted as the first node to activate t_j . $\phi(t_j|t_i; W)$ can be denoted as,

$$\phi(t_j|t_i; W) = f(t_j|t_i; W) \prod_{\substack{t_k \neq t_i \\ t_k < t_j}} (1 - f(t_j|t_k; W)). \quad (4)$$

From Eq. (4), we can calculate $\phi(t_j; W)$ by summing up the likelihood $\phi(t_j|t_i; W)$ to denote all possible activations from v_i ($t_i < t_j$) to v_j ,

$$\begin{aligned}\phi(t_j; W) &= \sum_{t_i < t_j} \phi(t_j|t_i; W) \\ &= \sum_{t_i < t_j} \left[f(t_j|t_i; W) \prod_{\substack{t_k \neq t_i \\ t_k < t_j}} (1 - f(t_j|t_k; W)) \right].\end{aligned}\quad (5)$$

Then, the likelihood of observing all activations in a cascade $\phi(\mathbf{c}^{\leq T}; W)$ is the joint probability of $\phi(t_j; W)$ for all $t_j \leq T$,

$$\begin{aligned}\phi(\mathbf{c}^{\leq T}; W) &= \prod_{t_j \leq T} \phi(t_j; W) = \prod_{t_j \leq T} \left(\sum_{t_i < t_j} \phi(t_j|t_i; W) \right) \\ &= \prod_{t_j \leq T} \left(\sum_{t_i < t_j} \left[f(t_j|t_i; W) \prod_{\substack{t_k \neq t_i \\ t_k < t_j}} (1 - f(t_j|t_k; W)) \right] \right) \\ &= \prod_{t_j \leq T} \left(\sum_{t_i < t_j} \frac{f(t_j|t_i; W)}{1 - f(t_j|t_i; W)} \prod_{t_k < t_j} (1 - f(t_j|t_k; W)) \right)\end{aligned}\quad (6)$$

The last step is to make the product independent of i by removing the condition $k \neq i$.

The inactivated nodes v_l ($t_l > T$) provide contrast information which is also useful to infer the latent structures. Consequently, the likelihood function for an entire cascade \mathbf{c} can be represented as,

$$\begin{aligned}\phi(\mathbf{c}; W) &= \phi(\mathbf{c}^{\leq T}; W) \times \phi(\mathbf{c}^{> T}; W) \\ &= \prod_{t_j \leq T} \left(\sum_{t_i < t_j} \frac{f(t_j|t_i; W)}{1 - f(t_j|t_i; W)} \prod_{t_k < t_j} (1 - f(t_j|t_k; W)) \right) \\ &\quad \times \left(\prod_{t_l > T} \prod_{t_j \leq T} (1 - f(t_l|t_j; W)) \right),\end{aligned}\quad (7)$$

where $\phi(\mathbf{c}^{> T}; W)$ denotes that nodes v_l are not activated by already activated nodes v_j in the cascade.

Considering the independence observation of all cascades, the estimation over all observed cascades turns to obtaining $\phi(\mathbf{C}; W) = \prod_{\mathbf{c} \in \mathbf{C}} \phi(\mathbf{c}; W)$. The objective required to be optimized is denoted as,

$$\max_{W \geq 0} \sum_{\mathbf{c} \in \mathbf{C}} \log \phi(\mathbf{c}; W).\quad (8)$$

Following the optimization process in [46], we can obtain the update rule of $W_{i,j}$ as follows:

$$W_{i,j} = \begin{cases} \frac{1}{\mathbf{N}_c} \sum_{\mathbf{c} \in \{\mathbf{C}: t_j \leq T, t_i < t_j\}} \frac{1}{t_j - t_i}, \\ \frac{1}{\mathbf{N}_c} \sum_{\mathbf{c} \in \{\mathbf{C}: t_j > T, t_i \leq T\}} \frac{1}{T - t_i}. \end{cases}\quad (9)$$

where \mathbf{N}_c is a normalizer that calculates the total number of node pairs that satisfy the constraint. By solving (9), we can obtain the latent transmission matrix which can be used for further feature learning.

Information Flow Perspective. The structure learning phase can be simply stated as building a dynamic process which starts at A and ends up with an equilibrium W . Initially, the system is balanced and static, when we add new information into the system, i.e., starting a diffusion process at a random node, the information tends to flow through the edges and finally creates another balance system over the graph. The traces generated during the information flow record the transmission times and detect possible connections. Notice that the flow of information is a region-based spreading instead of a path-based spreading (e.g., random walk paths), our model is consequently able to capture more complex properties in the graph.

Comparison with PPMI Matrix. It is easy to find an analogy of the structure learning in random walk based methods which generate a probability transmission matrix named PPMI matrix [11, 16, 47]. PPMI matrix is based on statistic features obtained from random walks, which can be denoted as [48],

$$\text{PPMI}_{u,v} = \max(\text{PMI}_{u,v}, 0)\quad (10)$$

where $\text{PMI}_{u,v} = \log \frac{p(u,v)}{p(u)p(v)}$ and $p(\cdot)$ is the probability. PPMI matrix depicts the statistics of co-occurrences between nodes. In contrast, the structure learning in our model attempts to develop a data-driven method to model the co-occurrences of nodes and parameterizes connections so that they can be derived from a learning process. The latent transmission matrix defines a diffusion process under a network and infers the transmission rates based on an optimization problem via maximum likelihood estimation.

3.3 Feature Learning Phase

The structure learning phase rebuilds the connections via inference. The following goal is to learn the embeddings that capture the proximity information distributed in the latent transmission matrix. Following the previous work [2], we optimize a feature learning model within a global graph auto-encoder framework constrained by a local latent unit.

The global latent structure describes the common neighbors revealed by the shared nodes between \mathcal{N}_u and \mathcal{N}_v , which is accomplished by the graph auto-encoder framework proposed in Eq. (1). The objective of global latent unit is,

$$\mathcal{L}_{\text{GLO}} = \sum_{\mathbf{x}} \|\mathbf{x} - g_{\psi}(f_{\phi}(\mathbf{x}))\|_2^2. \quad (11)$$

where \mathbf{x} and $g_{\psi}(f_{\phi}(\mathbf{x}))$ are the input and reconstruction of the auto-encoder. We set $X = W$ to learn the global latent information from the latent transmission matrix. The embeddings are calculated by the encoder $\mathbf{y} = f_{\phi}(\mathbf{x})$. The encoder is a feature extractor which creates a distribution in the embedding space to describe the structural information in the node domain.

The local latent structure describes the direct neighbors connected by latent ties and already existing edges. Compared with previous local constrains, the key enhancement of our model is that we encourage the model to learn closer representations in the embedding space for nodes and their high-order neighbors. Besides, the local latent unit is to measure how much closeness between two nodes instead of measuring whether two nodes are close. The goal is accomplished by imposing continuous weights rather than binary codes to the local constraint. In detail, we use a local operator derived from Laplacian Eigenmaps as the local latent unit [21],

$$\mathcal{L}_{\text{Loc}} = \sum_{1 \leq i < j \leq N} W_{i,j} \|f_{\phi}(\mathbf{x}_i) - f_{\phi}(\mathbf{x}_j)\|_2^2, \quad (12)$$

where $W_{i,j}$ is the weight between two nodes.

Notice that the new equilibrium obtained by structure learning brings useful latent information but also noises. Without extra regularization terms, the graph auto-encoders are unlikely to separate useful information from the noises [9]. To solve this problem, we propose to use an adversarial regularization to guide the auto-encoder to learn robust features [17].

To start with, we introduce some basic concepts of adversarial methods. Training a Generative Adversarial Network (GAN) [49] establishes an adversarial process based on two components, a generator $G_{\theta}(\cdot)$ and a discriminator $D_{\delta}(\cdot)$. Formally, GAN wish to optimize a min-max objective w.r.t. the generator and discriminator,

$$\min_{\theta} \max_{\delta} \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})} [\log D_{\delta}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_g(\mathbf{z})} [\log(1 - D_{\delta}(G_{\theta}(\mathbf{z})))] \quad (13)$$

where $p_d(\mathbf{x})$ is the data distribution and $p_g(\mathbf{z})$ is a distribution where fake samples drawn from, δ is the parameter of discriminator and θ is the parameter of generator. The discriminator $D_{\delta}(\mathbf{x})$ depicts the probability that \mathbf{x} comes from the real data distribution instead of the noise. Later, the Wasserstein GAN [50] is proposed to overcome the unstable issues of training GAN, which leverages an Earth-Mover metric to measure the distance between two distributions. The objective is then revised as,

$$\min_{\theta} \max_{\delta \in \Delta} \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})} [D_{\delta}(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_g(\mathbf{z})} [D_{\delta}(G_{\theta}(\mathbf{z}))], \quad (14)$$

where Δ is the Lipschitz constraint that is kept by clipping the parameters within $[-c, c]$.

As illustrated in Figure 1, the goal of LAEE is to minimize the distance of the embedding distribution from the encoder $f_{\phi}(\mathbf{x}) \sim p_{\phi}(\mathbf{x})$ and the representation distribution from the generator $G_{\theta}(\mathbf{z}) \sim p_{\theta}(\mathbf{z})$. We use the Earth Moving distance $\mathcal{W}(p_{\phi}(\mathbf{x}), p_{\theta}(\mathbf{z}))$ following WGAN to measure the distance. Convert the Earth Moving distance to its dual form with smooth constraint ($\{D_{\delta}(\cdot)\}_{\delta \in \Delta}$ are all K -Lipschitz for some K), we can obtain,

$$\mathcal{W}(p_{\phi}(\mathbf{x}), p_{\theta}(\mathbf{z})) \approx \max_{\delta \in \Delta} \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})} [D_{\delta}(f_{\phi}(\mathbf{x}))] - \mathbb{E}_{\mathbf{z} \sim p_g(\mathbf{z})} [D_{\delta}(G_{\theta}(\mathbf{z}))]. \quad (15)$$

Under the min-max optimization framework, the discriminator and generator can be trained separately. The discriminator loss function can be denoted as,

$$\mathcal{L}_D(\delta) = -\mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})}[\mathbf{D}_\delta(f_\phi(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_g(\mathbf{z})}[\mathbf{D}_\delta(\mathbf{G}_\theta(\mathbf{z}))]. \quad (16)$$

The generator loss function can be written as,

$$\mathcal{L}_G(\theta) = \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})}[\mathbf{D}_\delta(f_\phi(\mathbf{x}))] - \mathbb{E}_{\mathbf{z} \sim p_g(\mathbf{z})}[\mathbf{D}_\delta(\mathbf{G}_\theta(\mathbf{z}))]. \quad (17)$$

In summary, we aim to jointly minimize the latent global loss, latent local loss and the adversarial regularization process. The objective can be denoted in the following form,

$$\mathcal{L}(\phi, \psi, \theta, \delta) = \mathcal{L}_{\text{GLO}} + \lambda_1 \mathcal{L}_{\text{LOC}} + \lambda_2 \mathcal{W}(p_\phi(\mathbf{x}), p_\theta(\mathbf{z})), \quad (18)$$

where λ_1 and λ_2 are two positive hyper-parameters. We can then derive the gradients of the parameters as follows,

$$\begin{aligned} \nabla_\phi \mathcal{L} &= \nabla_\phi \sum_{\mathbf{x}} \|\mathbf{x} - g_\psi(f_\phi(\mathbf{x}))\|_2^2 \\ &\quad + \lambda_1 \nabla_\phi \sum_{1 \leq i < j \leq N} W_{i,j} \|f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_j)\|_2^2 \\ &\quad + \lambda_2 \nabla_\phi \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})}[\mathbf{D}_\delta(f_\phi(\mathbf{x}))] \end{aligned} \quad (19)$$

$$\nabla_\psi \mathcal{L} = \nabla_\psi \sum_{\mathbf{x}} \|\mathbf{x} - g_\psi(f_\phi(\mathbf{x}))\|_2^2 \quad (20)$$

$$\nabla_\delta \mathcal{L} = -\lambda_2 \nabla_\delta \mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})}[\mathbf{D}_\delta(f_\phi(\mathbf{x}))] + \lambda_2 \nabla_\delta \mathbb{E}_{\mathbf{z} \sim p_g(\mathbf{z})}[\mathbf{D}_\delta(\mathbf{G}_\theta(\mathbf{z}))] \quad (21)$$

$$\nabla_\theta \mathcal{L} = -\lambda_2 \nabla_\delta \mathbb{E}_{\mathbf{z} \sim p_g(\mathbf{z})}[\mathbf{D}_\delta(\mathbf{G}_\theta(\mathbf{z}))] \quad (22)$$

Based on above derivatives, we can alternatively optimizing the different components of LAEE by block coordinate descent following [10]. The training of encoder (update ϕ) is not only determined by the locally regularized reconstruction phase that captures latent information, but also influenced by the adversarial process that improves the robustness of learned embeddings. The overall algorithm for our model is listed in Algorithm 1.

Algorithm 1 Latent Adversarial Auto-Encoder Training

Input: Graph $G(V, E)$, adjacent matrix A , hyper-parameter λ_1, λ_2 , number of iterations $iters$, batch size B

- 1: Run diffusion sampling process to obtain K cascades $\mathbf{C} : (\mathbf{c}^1, \dots, \mathbf{c}^K)$;
 - 2: Solve Eq. (8) to obtain the diffusion information matrix W ;
 - 3: Set $X = W$;
 - 4: **for** $iter < iters$ **do**
 - 5: **[Training Graph Auto-encoder]**
 - 6: Draw samples $\{\mathbf{x}\}_{i=1}^B$ from the data distribution $p_d(\mathbf{x})$;
 - 7: Calculate the loss of graph auto-encoder $\mathcal{L}_{gae} = \mathcal{L}_{\text{GLO}} + \lambda_1 \mathcal{L}_{\text{LOC}}$;
 - 8: Update the parameters ϕ and ψ for the encoder and decoder by (19) and (20);
 - 9: **[Training Discriminator]**
 - 10: Draw positive samples $\{\mathbf{x}\}_{i=1}^B$ from the data distribution $p_d(\mathbf{x})$;
 - 11: Draw negative samples $\{\mathbf{z}\}_{i=1}^B$ from a prior distribution $p_g(\mathbf{z})$;
 - 12: Calculate $f_\phi(\mathbf{x})$ and $\mathbf{G}_\theta(\mathbf{z})$;
 - 13: Calculate the loss of discriminator \mathcal{L}_D by (16);
 - 14: Update parameters of discriminator δ by (21);
 - 15: Clip the parameters δ within $[-c, c]$;
 - 16: **[Training Generator]**
 - 17: Draw negative samples $\{\mathbf{z}\}_{i=1}^B$ from a prior distribution $p_g(\mathbf{z})$;
 - 18: Calculate $\mathbf{G}_\theta(\mathbf{z})$;
 - 19: Calculate the loss of Generator \mathcal{L}_G by (17);
 - 20: Update parameters of generator θ by (22);
 - 21: **end for**
-

4 Experiments

To demonstrate the effectiveness of the proposed model, we evaluate our model in link prediction and node classification tasks.

Table 1: Dataset Statistics

| Datasets | # Nodes | # Edges | # Labels |
|----------|---------|---------|----------|
| PPI | 3,890 | 76,584 | None |
| ADV | 5,155 | 39,285 | None |
| JDK | 6,434 | 53,892 | None |
| BLOG | 10,312 | 333,983 | None |
| DBLP | 2,760 | 3,818 | 4 |
| BLOG5K | 5,196 | 171,743 | 6 |

4.1 Experiment Setup

We conduct the experiments under 6 different datasets. The statistics of all datasets are listed in Table 1. All networks only contain structural information. The PPI network, JDK network, ADV network and BLOG network are used for link prediction while the DBLP network and BLOG5K network are used for node classification.

- Protein-Protein Interactions (PPI) [4, 51] is a biological network which describes the directed interactions between proteins. The network used in this paper is a subgraph of the PPI network for Homo Sapiens.
- Advogato (ADV)² [52] is a trust network built under an online community platform named Advogato. The nodes are the users and the directed edges are their trust relationships.
- JDK Dependency (JDK)³ [10] is a software dependency network for JDK 1.6.0.7 framework. The nodes are the classes while the edges are the directed dependencies between them.
- Blogcatalog (BLOG) [53] is an undirected social network that depicts the social relations between bloggers. The nodes are the bloggers and the edges indicate their friendships.
- DBLP [54] is an academic network in computer science. We use the sub-network of DBLP which only contains four areas. Five representative conferences are selected from each area. The nodes are the authors that have published papers in these conferences and the edges are their co-authorships.
- Blogcatalog5k (BLOG5K) [55] is a variant of Blogcatalog network which contains 5196 nodes associated with labels. The labels are selected from pre-defined classes and describe the interests of the bloggers.

We compare the proposed model with several baseline models that are widely used in network embedding.

- DeepWalk [3] is a local network embedding method which depends on random walk samplings. The goal is to map the proximity within short node sequences to a low-dimensional space.
- node2vec [4] is another random walk based model, which extends the sampling strategy in DeepWalk to its biased version which contains Depth-First-Samplings and Breadth-First-Samplings.
- Spectral Clustering (SC) [56] or spectral embedding applies spectral decomposition to the Laplacian matrix of the network. The embeddings are accumulated by the top eigenvectors of the Laplacian matrix.
- Structural Deep Network Embedding (SDNE) [2] is a deep network embedding method which leverages a graph auto-encoder and a local constraint to learn the node representations, so that the local information and global information can both be captured.
- Adversarial Network Embedding (AAE) [16] is a deep graph auto-encoder framework which introduces adversarial training into the model.
- NetRA [10] is also an adversarially regularized graph auto-encoder based on random walk sequences. This framework uses long short-term memory network (LSTM) as an encoder to preserve local structures.

For our model, to generate sufficient cascades, all nodes in the network are set as root nodes. The maximal time window T is set as 10.

4.2 Link Prediction

We first apply the proposed model in the link prediction task to evaluate whether the learned representations can capture the proximities and structures in the node domain. The goal of link prediction is to predicted missing edges in a network

²<http://konect.uni-koblenz.de/networks/advogato>

³http://konect.uni-koblenz.de/networks/subelj_jdk

where part of its edges have been removed. We can then consider the link prediction task as a supervised classification framework based on edge features. We randomly removed 50% edges for all networks while keeping the remaining network connected. An equal number of negative samples are also sampled from the node pairs without any edges.

Once the node representation vectors are computed after the unsupervised training, we use an operator to generate the edge features $\mathbf{e} = op(\mathbf{y}(u), \mathbf{y}(v))$ where $\mathbf{y}(u)$ and $\mathbf{y}(v)$ are the low-dimensional vectors of node u and v , and \mathbf{e} is the edge features between them. We select Hadamard and Weighted-L2 as operators following the setting in node2vec [4]. The Hadamard operator \diamond is defined as $[\mathbf{y}(u) \diamond \mathbf{y}(v)]_i = \mathbf{y}_i(u) * \mathbf{y}_i(v)$, and the Weighted-L2 operator $\|\cdot\|_2$ is defined as $\|\mathbf{y}(u) \cdot \mathbf{y}(v)\|_2 = |\mathbf{y}_i(u) - \mathbf{y}_i(v)|^2$.

After obtaining all positive and negative edge vectors, the link prediction can be considered as a binary classification task. We draw the Receiver Operating Characteristic (ROC) curves and report the Area Under Curve (AUC) scores for all models on PPI network, ADV network, JDK network, and BLOG network.

The AUC scores are listed in Table 2. It can be observed that LAAE outperforms all baseline models including shallow models and recently proposed deep graph auto-encoders. LAAE achieved at least 2% performance improvements on all datasets. It is noticeable that introducing adversarial training to original networks not necessarily improves the performances in some datasets. For example, SDNE obtains higher results on BLOG compared with AAE and NetRA with Hadamard operator. In addition, the performances of different models may vary w.r.t. different operators. On PPI network, NetRA outperforms AAE with Hadamard operator and obtains opposite results with Weighted-L2 operator. The proposed LAAE obtains best performance in both operators, which illustrates that the learned embeddings capture underlying information in networks.

Table 2: Link Prediction Results w.r.t. AUC scores.

| Operator | Methods | PPI | ADV | JDK | BLOG |
|-------------|-------------|---------------|---------------|---------------|---------------|
| Hadamard | Deepwalk | 0.6795 | 0.7197 | 0.8401 | 0.6821 |
| | node2vec | 0.7259 | 0.7451 | 0.8652 | 0.7295 |
| | Spectral | 0.7302 | 0.7901 | 0.5952 | 0.6032 |
| | SDNE | 0.7782 | 0.8318 | 0.8863 | 0.8861 |
| | AAE | 0.8281 | 0.8527 | 0.9293 | 0.8661 |
| | NetRA | 0.8321 | 0.8980 | 0.9148 | 0.8532 |
| | LAAE | 0.8752 | 0.9105 | 0.9444 | 0.9473 |
| Weighted-L2 | Deepwalk | 0.7715 | 0.8414 | 0.8472 | 0.8596 |
| | node2vec | 0.8048 | 0.8570 | 0.8445 | 0.8897 |
| | Spectral | 0.7179 | 0.7493 | 0.8551 | 0.7602 |
| | SDNE | 0.7282 | 0.7959 | 0.8643 | 0.7235 |
| | AAE | 0.8295 | 0.8590 | 0.8515 | 0.7626 |
| | NetRA | 0.7816 | 0.8500 | 0.8947 | 0.9093 |
| | LAAE | 0.8404 | 0.8821 | 0.9417 | 0.9312 |

The ROC curves on PPI, ADV, JDK, and BLOG are also plotted in Figure 2, Figure 3, Figure 4, Figure 5, respectively. The curves of the proposed LAAE are very close to the top-left corner compared with all baseline models, which indicates better representation ability for our model from another perspective.

4.3 Node Classification

In this subsection, we apply the representation vectors learned from our model to node classification task. Predicting the categories of nodes is a fundamental task in graph-based studies. We follow the experimental settings in DeepWalk and feed the learned features into the one-vs-rest logistic regression model [57]. To test the quality of features, the training samples used in the supervised phase are randomly split for training and test. The ratio of training samples ranges from 10% to 90%. We report the average Micro-F1 scores and Macro-F1 scores after ten runs for our model and baseline models on DBLP network and BLOG5K network.

The performances are shown in Figure 6 and Figure 7, respectively. LAAE achieves better classification results in both datasets, which indicates the learned embeddings capture useful information in networks. The performance gains in BLOG5K are more significant than that in DBLP. We also notice that, in a denser network such as BLOG5K, deep models achieve more stable results with respect to different ratios of training samples. LAAE is less affected by the sparsity of networks since the structure learning phase detects latent ties and increases the information in networks.

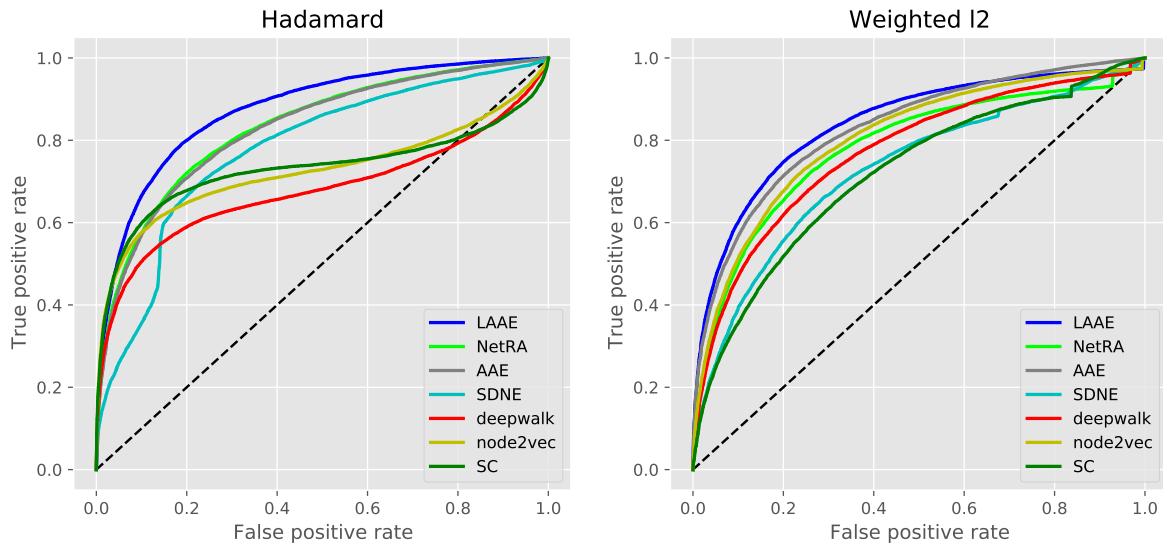


Figure 2: ROC curves on PPI network with Hadamard operator and Weighted L2 operator.

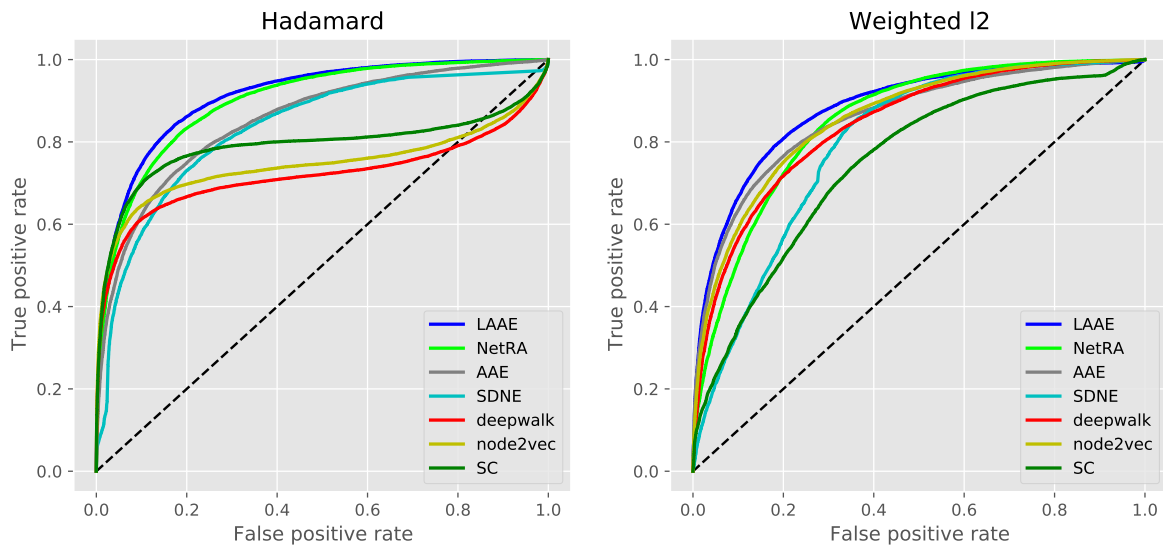


Figure 3: ROC curves on ADV network with Hadamard operator and Weighted L2 operator.

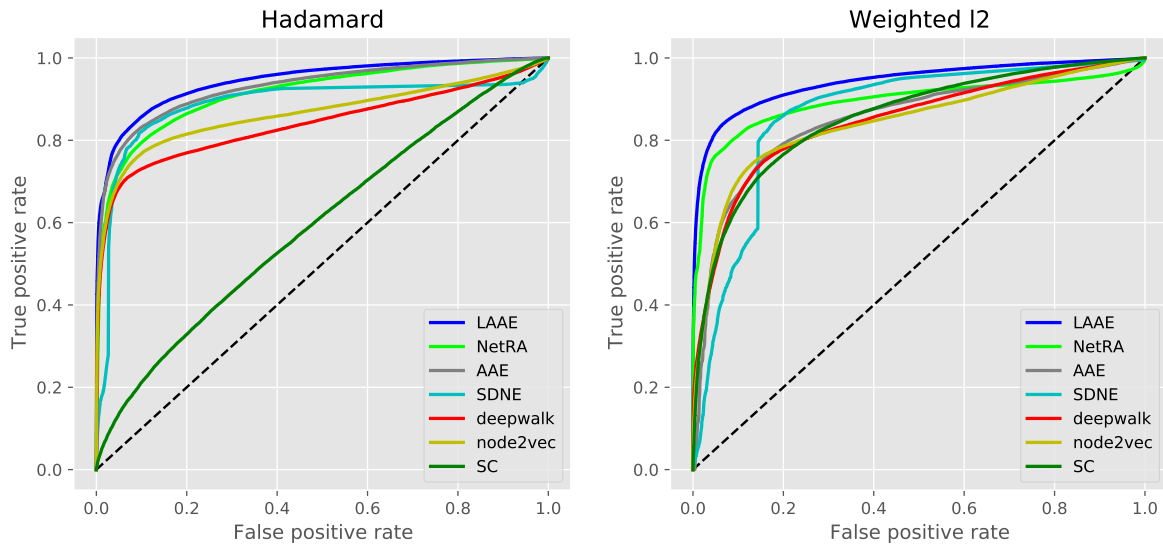


Figure 4: ROC curves on JDK network with Hadamard operator and Weighted L2 operator.

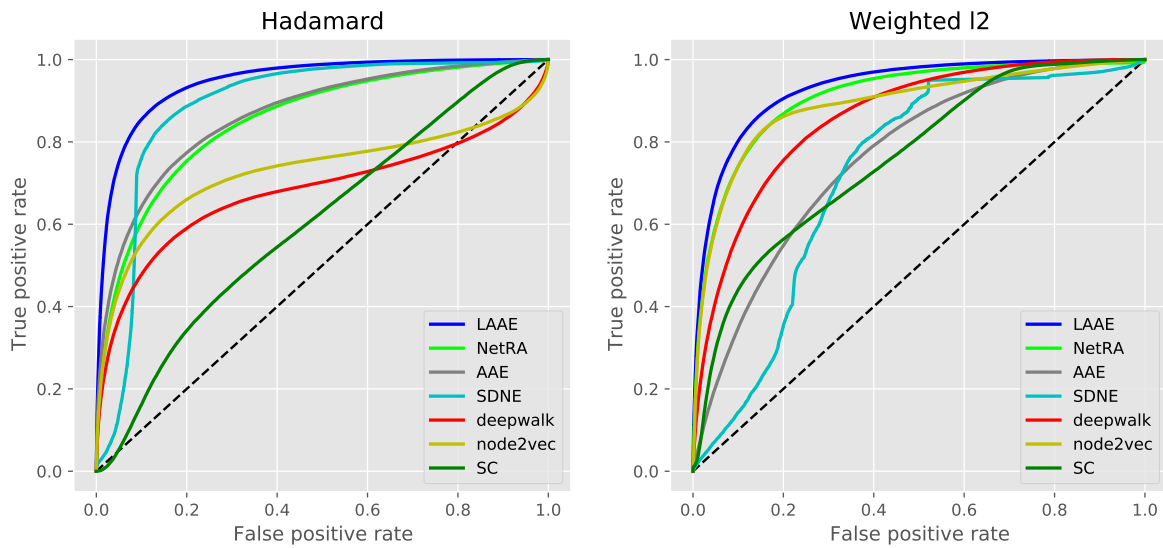


Figure 5: ROC curves on BLOG network with Hadamard operator and Weighted L2 operator.

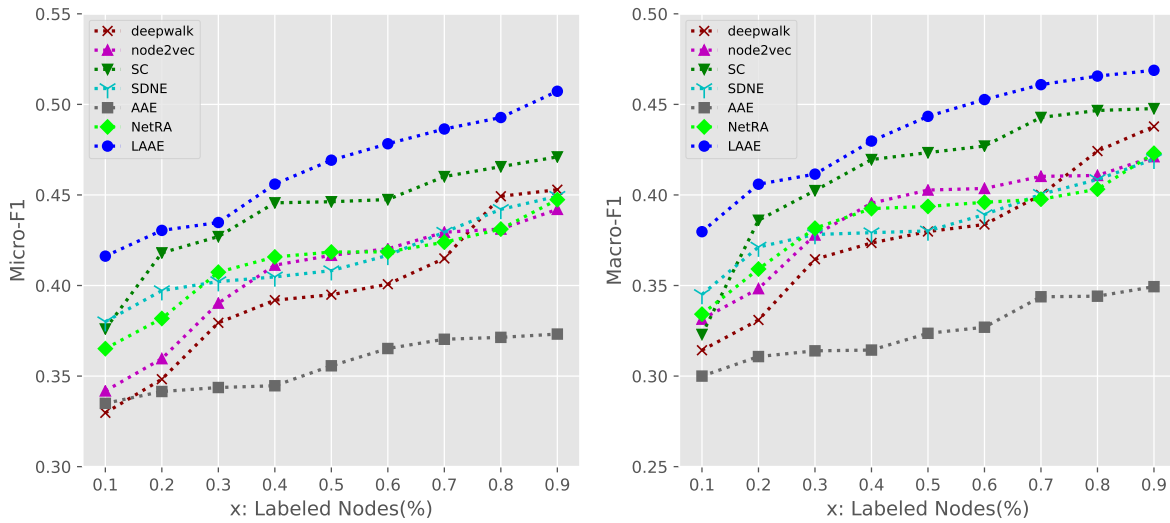


Figure 6: Classification results on DBLP w.r.t. Micro-F1 and Macro-F1 scores with the training ratio varies from 10% to 90%.

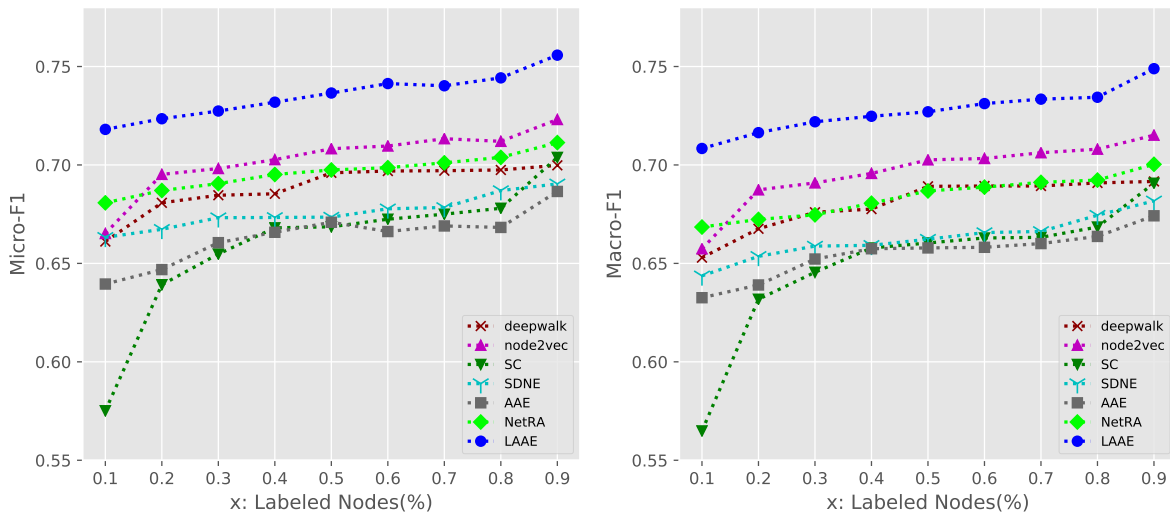


Figure 7: Classification results on BLOG5K w.r.t. Micro-F1 and Macro-F1 scores with the training ratio varies from 10% to 90%.

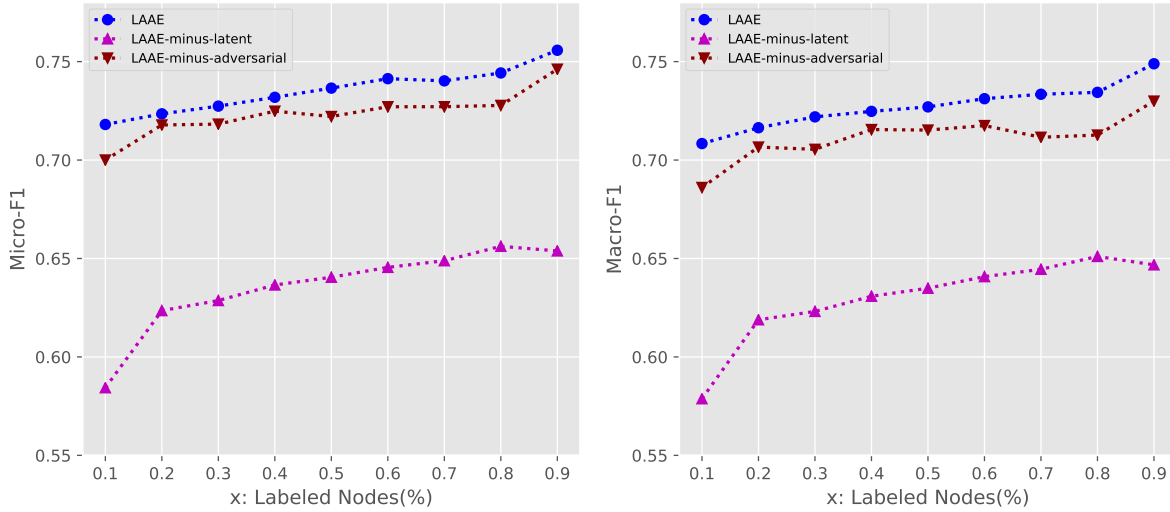


Figure 8: Comparisons between LAAE and its reduced versions on BLOG5K w.r.t. Micro-F1 and Macro-F1 scores. LAAE-minus-latent indicates removing latent ties from LAAE and LAAE-minus-adversarial indicates removing adversarial regularization from LAAE.

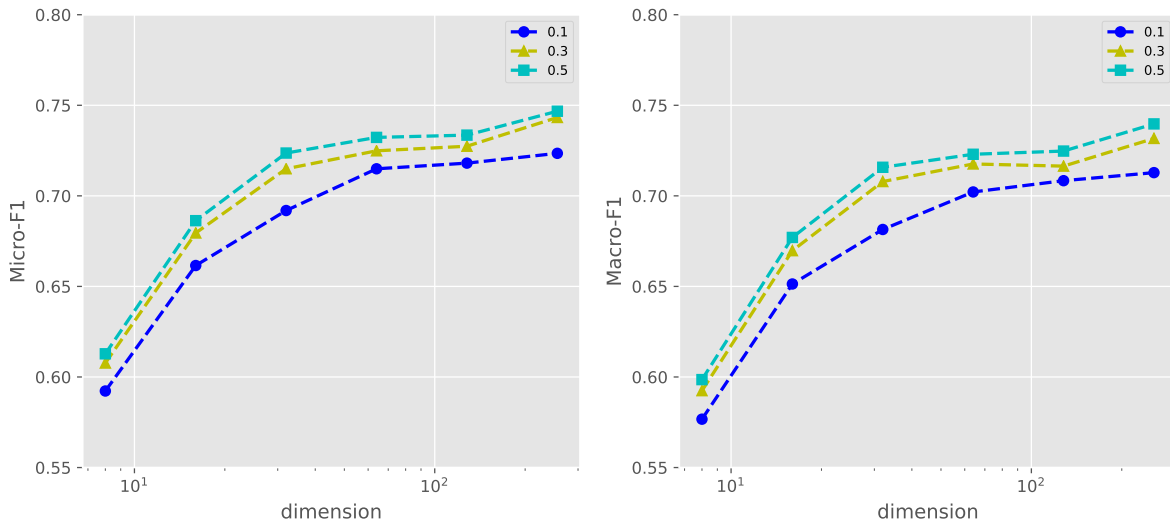


Figure 9: Performances in different dimensions ($d = 8, 16, 32, 64, 128, 256$) on BLOG5K w.r.t. Micro-F1 and Macro-F1 scores.

4.4 Model Analysis and Parameter Study

To further understand the roles of different parts in our model, we conduct several experiments to analyze our model from different perspectives. In this subsection, we would first like to answer the following questions: (1) How does latent structure learning affect the performances. (2) What is the role of adversarial regularization in our model. To answer these two questions, we separately remove the latent inference part and adversarial part and record the performance obtained in BLOG5K. The results of ablation experiments are plotted in Figure 8 accordingly. LAAE-minus-latent is the model that removes the latent ties from LAAE. The adjacent matrix A without any latent information is directly fed into the adversarially regularized auto-encoders and the embeddings are learned by optimizing Eq. (18). LAAE-minus-adversarial represents the model that removes the adversarial regularization term from LAAE.

We observe that without latent ties, the performance drops dramatically w.r.t. both Micro-F1 and Macro-F1 scores. Besides, the results are easily affected by the shortage of training samples (i.e., training ratio is 0.1) when the latent ties are removed. The role of latent ties is to increase the information hidden behind the observed network, which can largely boost the quality of learned features and also the classification results. Since introducing latent ties relieves the sparsity issue in original networks, the performance is more stable when the training ratios are relatively small, which is of great meaning in real scenarios where the node labels are generally difficult to obtain. When removing the adversarial regularization, the results experience a slight decrease. The performance fluctuates when the training ratios changes. The role of adversarial regularization is to improve the feature robustness and model stability. The unfavorable noises brought during the process of inferring latent ties can be filtered by the adversarial training.

To analyze the parameters, we further study the influences of hidden dimensions in our model. The results of classification in BLOG5K with respect to different dimensions are plotted in Figure 9. The performance ratchets up when the dimension increases and levels off when the dimension reaches 64.

5 Conclusion

We proposed in this paper a latent network embedding model based on adversarial auto-encoders, which served as a general framework for unsupervised representation learning in plain networks. First, the proposed model discovered latent ties from partial observations to capture the unobserved underlying structures. Second, our work aimed to distinguish the strong connections and weak connections of the latent graph by equipping the edges with different weights. The above steps provided a more comprehensive and precise way to describe the network structures. The experiments on link prediction and node classification showed that the introduction of latent information can boost the performance. Third, to cope with the possible noises in latent networks, we used an adversarial regularization term to increase the robustness of learned features. The ablation study showed that the adversarial term can improve the stability of learned features when varying the ratio of training samples.

The future direction of this paper first point to developing an end-to-end framework that can jointly optimize the structure learning phase and feature learning phase. It is also promising to apply our work to more complex graph types such as attribute graphs and heterogeneous networks. For example, consider a heterogeneous graph in recommendation systems where nodes have different types, inferring a latent graph not only provides more information about the similarities of users or items, but also the underlying preferences between users and items.

References

- [1] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 119–128, 2015.
- [2] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, 2016.
- [3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, 2014.
- [4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, 2016.
- [5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.
- [6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

- [7] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially regularized graph autoencoder for graph embedding,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 2609–2615, 2018.
- [8] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the International Conference on Machine Learning*, pp. 1096–1103, 2008.
- [9] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.
- [10] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, “Learning deep network representations with adversarially regularized autoencoders,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2663–2671, 2018.
- [11] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1145–1152, 2016.
- [12] S. Myers and J. Leskovec, “On the convexity of latent social network inference,” in *Advances in Neural Information Processing Systems*, pp. 1741–1749, 2010.
- [13] S. Linderman and R. Adams, “Discovering latent network structure in point process data,” in *Proceedings of the International Conference on Machine Learning*, pp. 1413–1421, 2014.
- [14] C. Haythornthwaite, “Strong, weak, and latent ties and the impact of new media,” *The Information Society*, vol. 18, no. 5, pp. 385–401, 2002.
- [15] Y. Shi, M. Lei, H. Yang, and L. Niu, “Diffusion network embedding,” *Pattern Recognition*, vol. 88, pp. 518–531, 2019.
- [16] Q. Dai, Q. Li, J. Tang, and D. Wang, “Adversarial network embedding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
- [17] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, “Adversarial autoencoders,” in *International Conference on Learning Representations Workshop*, 2016.
- [18] D. Jin, B. Li, P. Jiao, D. He, and W. Zhang, “Network-specific variational auto-encoder for embedding in attribute networks,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 2663–2669, 2019.
- [19] B. Yu, J. Hu, Y. Xie, C. Zhang, and Z. Tang, “Rich heterogeneous information preserving network representation learning,” *Pattern Recognition*, vol. 108, p. 107564, 2020.
- [20] Z. Zhang, D. Chen, Z. Wang, H. Li, L. Bai, and E. R. Hancock, “Depth-based subgraph convolutional auto-encoder for network representation learning,” *Pattern Recognition*, vol. 90, pp. 363–376, 2019.
- [21] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *Advances in Neural Information Processing Systems*, pp. 585–591, 2002.
- [22] S. Cao, W. Lu, and Q. Xu, “Grarep: Learning graph representations with global structural information,” in *Proceedings of the ACM International on Conference on Information and Knowledge Management*, pp. 891–900, 2015.
- [23] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” *IEEE Data Engineering Bulletin*, vol. 40, pp. 52–74, 2017.
- [24] D. Wu, R. Hu, Y. Zheng, J. Jiang, N. Sharma, and M. Blumenstein, “Feature-dependent graph convolutional autoencoders with adversarial training methods,” in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8, IEEE, 2019.
- [25] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, “Symmetric graph convolutional autoencoder for unsupervised graph representation learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6519–6528, 2019.
- [26] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *International Conference on Learning Representations*, 2014.
- [27] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” in *NIPS Workshop on Bayesian Deep Learning*, 2016.
- [28] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, “D-vae: A variational autoencoder for directed acyclic graphs,” in *Advances in Neural Information Processing Systems*, pp. 1588–1600, 2019.
- [29] A. Sarkar, N. Mehta, and P. Rai, “Graph representation learning via ladder gamma variational autoencoders,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5604–5611, 2020.

- [30] S. Pan, R. Hu, S.-f. Fung, G. Long, J. Jiang, and C. Zhang, “Learning graph embedding with adversarial training methods,” *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2475–2487, 2019.
- [31] J. Pouget-Abadie and T. Horel, “Inferring graphs from cascades: A sparse recovery framework,” in *Proceedings of the International Conference on Machine Learning*, pp. 977–986, 2015.
- [32] S. Bourigault, S. Lamprier, and P. Gallinari, “Representation learning for information diffusion through social networks: an embedded cascade model,” in *Proceedings of the Ninth ACM international conference on Web Search and Data Mining*, pp. 573–582, 2016.
- [33] J. Hoffmann and C. Caramanis, “Learning graphs from noisy epidemic cascades,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 2, p. 40, 2019.
- [34] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, “Influence maximization on social graphs: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1852–1872, 2018.
- [35] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, “Kronecker graphs: an approach to modeling networks.,” *Journal of Machine Learning Research*, vol. 11, no. 2, 2010.
- [36] M. Farajtabar, Y. Wang, M. Gomez-Rodriguez, S. Li, H. Zha, and L. Song, “Coevolve: A joint point process model for information diffusion and network evolution,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1305–1353, 2017.
- [37] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, “Learning graphs from data: A signal representation perspective,” *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.
- [38] M. Gomez Rodriguez, J. Leskovec, and A. Krause, “Inferring networks of diffusion and influence,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1019–1028, 2010.
- [39] M. Gomez Rodriguez, D. Balduzzi, B. Schölkopf, G. T. Scheffer, *et al.*, “Uncovering the temporal dynamics of diffusion networks,” in *Proceedings of the International Conference on Machine Learning*, pp. 561–568, 2011.
- [40] E. Sefer and C. Kingsford, “Convex risk minimization to infer networks from probabilistic diffusion data at multiple scales,” in *Proceedings of the IEEE International Conference on Data Engineering*, pp. 663–674, IEEE, 2015.
- [41] S. Shaghaghian and M. Coates, “Online bayesian inference of diffusion networks,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 500–512, 2017.
- [42] L. Prokhorenkova, A. Tikhonov, and N. Litvak, “Learning clusters through information diffusion,” in *Proceedings of the World Wide Web Conference*, pp. 3151–3157, 2019.
- [43] S. Mukherjee and S. Günnemann, “Ghostlink: Latent network inference for influence-aware recommendation,” in *Proceedings of the World Wide Web Conference*, pp. 1310–1320, 2019.
- [44] C. Li, J. Ma, X. Guo, and Q. Mei, “Deepcas: An end-to-end predictor of information cascades,” in *Proceedings of the International Conference on World Wide Web*, pp. 577–586, 2017.
- [45] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, “Community preserving network embedding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.
- [46] Q. Zhang, J. Wu, P. Zhang, G. Long, I. W. Tsang, and C. Zhang, “Inferring latent network from cascade data for dynamic social recommendation,” in *Proceedings of the IEEE International Conference on Data Mining*, pp. 669–678, IEEE, 2016.
- [47] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, “Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec,” in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 459–467, 2018.
- [48] J. A. Bullinaria and J. P. Levy, “Extracting semantic representations from word co-occurrence statistics: A computational study,” *Behavior Research Methods*, vol. 39, no. 3, pp. 510–526, 2007.
- [49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.
- [50] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *Proceedings of the International Conference on Machine Learning*, pp. 214–223, 2017.
- [51] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood, *et al.*, “The biogrid interaction database: 2008 update,” *Nucleic acids research*, vol. 36, no. suppl_1, pp. D637–D640, 2007.

-
- [52] P. Massa, M. Salvetti, and D. Tomasoni, “Bowling alone and trust decline in social network sites,” in *Proceedings of the IEEE International Conference on Dependable, Autonomic and Secure Computing*, pp. 658–663, IEEE, 2009.
 - [53] L. Tang and H. Liu, “Relational learning via latent social dimensions,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 817–826, 2009.
 - [54] Y. Sun, Y. Yu, and J. Han, “Ranking-based clustering of heterogeneous information networks with star network schema,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 797–806, 2009.
 - [55] X. Huang, J. Li, and X. Hu, “Label informed attributed network embedding,” in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 731–739, 2017.
 - [56] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Advances in Neural Information Processing Systems*, pp. 849–856, 2002.
 - [57] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” *Journal of Machine Learning Research*, vol. 9, no. Aug, pp. 1871–1874, 2008.