Pattern Recognition 136 (2023) 109266

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

AdaNS: Adaptive negative sampling for unsupervised graph representation learning

Yu Wang[a,c], Liang Hu[a], Wanfu Gao[a,b,*], Xiaofeng Cao[c,*], Yi Chang[c]

[a] *College of Computer Science and Technology, Jilin University, Changchun 130012, China*
[b] *College of Chemistry, Jilin University, Changchun 130012, China*
[c] *School of Artificial Intelligence, Jilin University, Changchun 130012, China*

A R T I C L E I N F O

Article history:
Received 8 February 2021
Revised 25 November 2022
Accepted 14 December 2022
Available online 15 December 2022

Keywords:
Graph representation learning
Negative sampling
Noise contrastive estimation

A B S T R A C T

Recently, unsupervised graph representation learning has attracted considerable attention through effectively encoding graph-structured data without semantic annotations. To accelerate its training, noise contrastive estimation (NCE) samples uniformly negative examples to fit an unnormalized graph model. However, this uniform sampling strategy may easily lead to slow convergence, even the vanishing gradient problem. In this paper, we theoretically show that sampling those hard negatives close to the current anchor can relieve the above difficulties. With this finding, we then propose an **Ada**ptive **N**egative **S**ampling strategy, namely AdaNS, which efficiently samples the hard negatives from the mixing distribution regarding the dimensional elements of the current node representation. Experiments show that our AdaNS sampling strategy applied on top of representative unsupervised models, e.g., DeepWalk, GraphSAGE, can outperform the existing negative sampling strategies in the tasks of node classification and visualization. This also further demonstrates that sampling those hard negatives can bring performance improvements for learning the node representations.

1. Introduction

With the explosive generation of graph-structured data in the real world, graph representation learning methods that learn latent, informative, and low-dimensional representations for nodes have been rapidly developed in recent years [1–3]. Among them, supervised learning methods have achieved desirable performance in several graph-related tasks relying on extensive manual semantic annotations [2,4]. As it is impractical to obtain large-scale annotations, there are obstacles to the scaling up of the supervised methods. In response, we have witnessed an increasing interest in developing graph representation learning without semantic annotations, also known as unsupervised graph representation learning [3,5–7].

Recent unsupervised graph representation learning methods using contrastive objectives have achieved remarkable results [1,3]. Contrastive methods comprise a trainable encoder and a pair of samplers for positive and negative examples. Specifically, a contrastive method may employ a scoring function, training the encoder to increase the score on ``real'' input and decrease the score on ``fake'' input. Essentially, contrastive methods are developed as estimators, which optimize the encoder to maximize the compatibility between the representations of positive examples and push the representations of negative examples apart. However, it is impractical to optimize straightforwardly, unless the compatibility metric is unnormalized. Negative sampling [8], as a simplified version of Noise Contrastive Estimation (NCE) [9,10], is a feasible strategy for approximate optimization, which simplifies the estimation of the normalized compatibility as a logistic regression problem that discriminates between positive and negative samples. Therefore, the contrastive method with negative sampling is developed as a discriminator. Based on whether the encoder in the discriminator employs a single-hidden-layer forward neural network or a message-passing scheme, contrastive methods can be categorized into two groups: shallow network embedding methods [7,11,12] and Graph Neural Networks (GNNs) embedding methods [2,3,5]. Sampling robust and generic positive nodes have been the main focus for improving the performance of shallow network embedding methods in recent years [1,7,12]. Meanwhile, the line of GNNs is dedicated to designing advanced encoders [2,5]. Another critical component, namely the negative sampling strategy, is not sufficiently explored. We argue that the strategy in which we choose negative samples can significantly affect the quality of

* Corresponding authors.
E-mail addresses: yu_w18@mails.jlu.edu.cn (Y. Wang), 543786450@qq.com (L. Hu), gaowf@jlu.edu.cn (W. Gao), xiaofengcao@jlu.edu.cn (X. Cao), yichang@jlu.edu.cn (Y. Chang).

https://doi.org/10.1016/j.patcog.2022.109266
0031-3203/© 2022 Elsevier Ltd. All rights reserved.

the representations. For instance, distinguishing a pair of similar samples gives completely different feedback to the encoder than distinguishing a pair of absolutely unrelated samples. More fine-grained discrimination may benefit the expressiveness of the node representations.

Since the negative sampling is derived from the simplification of NCE, the uniform sampling strategy is naturally followed. However, the uniform sampling strategy suffers seriously from slow convergence, even the vanishing gradient problem, which prevents the model from achieving the desired representations [13]. To relieve the issues caused by the uniform sampling strategy, modeling adaptive negative sampling distributions based on the current training process emerges as a solution for sampling high-quality negative examples [14–16]. In essence, there are several significant limitations. First, common adaptive sampling strategies based on generative adversarial network (GAN) architectures with extra generators introduce a myriad of parameters, which significantly limits the scaling to big graphs. Moreover, even if adaptive negative sampling may relieve the vanishing gradient problem, in practice its impact on the expressiveness of the yielded node representations in downstream tasks is unknown.

In this paper, we theoretically prove that adaptively sampling hard negatives close to the anchor can relieve the vanishing gradient problem and speed up the model training. Then, we propose an efficient **Ada**ptive **N**egative **S**ampling strategy, named AdaNS, which exploits the mixing probability of distributions with respect to dimensional elements to sample negative nodes efficiently. We then apply the proposed AdaNS on representative unsupervised graph representation learning models, namely DeepWalk [7] and GraphSAGE [5], to sample negatives, yielding informative representations for downstream node classification and visualization tasks. Extensive experiments are conducted to evaluate the proposed negative sampling strategy. We experimentally find that sampling hard negatives are beneficial in improving the performance of the node representations on downstream tasks. The contributions of this work can be summarized as follows:

- We mathematically prove that sampling hard negatives close to the anchor may relieve the vanishing gradient problem and speed up the model training.
- We design an efficient adaptive negative sampling strategy named AdaNS, which not only relieves the vanishing gradient problem but also alleviates the computational consumption.
- We conduct experiments on DeepWalk and GraphSAGE models, applying the proposed AdaNS strategy to sample negatives, to optimize the node representations for node classification and visualization tasks. The experimental results on seven real-world standard graph datasets show the superior performance of the proposed AdaNS.

The remainder of this paper is organized as follows: Section 2 surveys the related work. Section 3 covers notations and necessary preliminaries. In Section 4, we present theoretical insights on noise contrastive estimation and negative sampling. The proposed model is presented in Section 5. Section 6 reports the experimental results. Finally, Section 7 concludes this paper and suggests a future direction.

2. Related Work

The proposed work builds on a rich line of recent research regarding graph representation learning and negative sampling strategies. This section reviews related work, including graph representation learning and negative sampling, to facilitate the acquaintance of researchers.

2.1. Graph Representation Learning

A wide variety of graph representation learning models have been proposed in the past few years, which fall into two categories: traditional shallow network embedding methods and graph neural networks (GNNs). Shallow network embedding methods attempt to optimize the node embeddings as parameters by minimizing a reconstruction error, and this type of model is devoted to studying the sampling strategy of positive node pairs. Initially proposed by Perozzi et al. [7], DeepWalk employs random walk sequences to explore the structural information in a graph and then learns node embeddings based on the skip-gram model [8] by maximizing the log-likelihood of the ǣcontextǥ nodes within a fixed window on the sequences for the given node. LINE [11] extends DeepWalk with both first- and second-order neighbor nodes sampled as positive node pairs and first formally applies negative sampling for graph representation learning. After that, Node2vec [1] proposes a biased random walk strategy considering both depth-first and breadth-first search strategies, based on which the sampling of positive node pairs can be flexibly adjusted for various tasks and graphs. SDNE [17] employs a deep autoencoder to capture the high non-linearity in graphs. AROPE [18] further captures the arbitrary-order node relationships by performing eigendecomposition on the adjacency matrix. Furthermore, there are some models for sampling positive node pairs that take advantage of other node properties on the graph, such as Personalized PageRank [19,20], diffusion patterns [21,22], structural roles [12,22], and adjacency matrices [23,24]. Rather than learning parametric embeddings, GNNs learn mappings from graph structure and node features into embeddings, which are trained end-to-end supervised or semi-supervised by neural network parametrization [2,25]. The original GCN algorithm [2] is proposed to adopt the localized 1-step spectral convolution to design the message-passing layer for the semi-supervised classification task. GraphSAGE [5] extends GCN to the unsupervised task by employing trainable aggregation functions to sample positive and negative examples for the contrastive objective. Afterwards, DGI [6] attempts to train the contrastive model by relying on maximizing local mutual information between graph-level and node-level embeddings. Similarly, GRACE [26] maximizes the agreement of node embeddings generated by two various augmentations. Collectively, these unsupervised methods can be unified into the graph contrastive learning paradigm, and a more comprehensive survey of graph representation learning is provided by [27].

2.2. Negative Sampling

Negative sampling is originally proposed to reduce the complexity of softmax in word2vec [8]. From the perspective of the sampling scheme, existing negative sampling strategies can be categorized into two types: static negative sampling and adaptive sampling for hard negatives. Static negative sampling refers to applying a predefined noise distribution to all nodes, such as degree-based sampling [8], uniform sampling [28], and WRMF [29]. Although the static sampling strategy is easy to implement, it suffers from the vanishing gradient and thereby cannot achieve the desired performance. The more informative the negative sample can be by drawing a finite number of negative samples, the larger the magnitude of the gradient of the loss function becomes, allowing training to be accelerated. The adaptive sampling strategies are intended for this purpose. DNS [30] is proposed to dynamically choose negative samples from the ranked list produced by the current prediction scores. WARP [31] follows the assumption that the rating score of positive items should be higher than those of negative items, so that for a given positive item, it samples all items uniformly until a negative item is found. PinSAGE [32] adds ǣhard

Table 1
Summary of the Main Notations.

$\mathcal{G}, \mathcal{V}, \mathcal{E}$	Graph, Node set, Edge set
$\mathcal{S}$	Sequence of nodes
d	Dimension of the learned node representation
$\mathcal{F}$	Mapping function from nodes to d-dimensional representations
$\mathbf{z}_v$	Representation vector or embedding vector of node v
w, h	Realizations of the anchor and context nodes
Θ	Model parameters
$g_\theta(\cdot)$	Encoder function
$s_\theta(\cdot,\cdot)$	Scoring function
$\cdot^\top$	Transpose of a vector
$p(h\|w)$	Probability conditioned on w
$P_d(\cdot), P_n(\cdot)$	Data distribution and noise distribution
$\mathcal{J}$	Objective function
$\sigma(\cdot)$	Sigmoid function
$\text{sgn}(\cdot)$	Sign of the scalar

negative itemsg according to their Personalized PageRank scores to provide fine enough æresolutiong for the system to learn. Recently, the GAN-based negative sampling strategy, which has achieved excellent performance in information retrieval [14] and knowledge graphs [33], has also been transferred to graph representation learning and achieved good performance [15]. Moreover, InterCLR [34] and Ring [35] argue that the most similar examples might be better suited as positives rather than negatives. Thus, they choose fairly similar examples, but not too hard ones, as negatives.

3. Preliminaries

In this section, we introduce the definition and notations for unsupervised graph representation learning and then present a typical model - DeepWalk. Main notations are summarized in Table 1.

3.1. Unsupervised Graph Representation Learning

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$ and edge set $\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$. Unsupervised graph representation learning aims to learn a mapping function $\mathcal{F} : v_i \to \mathbf{z}_i \in \mathbb{R}^d$ that maps each node to a d-dimensional representation, where $d \ll |\mathcal{V}|$ and $\mathbf{z}_i$ denotes the embedded vector of node v_i. Desired node representations should preserve both structural and semantic information, thereby facilitating downstream tasks such as node classification and visualization.

3.2. DeepWalk

To thoroughly understand the important role of negative sampling in unsupervised graph representation learning, we review a classical method, namely DeepWalk [7]. First, we can split each node v_i into two roles: anchor node w_i and context node h_i of other anchor nodes. Given the anchor node w, it obtains a sequence of nodes $\mathcal{S}_w$ from truncated random walks as the set of context nodes h. Then, DeepWalk employs the Skip-gram [8] model to learn node representations, which are essential to predicting the context nodes of each anchor by maximizing the log-likelihood function as follows:

$$\max \sum_{w \in \mathcal{V}} \sum_{h \in \mathcal{S}_w} \log\left(p(h|w)\right), \tag{1}$$

where w denotes the anchor and h denotes the context nodes, and $p(h|w)$ is a conditional probability of context node h given the anchor w.

To model the conditional probability $p(h|w)$, we introduce an encoder $g_\theta : v_i \to \mathbf{z}_i \in \mathbb{R}^d$ that maps a node to a d-dimensional representation, where θ denotes parameters of the encoder and $\mathbf{z}_i$ denotes the node embedding of node v_i. For instance, $g_\theta(w) = \mathbf{z}_w$ denotes that the embedding vector of the anchor node w. Generally, the encoder is implemented as an embedding lookup in the network embedding methods [1,7,11], while in the GNN methods it is a message-passing based neural network model [5,6]. With those node representations, a scoring function $s_\theta : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is employed to quantify the compatibility between the context nodes and the anchor node. For instance, given an anchor node w and its context nodes h, the score between them is denoted as $s_\theta(h, w)$. Generally, the scoring function is instanced as the inner product of the node embeddings, e.g., $s_\theta(h, w) = g_\theta(h)^\top g_\theta(w) = \mathbf{z}_h^\top \mathbf{z}_w$. In terms of modeling node representations, the conditional distribution corresponding to anchor w, $P_\theta(h|w)$, is defined as:

$$p_\theta(h|w) = \frac{\exp\left(s_\theta(h, w)\right)}{\sum_{h' \in \mathcal{V}} \exp\left(s_\theta(h', w)\right)}. \tag{2}$$

4. Theoretical Insights

In this section, we introduce noise contrastive estimation (NCE) in Section 4.1 and its simplified version customized for graph representation learning, called negative sampling (NEG), in Section 4.2. In Section 4.3, we prove that NEG essentially adopts uniform sampling over a static marginal distribution and that sampling over an adaptive conditional distribution is crucial for representation learning. Finally, we show theoretically the principle of the negative sampling strategy in Section 4.4.

4.1. Noise Contrastive Estimation

Unfortunately, computing $p_\theta(h|w)$ of Eq. (2) requires normalizing the entire node set $\mathcal{V}$, which means the model training takes time linearly in the size of nodes $|\mathcal{V}|$. Thus, as a computationally efficient approximation of the representation learning model, Noise Contrastive Estimation (NCE) is a stable and efficient alternative solution [9]. The intuition is that converting the probabilistic density estimation to the probabilistic binary classification problem, discriminating between samples from the original data distribution or noise distribution. For instance, given some context nodes h sampled from the data distribution $P_d(h|w)$ and negative nodes from a noise distribution $P_n(h)$, the optimization objective function for NCE is

$$\begin{aligned}\mathcal{J}_{NCE}^{w}(\theta) = &\ \mathbb{E}_{P_d(h|w)}[\log(\sigma(\Delta s_\theta(h, w)))] \\ &+ k\mathbb{E}_{P_n(h)}[\log(1 - \sigma(\Delta s_\theta(h, w)))]\end{aligned} \tag{3}$$

where $\Delta s_\theta(h, w) = s_\theta(h, w) - \log(kP_n(h))$ denotes the difference in the scores of the context node h under the training model θ and the noise distribution $P_n(h)$. The scaling coefficient k denotes that the number of samples from the noise distribution is k times more frequent than those from the data distribution. Moreover, $\sigma(\cdot)$ denotes the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$. With DeepWalk as an instance, the context nodes h are sampled from the truncated random walk sequence ($P_d(h|w)$), while the negative nodes h' are sampled from the node-degree distribution raised to the 3/4th power ($P_n(h)$). The objective is to fit the conditional probability model $P_\theta(h|w)$ of Eq. (2) to $P_d(h|w)$ [7].

4.2. Negative Sampling

NCE can be considered as to approximately maximize the log likelihood of Eq. (1), while graph representation learning model is only concerned with the high-quality node representations [36]. Thus, while ensuring the high quality of node representations, NCE can be further simplified by omitting the numerical probability of the noise distribution and holding merely the negative samples, termed as Negative sampling, or NEG [7,8,36]. Next, we briefly recall the objective of NEG:

$$\mathcal{J}_{NEG}^{w} = \mathbb{E}_{h\sim P_d(h|w)}[\log\sigma(\mathbf{z}_h^{\top}\mathbf{z}_w)] + \sum_{j=1}^{k}\mathbb{E}_{h'\sim P_n(h')}[\log\sigma(-\mathbf{z}_{h'}^{\top}\mathbf{z}_w)], \tag{4}$$

where $h\sim P_d(h|w)$ denotes that sampling context nodes h from the data distribution $P_d(h|w)$, $h'\sim P_n(h')$ denotes that sampling negative nodes h' from the noise distribution $P_d(h')$, and $\mathbf{z}_h^{\top}\mathbf{z}_w$ denotes the inner product of two node embeddings. Thus, the task is converted to distinguish between the context nodes h and the negative samples h' drawn from the noise distribution $P_n(h')$. In particular, the coefficient k denotes sampling k negative nodes for each context node.

4.3. From NCE to NEG

In NCE, the negative nodes are sampled i.i.d. from the static marginal distribution $P_n(h')$. Indeed, NEG, as the simplification, has the same assumption and further theoretically assumes that $P_n(h')$ is a uniform distribution. The next theorem shows that NCE can be mathematically converted to NEG with a uniform sampling strategy.

Theorem 1. *Let Eqs. (3) and (4) be the objective functions of NCE and NEG, respectively, where $P_n(\cdot)$ denotes the negative noise distribution and k denotes the scaling coefficient, and $|\mathcal{V}|$ be the size of the node set $\mathcal{V}$. NEG is a particular case of NCE, if $k=|\mathcal{V}|$ and $P_n(\cdot)$ is the uniform distribution.*

Proof of Theorem 1 is given in Appendix A. According to Theorem 1, we know that NEG is mathematically a special case of NCE with the uniform sampling strategy over the static marginal distribution.

However, uniformly sampling negatives from static marginal distribution $P_n(h')$ may not the optimal manner for learning the node representations. For instance, prior work has shown the effectiveness of dynamic hard negatives in the information retrieval field [30,31,33]. In this work, we similarly explore sampling negative nodes from the conditional distribution on the current training state.

Suppose we define a noise distribution $P_n(h|w)$ conditioned on the representation of the anchor node w under the current training state. Then, we can rewrite the objective function of NEG as follows:

$$\mathcal{J}_{NEG}^{w} = \mathbb{E}_{h\sim P_d(h|w)}[\log\sigma(\mathbf{z}_h^{\top}\mathbf{z}_w)] + \sum_{j=1}^{k}\mathbb{E}_{h'\sim P_n(h'|w)}[\log\sigma(-\mathbf{z}_{h'}^{\top}\mathbf{z}_w)]. \tag{5}$$

Next, to identify the importance of adaptive negative sampling on the conditional distribution, we show the optimization objective for node representation on NEG as follows:

Theorem 2. *Let Eq. (5) be the objective function of NEG, where $P_d(h|w)$ denotes the positive data distribution and $P_n(h|w)$ denotes the negative noise distribution, and $s_\theta(h,w)$ be the scoring function. For each pair of anchor node w and context node h, the optimal representation vectors satisfy:*

$$s_\theta(h,w) = -\log\frac{kP_n(h|w)}{P_d(h|w)}. \tag{6}$$

Proof of Theorem 2 is given in Appendix B. According to Theorem 2, we can clearly identify that sampling negative examples conditioned on the node representations in the current training state has the same importance as positive samples. Indeed, the adaptive conditional distribution may significantly facilitate the optimization of the node representations more than the static marginal distribution.

4.4. The Principle of Negative Sampling Strategy

After determining the importance of adaptively sampling negative examples, the following query is raised: *how to specify an effective adaptive negative sampling distribution?* In response, we demonstrate that the vanishing gradient problem can be solved and convergence can be sped up by adaptively sampling hard negatives from a stochastic gradient descent (SGD) optimization perspective. In this case, **Hard negatives** are samples with higher scores or samples located closer to the anchor in the embedding space. Intuitively, discriminating similar samples can provide more information to the model than random samples, hence speeding up the optimization.

The gradient of objective function of NEG is:

$$\frac{\partial\mathcal{J}_{NEG}}{\partial\theta} = (\sigma(\mathbf{z}_h{}^{\top}\mathbf{z}_w)-1)\frac{\partial\mathbf{z}_h{}^{\top}\mathbf{z}_w}{\partial\theta} + k\sum_{h'}\sigma(\mathbf{z}_{h'}{}^{\top}\mathbf{z}_w)\frac{\partial\mathbf{z}_{h'}{}^{\top}\mathbf{z}_w}{\partial\theta}, \tag{7}$$

where θ denotes the parameters of the encoder model, e.g., node embedding lookup.

Given a set of (w,h,h') uniformly sampled in a batch, the stochastic gradient descent step is performed as follows:

$$\begin{aligned}\mathbf{z}_h^{new} &\leftarrow \mathbf{z}_h^{old} - \eta(\sigma(\mathbf{z}_h^{\top}\mathbf{z}_w)-1)\mathbf{z}_w,\\ \mathbf{z}_{h'}^{new} &\leftarrow \mathbf{z}_{h'}^{old} - \eta(\sigma(\mathbf{z}_{h'}^{\top}\mathbf{z}_w))\mathbf{z}_w,\end{aligned} \tag{8}$$

where η is the learning rate. The representation learning model is optimized by looping over the above Eq. (8). We can notice that the gradient magnitude of the embedding for the negative sample is dependent on the score (e.g., inner product) with the embedding of the anchor point $\sigma(\mathbf{z}_{h'}^{\top}\mathbf{z}_w)$. It is obvious that if $\sigma(\mathbf{z}_{h'}^{\top}\mathbf{z}_w)$ is close to 0, nothing can be learned from the sampled case (w,h') due to its vanishing gradient. Therefore, to alleviate the vanishing gradient problem and speed up the model updating, nodes with higher scores should be sampled with greater probability. This motivates that sampling the hard examples h' to the anchor w. It is worth noting that the score depends on the current model parameters θ, and thus the negative sampling distribution is adaptive and dynamic in the learning process. Formally, given the anchor node w, we define the adaptive negative sampling distribution as follows:

$$P_n(h|w) = \frac{\mathbf{z}_h{}^{\top}\mathbf{z}_w}{\sum_{h'\in\mathcal{V}}\mathbf{z}_{h'}{}^{\top}\mathbf{z}_w}. \tag{9}$$

5. AdaNS: A New Strategy

With the theoretical findings above, we propose an efficient adaptive negative sampling strategy under the noise contrastive estimation framework for unsupervised graph representation learning.

5.1. An Efficient Adaptive Negative Sampling Strategy

We deduce that sampled negative examples should be hard negatives close to the anchor in Eq. (9), however, each sampling involves calculating all examples and requires $O(|\mathcal{V}|\cdot d)$ time. To efficiently sample negative examples, in this work, we propose an approximate negative sampling strategy that formalizes the negative sampling distribution as a mixing distribution on dimensions.

Firstly, let the inner products in the scoring function above be a matrix factorization as follows:

$$\mathbf{z}_h^{\top}\mathbf{z}_w = \sum_{f=1}^{d}\mathbf{z}_{h,f}\mathbf{z}_{w,f}, \tag{10}$$

where d is the dimension of node embeddings. The negative sampling distribution is defined as follows:

$$
\begin{aligned}
P_n(h|w) &= \frac{\sum_{f=1}^{d} \mathbf{z}_{h,f}\mathbf{z}_{w,f}}{\sum_{h'\in\mathcal{V}} \sum_{f=1}^{d} \mathbf{z}_{h',f}\mathbf{z}_{w,f}} \\
&= \frac{\sum_{f=1}^{d} \mathbf{z}_{h,f}\mathbf{z}_{w,f}}{\sum_{f=1}^{d} \sum_{h'\in\mathcal{V}} \mathbf{z}_{h',f}\mathbf{z}_{w,f}} \\
&= \sum_{f=1}^{d} \frac{\mathbf{z}_{h,f}\mathbf{z}_{w,f}}{\sum_{h'\in\mathcal{V}} \mathbf{z}_{h',f}\mathbf{z}_{w,f}} \\
&= \sum_{f=1}^{d} \left|\mathbf{z}_{w,f}\right| \mathrm{sgn}(\mathbf{z}_{w,f}) \frac{\mathbf{z}_{h,f}}{\sum_{h'\in\mathcal{V}} \mathbf{z}_{h',f}},
\end{aligned}
\tag{11}
$$

where $\mathrm{sgn}(\cdot)$ denotes the sign of the scalar.

Assume that node embedding vectors for each dimension f follow the normal distribution[1], the sampling probability can be defined as the mixing distribution as follows:

$$
P_n(h|w) = \sum_{f=1}^{d} p(f|w)p(h|w,f), \tag{12}
$$

where $p(f|w)$ denotes the importance of the dimension f for embedding vector of node w.

Following Eqs. (11) and (12), the probability function can be defined as follows:

$$
p(f|w) \propto \left|\mathbf{z}_{w,f}\right|, \tag{13}
$$

$$
p(h|w,f) = \mathrm{sgn}(\mathbf{z}_{w,f}) \frac{\mathbf{z}_{h,f}}{\sum_{h'\in\mathcal{V}} \mathbf{z}_{h',f}}. \tag{14}
$$

It worth noting that the elements of node vectors can be negative. To feasibly calculate the probability function, $p(h|w,f)$ can be further defined as follows:

$$
p(h|w,f) = \sigma(\mathrm{sgn}(\mathbf{z}_{w,f})\mathbf{z}_{h,f}) = \frac{\exp(\mathrm{sgn}(\mathbf{z}_{w,f})\mathbf{z}_{h,f})}{\sum_{h'\in\mathcal{V}} \exp(sgn(\mathbf{z}_{w,f})\mathbf{z}_{h',f})}. \tag{15}
$$

5.2. *The Proposed AdaNS Algorithm*

We summarize the sampling process from the mixing distribution, and the detailed pseudocode is shown in Algorithm 1.

Algorithm 1 The training process of AdaNS.

```
1: Initialize parameter of Encoder $\theta$
2: repeat
3:     for each node $w$ do
4:         Draw a positive sample $h$ from $P_d(h|w)$
5:         Draw a dimension factor $f$ from $P(f|w) \propto |\mathbf{z}_{w,f}|$
6:         Draw a negative sample $h'$ from$P(h'|w,f)$ (Eq. (15))
7:         Update $\theta$ by Stochastic Gradient Descent (Eq. (8))
8: until Early Stopping or Maximum of Epoch
```

6. Experiments

To comprehensively assess the proposed negative sampling strategy, we conduct both node classification and node visualization on seven commonly used graph datasets. We apply our proposed strategy and seven negative sampling baselines to representative shallow network embedding and GNNs embedding models, e.g., DeepWalk [7] and GraphSAGE [5], to learn node representations. The desired node representation would preserve more information, thereby achieving a higher classification performance and a more distinguishable node visualization.

[1] In practice, the node vectors for each dimension follow the standard normal distribution.

Table 2
Statistics of datasets.

Task	Datasets	#Nodes	#Edges	#Labels	#Features
Shallow Network Embedding	Cora-WF	2,708	5,429	7	–
	Wiki	2,405	15,985	19	–
	PPI	3,890	76,584	50	–
	BlogCatalog	10,312	333,983	39	–
GNNs Embedding	Cora	2,708	5,429	7	1,433
	CiteSeer	3,327	4,732	6	3,703
	PubMed	19,717	44,338	3	500

6.1. *Datasets*

We evaluate the proposed negative sampling strategy with seven public graph datasets applied to two embedding tasks, respectively. Specifically, we use four graph datasets that have no features to learn node embeddings with preserved graph structure. In addition, we use three graph datasets containing both features and graph structure to deploy GNNs-based node embedding tasks. For clarity, we summarize the statistics of seven datasets in Table 2 and describe in detail the characteristics of each dataset as follows:

We first describe four datasets without node features for shallow network embedding.

- Cora-WF [37]: Cora is a research paper set, where nodes and edges stand for machine learning papers and citation relationships, respectively. It has 2,708 papers grouped in 7 classes and 5,429 relationships between them.
- Wiki [38]: Wiki is a network of web pages with hyperlinks. It contains 2,405 web pages divided into 19 categories and 15,985 links between them.
- PPI [39]: PPI is a subgraph of the Protein-Protein Interactions network for Homo Sapiens. There are 3,890 nodes classified into 50 categories according to their biological states and 76,584 edges.
- BlogCatalog [40]: BlogCatalog is a social blogger network. It has 10,312 bloggers and 333,983 social relationships between them. Besides, 39 interested topics submitted by bloggers are considered as labels.

Then, we describe three datasets with node features for GNNs embedding, namely **Cora, Citeseer**, and **Pubmed**. These three citation networks are standard graph benchmark datasets [37]. In these datasets, nodes represent papers, and edges refer to citation relations. And the bag-of-words representations of papers are considered as node features.

6.2. *Shallow Network Embedding*

We first evaluate the proposed negative sampling strategy in the shallow network embedding model. Specifically, the shallow network embedding model is characterized by embedding lookup tables containing node embeddings as row or column vectors, which are treated as parameters and can be updated during the training process [7,11]. We use the classical model named DeepWalk [7] as the backbone model and deploy various negative sampling strategies on it. Theoretically, DeepWalk uses truncated random walks to capture context nodes and then optimizes node embedding by maximizing the co-occurrence probability of nodes with their contexts, which is measured by normalizing the scores between nodes and their contexts. In practice, the scoring is implemented as an inner product of node embeddings, and the optimization objective is to maximize the score between a node and its contexts and minimize the score between the node and its sampled negative nodes, that is, to be close to the context nodes and push apart the negatives in the embedding space. Thus, the learned node embeddings would preserve the information of the graph structure. Then, we evaluate negative sampling strategies by linear classification on the learned embeddings.

6.2.1. Baselines for DeepWalk

We include the following two groups of negative sampling strategies as baselines.

Static Negative Sampling Strategies

- RNS [28]: Random negative sampling (RNS) is one prevalent strategy to sample negative nodes with uniform distribution.
- Degree-based Negative Sampling [8]: This strategy is widely used in the field of graph representation learning. It biases the uniform distribution to the node-degree distribution raised to the 3/4rd power.

Hard-based Negative Sampling Strategies

- DNS [30]: Dynamic negative sampling (DNS) is a state-of-the-art sampling strategy for collaborative filtering, which adaptively picks the negative item scored highest by the current recommender among a randomly sampled set of unobserved items.
- WARP [31]: The weighted approximate-rank pairwise (WARP) adopts uniform sampling with rejection to draw informative negative samples, whose score should be larger than the positive one.
- KBGAN [33]: Such model is an adversarial sampler, which uniformly randomly samples N_s negative examples to calculate the probability of generating negative samples.

6.2.2. Implementation Details for DeepWalk

For fairness, the number of dimensions is set to be the same (128) for all negative sampling strategies. When sampling negative nodes, we set the number of negative nodes as 1. The size of the unobserved item set in DNS is set to 5 for all datasets. The maximal trial of sampling in WARP is set to 50 for efficiency. N_s, the number of negative items sampled in KBGAN is set to 10. The implementation program is based on Tensorflow. We train all models for a maximum of 200 epochs and use the early stopping strategy with a patience of 20 epochs. The optimizer adopts Adam to update model parameters, and the learning rate is 0.001.

In the node classification task for DeepWalk, logistic regression is adopted as a supervised classifier. In detail, we randomly select T_f from 10% to 90% fraction of the labeled nodes as the training set and the remaining nodes as the test set. The performance of the node classification is assessed by the Micro-$F1$ score and Macro-$F1$ score. The formulas of Micro-$F1$ and Macro-$F1$ are:

$$\text{Micro-}F1 = \frac{2\sum_{i=1}^{z} TP_i}{\sum_{i=1}^{z} 2TP_i + FP_i + FN_i},$$

$$\text{Macro-}F1 = \frac{1}{z}\sum_{i=1}^{z} \frac{2TP_i}{2TP_i + FP_i + FN_i},$$

where z denotes the number of labels, T, F, P, and N denote True, False, Positive, and Negative, respectively. We repeat the trial 10 times and report the average scores with different training ratios.

6.2.3. Classification Results of DeepWalk

Tables 3–6 report the performance of AdaNS in comparison to baseline strategies. Notably, the best results are shown in bold. From the results, we can draw the following observations and conclusions.

- For static samplers, the Degree-based negative sampling model achieves higher Micro-$F1$ and Macro-$F1$ results compared with RNS in most cases. It demonstrates that the Degree-based negative sampling strategy can alleviate the vanishing gradient problem caused by RNS, thereby improving the performance of the node representations.
- The adaptive samplers such as DNS, KBGAN, and WARP outperform the static samplers in most cases, suggesting that dynamically sampling the hard negatives is better than the predefined static distributions. DNS achieves performance over other baselines second only to AdaNS on Cora, Wiki, and BlogCatalog, which proves that drawing negative samples with the highest scores in the subset is beneficial to improving model performance. The performance achieved by both KBGAN is unstable, only surpasses RNS consistently, and is inferior to the Degree-based model on the Wiki and PPI datasets. This may be because KBGAN is essentially equivalent to importance sampling in subsets, and thus its performance is highly dependent on the sampling of subsets by uniform sampling. WARP outperforms RNS and Degree-based strategies in almost all cases and KBGAN in 75% of cases, but its performance is lower than that of DNS because its rejection sampling makes it difficult to sample matching nodes before the patient round after the model has been trained to a certain level, which hinders its further performance improvement.
- AdaNS achieves more satisfactory performance over baselines, especially on the BlogCatalog, where AdaNS consistently outperforms all baselines regardless of training set ratios as well as metrics. It indicates that our proposed mixing distribution sampling is more effective than existing adaptive samplers and static samplers.

6.3. GNNs Embedding

Different from the shallow network embedding models, GNNs embedding models can utilize graph features more effective. The basic idea of GNNs is iteratively aggregating node feature information from the neighborhood to yield continuous smoothing node embeddings over the graph structure, which is the message-passing framework [2,5,6]. In this experiment, we adopt the GraphSAGE [2] as the backbone model due to its two advantages: 1) It captures contextual nodes by sampling the neighborhood, which is more efficient than the classical message-passing scheme; 2) it has both unsupervised learning and supervised learning training implementations, which will yield more intuitive node visualization results.

It is noting that any current SOTA GNN methods can be used as the encoder in the unsupervised contrastive paradigm, just like GraphSAGE, thus AdaNS can improve the overall model equipped with any elaborate message-passing scheme.

6.3.1. Baselines for GraphSAGE

For baselines, beyond those mentioned above, we further add two following semi-hard strategies:

- InterCLR [34]: InterCLR presents a semi-hard negative sampling, which first samples a pool with the top 10% most similar example and then randomly draws negatives from the pool.
- Ring [35]: Ring argues that the most similar examples might be better suited as positive examples rather than negative ones. Therefore, it chooses fairly similar examples, but not too hard ones, as negatives.

Table 3
Node classification results of DeepWalk on Cora.

Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0. 6103	0. 6880	0. 7099	0. 7305	0. 7312	0. 7399	0. 7540	0. 7454	0. 7565
	RNS	0. 6354	0. 6908	0. 7157	0. 7262	0. 7201	0. 7269	0. 7466	0. 7565	0. 7565
Micro-$F1$	DNS	0. 6920	0. 7310	0. 7511	0. 7588	0. 7696	0. 7694	0. 7835	0. 7768	0. 8007
	KBGAN	0. 6505	0. 7084	0. 7215	0. 7317	0. 7341	0. 7380	0. 7417	0. 7399	0. 7528
	WARP	0. 6707	0. 7116	0. 7409	0. 7480	0. 7631	0. 7648	0. 7586	0. 7556	0. 7648
	AdaNS	**0. 6924**	**0. 7476**	**0. 7574**	**0. 7717**	**0.7792**	**0. 7749**	**0.7872**	**0.7970**	**0.8229**
Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0. 5756	0. 6736	0. 6904	0. 7091	0. 7119	0. 7306	0. 7372	0. 7432	0. 7329
	RNS	0. 6185	0. 6828	0. 7056	0. 7135	0. 7104	0. 7093	0. 7326	0. 7377	0. 7091
Macro-$F1$	DNS	**0.6755**	0. 7197	0. 7383	0. 7484	0. 7576	0. 7584	0. 7673	0. 7612	0. 7717
	KBGAN	0. 6285	0. 6971	0. 7077	0. 7217	0. 7234	0. 7241	0. 7315	0. 7300	0. 7109
	WARP	0. 6532	0. 7028	0. 7286	0. 7417	0. 7508	0. 7555	0. 7455	0. 7435	0. 7556
	AdaNS	0. 6754	**0.7368**	**0. 7471**	**0.7607**	**0.7663**	**0. 7644**	**0.7801**	**0.7795**	**0.7898**

Table 4
Node classification results of DeepWalk on Wiki.

Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0. 5630	0. 5962	0. 6105	0. 6334	0. 6509	0. 6632	0. 6648	0. 6694	0. 6349
	RNS	0. 5303	0. 5998	0. 6188	0. 6202	0. 6467	0. 6414	0. 6371	0. 6528	0. 6390
Micro-$F1$	DNS	0. 5557	0. 6107	0. 6366	0. 6542	0. 6717	0. 6663	0. 6787	0. 6861	0. 6390
	KBGAN	0. 5644	0. 6081	0. 6229	0. 6286	0. 6517	0. 6486	0. 6537	0. 6549	0. 6100
	WARP	0. 5636	0. 6072	0. 6257	0. 6324	0. 6585	0. 6662	0. 6804	0. 6825	0. 6115
	AdaNS	**0.5686**	**0.6201**	**0. 6449**	**0. 6694**	**0.6733**	**0. 6684**	**0. 6814**	**0.6882**	**0.6681**
Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0. 3975	0. 4740	0. 5084	0. 5105	0. 5322	0. 5261	0. 5634	0. 5730	0. 5275
	RNS	0. 3847	0. 4559	0. 5043	0. 5179	0. 5349	0. 5264	0. 5381	0. 5326	0. 5341
Macro-$F1$	DNS	0. 4206	**0. 5010**	**0.5362**	0. 5350	0. 5554	**0. 5634**	0. 5773	0. 5940	0. 5414
	KBGAN	**0. 4293**	0. 4774	0. 5113	0. 5197	0. 5442	0. 5263	0. 5397	0. 5308	0. 5003
	WARP	0. 4262	0. 4839	0. 5194	0. 5290	0. 5304	0. 5419	0. 5508	0. 5676	0. 5275
	AdaNS	0. 4030	0. 4663	0. 5338	**0.5659**	**0.5586**	0. 5497	**0.5836**	**0.5961**	**0. 5564**

Table 5
Node classification results of DeepWalk on PPI.

Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0.1668	0.1802	0.1976	0.1994	0. 2141	0. 2101	0. 2149	0. 2229	**0. 2496**
	RNS	0.1595	0.1753	0.1882	0.1934	0. 2010	0. 2041	0. 2105	0. 2148	0. 2237
Micro-$F1$	DNS	0.1691	**0.1994**	0.1966	0. 2012	0. 2108	0. 2214	0. 2221	0. 2226	0. 2237
	KBGAN	0.1702	0.1791	0.1886	0.1904	0. 2043	0. 2082	0. 2193	0. 2185	0. 2323
	WARP	0.1733	0.1951	0.1959	0.1977	0.1996	0. 2058	0. 2248	0. 2225	0. 2338
	AdaNS	**0.1803**	0.1988	**0.2085**	**0. 2197**	**0.2255**	**0. 2315**	**0.2379**	**0.2331**	0. 2439
Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0.1228	0.1396	0.1567	0.1633	**0.1756**	0.1750	0.1736	0.1755	**0.1982**
	RNS	0.1207	0.1363	0.1464	0.1544	0.1637	0.1703	0.1739	0.1713	0.1905
Macro-$F1$	DNS	0.1197	0.1393	0.1401	0.1455	0.1499	0.1554	0.1573	0.1579	0.1594
	KBGAN	0.1269	0.1405	0.1493	0.1563	0.1707	0.1773	0.1762	0.1735	0.1808
	WARP	0.1318	0.1345	0.1455	0.1476	0.1511	0.1564	0.1729	0.1747	0.1775
	AdaNS	**0.1340**	**0.1471**	**0.1607**	**0.1745**	0.1754	**0.1843**	**0.1849**	**0.1786**	0.1901

Table 6
Node classification results of DeepWalk on BlogCatalog.

Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0. 2911	0. 3246	0. 3427	0. 3557	0. 3635	0. 3661	0. 3754	0. 3825	0. 4018
	RNS	0. 3026	0. 3289	0. 3483	0. 3556	0. 3589	0. 3628	0. 3680	0. 3781	0. 3762
Micro-$F1$	DNS	0. 3494	0. 3743	0. 3845	0. 3888	0. 3876	0. 3971	0. 3981	0. 4071	0. 4118
	KBGAN	0. 2911	0. 3261	0. 3470	0. 3599	0. 3612	0. 3720	0. 3790	0. 3911	0. 4031
	WARP	0. 2906	0. 3220	0. 3471	0. 3532	0. 3578	0. 3802	0. 3896	0. 4015	0. 4053
	AdaNS	**0.3576**	**0. 3819**	**0.3880**	**0.3957**	**0.3986**	**0. 4075**	**0. 4167**	**0. 4279**	**0. 4280**
Measure	Strategies	10%	20%	30%	40%	50%	60%	70%	80%	90%
	Degree	0.1677	0.1968	0. 2115	0. 2381	0. 2312	0. 2306	0. 2401	0. 2416	0. 2755
	RNS	0.1702	0.1975	0. 2200	0. 2229	0. 2237	0. 2300	0. 2492	0. 2538	0. 2633
Macro-$F1$	DNS	0.1869	0. 2193	0. 2322	0. 2409	0. 2393	0. 2517	0. 2518	0. 2701	0. 2847
	KBGAN	0.1693	0.1999	0. 2138	0. 2221	0. 2248	0. 2323	0. 2394	0. 2595	0. 2793
	WARP	0.1915	0.1988	0. 2242	0. 2408	0. 2382	0. 2383	0. 2470	0. 2638	0. 2848
	AdaNS	**0. 2017**	**0.2367**	**0. 2449**	**0. 2473**	**0. 2477**	**0. 2649**	**0.2766**	**0. 2913**	**0.2930**

Table 7
The classification accuracy results of GraphSAGE with various negative sampling strategies.

Strategies	Cora			Citeseer			Pubmed		
	full	random	public	full	random	public	full	random	public
Degree	0.815	0.75	0.768	0.694	0.613	0.615	0.821	0.755	0.778
RNS	0.799	0.747	0.749	0.69	0.592	0.639	0.817	0.724	0.758
DNS	0.779	0.754	0.739	0.674	0.589	0.588	0.815	0.745	0.751
KBGAN	0.797	0.759	0.744	0.664	0.584	0.589	0.822	0.755	0.767
InterCLR	0.819	0.769	0.762	0.689	0.61	0.621	0.822	0.756	0.788
Ring	0.817	0.756	0.775	0.69	0.628	0.65	0.824	**0.77**	0.784
AdaNS	**0.833**	**0.78**	**0.779**	**0.704**	**0.633**	**0.653**	**0.826**	0.76	**0.792**

6.3.2. Implementation Details for GraphSAGE

We implement the negative sampling strategies on top of the unsupervised GraphSAGE [5]. For GraphSAGE, the mean-aggregator is used in our experiments. We set the dimension of the hidden layer as 64. In training, we use SGD with a learning rate of 0.01 and no weight decay for 100 epochs and batch size 256. For Ring, we set the upper percent as 90%. For three citation network datasets, we use the default three splits, namely full, random, and public [41]. We evaluate the performance of models by node classification accuracy.

6.3.3. Classification Results of GraphSAGE

We report the classification results of GraphSAGE with various negative sampling strategies in Table 7. The best results in each setting are shown in bold. From the results, we can obtain the following observations.

- For two static negative sampling strategies, namely Degree and RNS, the results are lower than most semi-hard-based strategies but higher than hard-based ones. Specifically, the performance of Degree outperforms RNS in eight of nine settings, which implies that the node-degree-based negative sampling strategy benefits node representation learning in the GNNs model.
- We can observe that two semi-hard-based strategies, namely InterCLR and Ring, are superior to two hard-based strategies, namely DNS and KBGAN. This phenomenon is attributed to the fact that the message-passing scheme in GNNs constrains the smoothness of the neighboring nodes, making their representations more similar. Since GNNs follow the assumption of homogeneity [42], i.e., neighboring nodes are more likely to belong to the same class, the most similar nodes, or the hardest nodes, are more likely to be positive examples, which deviates from the hard-based negative sampling strategy. To verify the above, we study the impact of the message-passing scheme in GNNs on the hard-based negative sampling strategies. Specifically, we adopt node embeddings learned on DeepWalk and GraphSAGE with 50 epochs and then choose the 100 most similar nodes for each anchor, where the nodes with the same class as the anchor are regarded as positive samples. Finally, we calculate the frequency of positive samples for each node. As shown in Fig. 1, we plot the histograms of positive sample frequencies of DeepWalk or GraphSAGE on Cora. We can observe that the node embeddings learned by GraphSAGE based on the message-passing scheme significantly increase the frequency of positive samples among similar nodes compared to DeepWalk, which accordingly means that the negative sampled on hard-based strategies are more likely false negative samples. Therefore, appropriately relaxing the hardness, such as by using the strategy based on semi-hard, achieves superior classification performance.
- Our proposed strategy adaptively samples negatives from the mixing distribution, which enables our strategy to sample hard negative examples. Moreover, our strategy focuses on only some of the dimensional elements, which can be considered a hardness relaxation, so our proposed strategy can also be viewed as a semi-hard sampling strategy. Such a property facilitates the node classification in GNNs, and thus our proposed AdaNS achieves superior classification performance in eight of nine settings.

6.4. Graph Visualization

To evaluate the qualities of node representations, visualization is the most common task. In the visualization task, we employ the t-distributed stochastic neighbor embedding (t-SNE) [43], a nonlinear dimension reduction and visualization approach, to transform the node representations into a 2-dimensional space.

First, we aim to evaluate the impact of different negative sampling strategies on the discriminability of node representations. Specifically, we deploy different negative sampling strategies on top of the GraphSAGE model to learn node representations on the Cora dataset. Furthermore, to improve the quality of visualization, we adopt a semi-supervised GraphSAGE in this experiment, which includes both supervised loss and unsupervised loss to train the model jointly. It is worth noting that since InterCLR can theoretically be considered as a special case of Ring [35], and the visualization of node representations learned by InterCLR is very similar to that learned by Ring, we exhibit them jointly in a view. The visualization results are shown in Fig. 2, where different colors denote nodes in different categories, and the symbol ”x” denotes the cluster centroid of each category. The centroid can take into account the nodes that deviate from the cluster, thus measuring the discriminability of the nodes globally. Geometrically, a larger region enclosed by cluster centroids indicates better discriminability. We plot the geometrically enclosed region of AdaNS in red and those of the baselines in black. We can observe that the area of the geometrically enclosed region generated by our proposed AdaNS exceeds all baselines. Numerically, we measure the discriminability of node representations via the mean distance between the pairwise cluster centroids, where a larger distance indicates that the node representations preserve better discriminability, and vice versa indicates lower discriminability, namely, that nodes of different classes tend to be mixed. We annotate the numerical results of the centroid distance, and the results show that the node representation generated by AdaNS has the largest value, which confirms the superiority of AdaNS.

Next, we study the sampling effects in practice using various sampling strategies. Specifically, we randomly sample a batch of 100 nodes from the Cora dataset. Given the anchor node, we use various sampling strategies to draw 10 negative samples respectively. A visualization of the sampling results in the embedding space is shown in Fig. 3. It is worth noting that in the embedding space, close nodes indicate similarity. We can find that both Degree and RNS statically sample negative nodes approximately at random within the whole batch. In contrast, DNS clearly tends to select the closest nodes, and KBGAN can be seen as a relaxed variant of DNS, which has the potential to select the more distant nodes as neg-

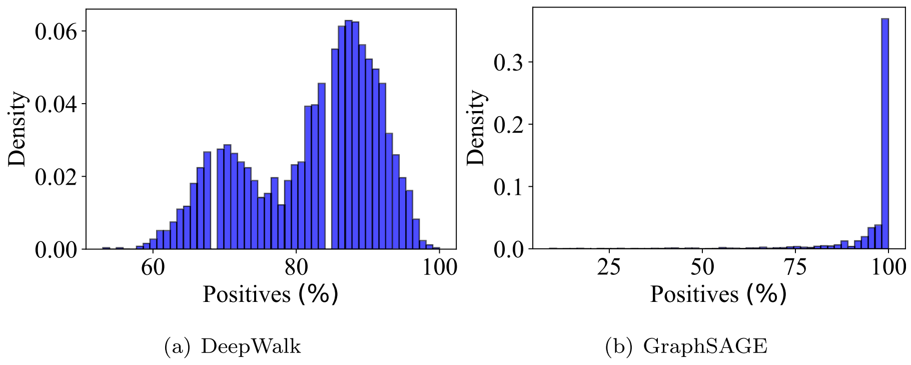


Fig. 1. Positive samples histograms of models with DeepWalk or GraphSAGE as encoders on Cora.

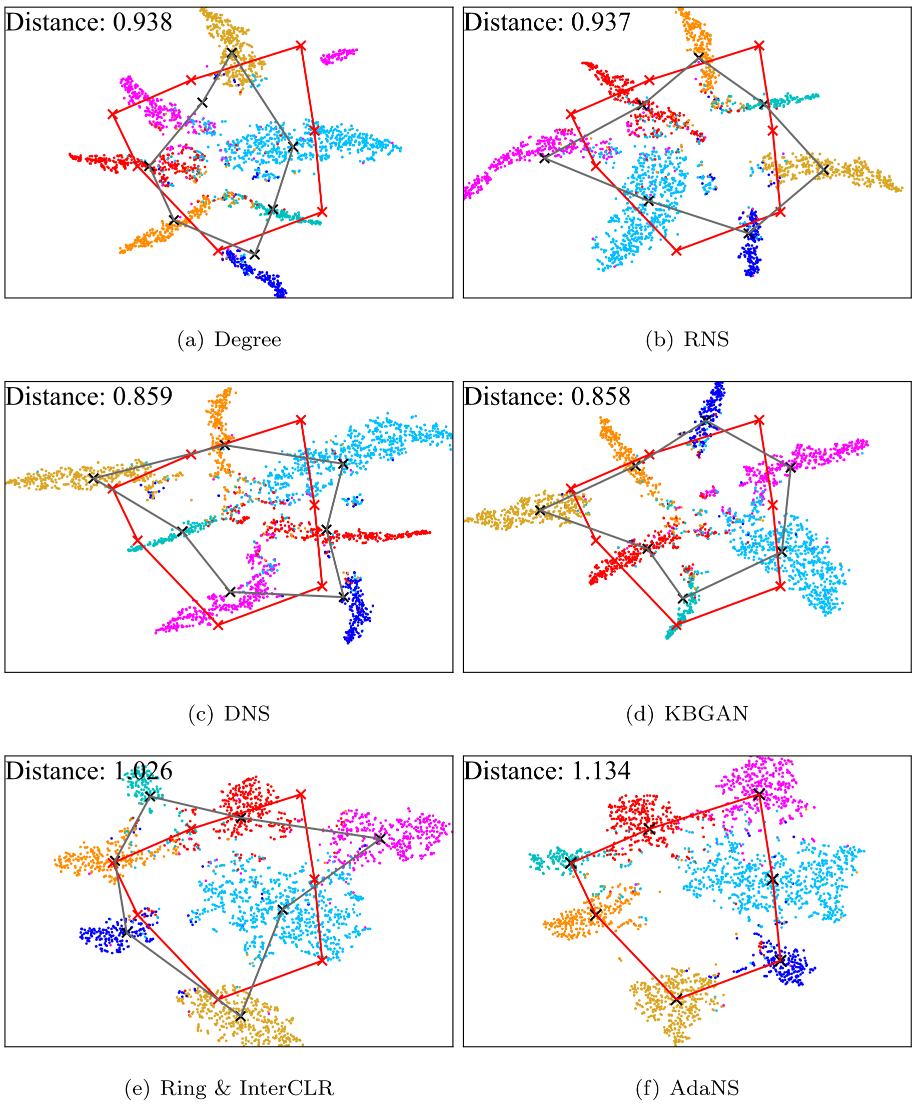


Fig. 2. Visualization of node representations on Cora dataset. Different colors denote different categories of nodes.

ative samples. While Ring, InterCLR, and AdaNS tend to select the nodes that are relatively closest, but at a certain distance.

6.5. Study and Analysis

6.5.1. Classification Quality

Figure 4 presents the classification quality as a function of training epochs. We conduct experiments on top of DeepWalk and GraphSAGE, on Cora dataset with {10%, 50%, 90%} training set and three splits, respectively. As shown in Fig. 4(a), (c) and (e), the performance on DeepWalk with adaptive strategies such as DNS, WARP, KBGAN, and AdaNS, are drastically increased in the early training process compared to the static strategies. This demonstrates that sampling hard negatives improve both the effectiveness and efficiency of the model. Besides, we observe that DNS and WARP, after peaking in performance, begin to degrade as they

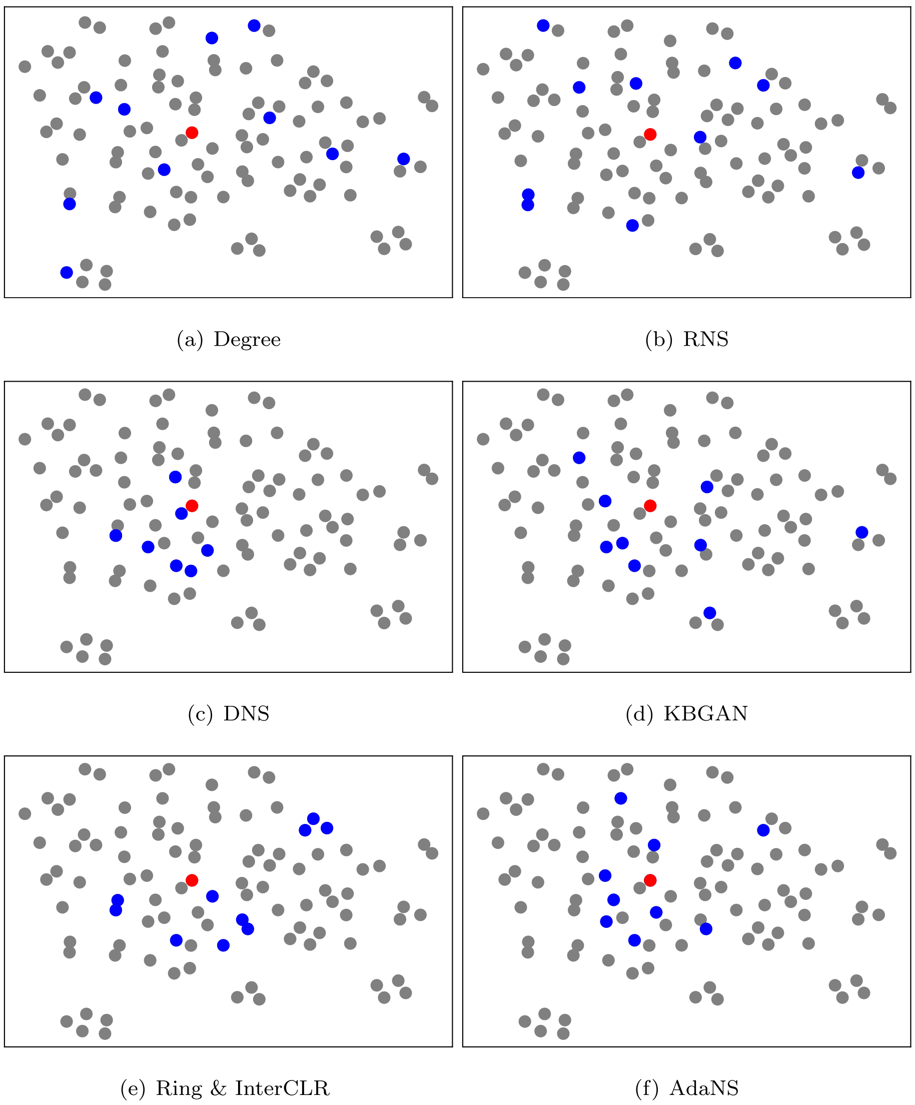


Fig. 3. Visualization of various negative sampling strategies in the embedding space. Given an anchor node (red dot), we draw negative samples (blue dots) with different strategies.

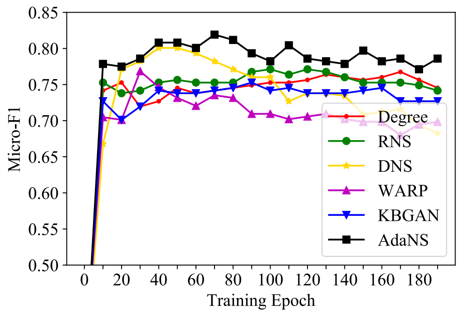
are over-trained. In contrast, our proposed AdaNS, after efficiently achieving optimal performance, remains in a stable state as training proceeds. As for GraphSAGE in Fig. 4(b), (d) and (f), due to the message-passing scheme, neighboring nodes that tend to belong to the same category naturally have high similarity, so most of the strategies achieve good accuracy at the early stage of training, where the adaptive strategies still show the higher ceiling. And as the training proceeds, the strategies based on static distribution, i.e., Degree and RNS, exhibit lackluster performance. In particular, DNS still declines sharply after reaching peak performance, even earlier than on DeepWalk, due to the smoothness of the message-passing scheme in GNNs. In contrast, semi-hard-based strategies achieve more stable performance.

6.5.2. Training Loss

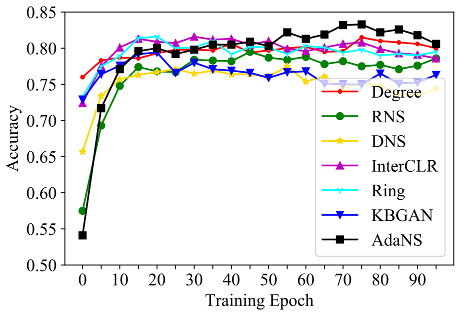
To investigate the effect of adaptive negative sampling on alleviating vanishing gradient, we experimentally record the training loss as a function of training epochs for various negative sampling strategies on the Cora dataset. As shown in Fig. 5, we can observe that the adaptive sampling strategy can better optimize the training loss compared to the two static sampling strategies, Degree and RNS. Taking AdaNS as a benchmark, we can see the obvious fluctuation of DNS, which is due to its hard-based sampling strategy that mistakenly selects positive samples as negative samples during training. In contrast, KBGAN, InterCLR, and Ring exhibit more stable training losses. In particular, in Fig. 5(a), we can observe that the training loss of WARP decreases slowly since its negative nodes sampled must be larger than the positive ones, which inevitably sample a lot of false negatives. In contrast, DNS consistently samples the hardest negative nodes, resulting in a rapid and continuous decrease in training loss. In conclusion, adaptive negative sampling strategies are able to obtain lower training losses faster and more consistently, which demonstrates the capability to mitigate the vanishing gradient problem.

6.5.3. Efficiency Analysis

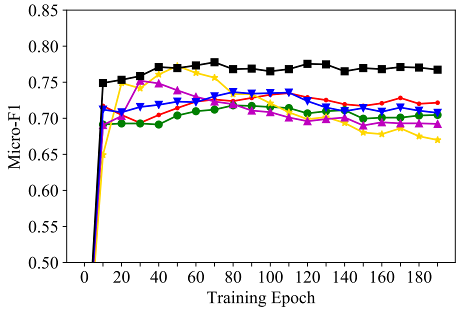
Adaptive sampling is time-consuming in comparison to static sampling while improving performance, hence the efficiency of sampling negatives is critical. The average running time per epoch for adaptive strategies is summarized in Fig. 6. As the rejection mechanism of WARP increases in difficulty as training progresses, we adopt the average running time per epoch for a fair comparison. From the figure, we can find that the running time of AdaNS is the least, while WARP takes the most running time, due to its



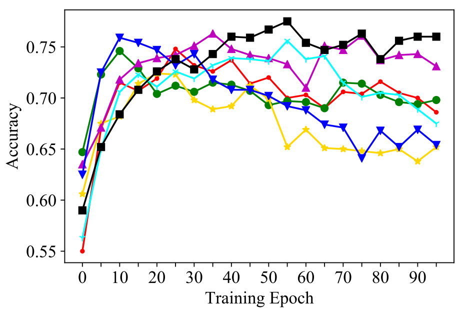
(a) DeepWalk-Cora-0.9



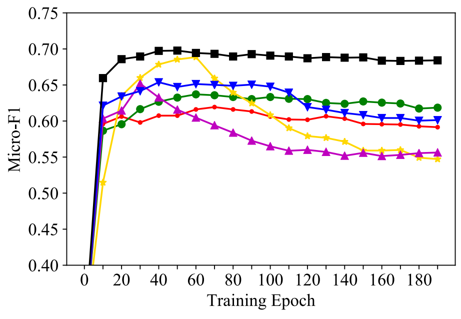
(b) GraphSAGE-Cora-full



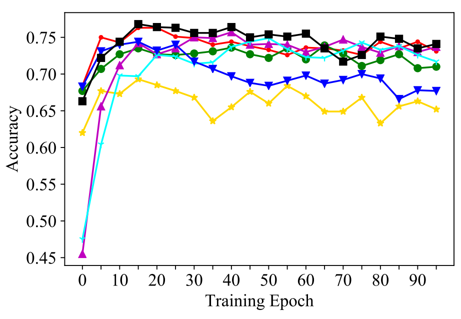
(c) DeepWalk-Cora-0.5



(d) GraphSAGE-Cora-random

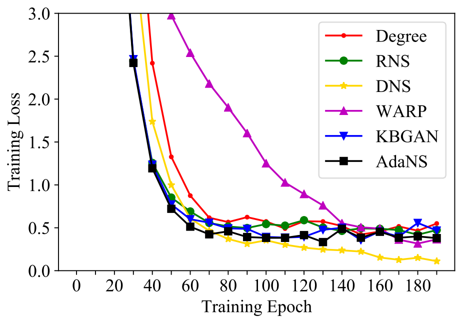


(e) DeepWalk-Cora-0.1

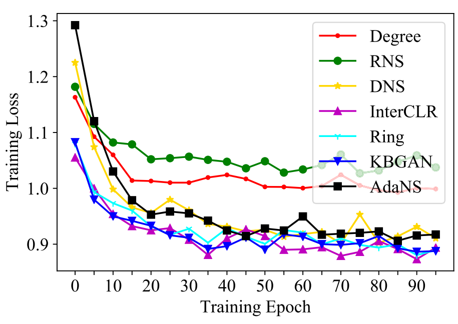


(f) GraphSAGE-Cora-public

Fig. 4. The classification quality as a function of training epochs.



(a) DeepWalk



(b) GraphSAGE

Fig. 5. The training Loss as a function of training epochs.

rejection mechanism. Both DNS and KBGAN calculate the probability of the negative samples from a subset of candidates, while KBGAN is the relatively more efficient strategy. In short, the proposed adaptive sampling strategy AdaNS is satisfactory in terms of performance and efficiency compared to other state-of-the-art strategies.

7. Conclusion and Discussions

Summary. In this paper, an adaptive negative sampling strategy, named AdaNS, for unsupervised graph representation learning is proposed. Different from the existing strategies that sample randomly negative nodes, AdaNS adopts an efficient and effective

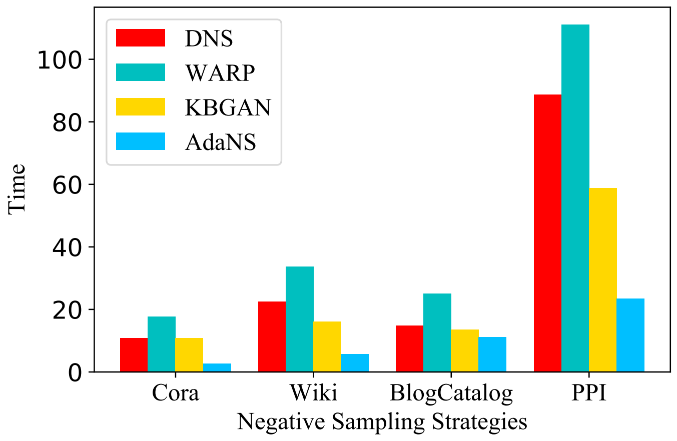


Fig. 6. The running time per epoch for different negative sampling strategies. Note that the unit of time for BlogCatalog is minutes, while the unit of time for the other datasets is seconds.

way to implement negative sampling by drawing hard negatives from the mixing distribution with respect to the dimensional elements in the node vectors. We conduct experiments on node classification and visualization tasks to evaluate the proposed strategy. The experimental results on seven benchmark datasets show that AdaNS is very competitive with state-of-the-art strategies.

Limitations of this work. There are several limitations from theoretical analysis and experimental justification. 1) To make the theoretical analysis more feasible, we make a few assumptions. We derive the parametric embedding as an instance in our analysis, since the main focus is the effect of negative samples on the gradient update. While the experiment results empirically demonstrate that our analysis seems to hold with GNN models, more formal investigation for deep neural network models is valuable. 2) Our proposed method is only verified on benchmark graphs, while there are more challenging in real-world scenarios, e.g., dynamic graphs and hypergraphs. Hence, it is crucial to devote more efforts to studying more complicated graphs. We believe our findings established a solid foundation for further research.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

This work is funded by: Postdoctoral Innovative Talents Support Program under Grant No. BX20190137, and China Postdoctoral Science Foundation funded project under Grant No. 2020M670839, and by the Fundamental Research Funds for the Central Universities, JLU No. 93K172020K36, and by Science Foundation of Jilin Province of China under Grant No. 2020122209JC.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2022.109266.

References

[1] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: ACM SIGKDD, 2016, pp. 855–864.
[2] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, ICLR, 2017.
[3] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Graph contrastive learning with adaptive augmentation, in: WWW, 2021, pp. 2069–2080.
[4] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: International conference on machine learning, PMLR, 2019, pp. 6861–6871.
[5] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NeurIPS, 2017, pp. 1024–1034.
[6] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, ICLR, 2019.
[7] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: ACM SIGKDD, 2014, pp. 701–710.
[8] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: NeurIPS, 2013, pp. 3111–3119.
[9] M. Gutmann, A. Hyvärinen, Noise-contrastive estimation: A new estimation principle for unnormalized statistical models, in: AISTATS, volume 9, 2010, pp. 297–304.
[10] A. Mnih, Y.W. Teh, A fast and simple algorithm for training neural probabilistic language models, ICML, 2012.
[11] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE: large-scale information network embedding, in: WWW, 2015, pp. 1067–1077.
[12] L.F.R. Ribeiro, P.H.P. Saverese, D.R. Figueiredo, struc2vec: Learning node representations from structural identity, in: ACM SIGKDD, 2017, pp. 385–394.
[13] P. Wang, S. Li, R. Pan, Incorporating GAN for negative sampling in knowledge representation learning, in: AAAI, 2018, pp. 2005–2012.
[14] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, D. Zhang, IRGAN: A minimax game for unifying generative and discriminative information retrieval models, in: SIGIR, 2017, pp. 515–524.
[15] H. Gao, H. Huang, Self-paced network embedding, in: Y. Guo, F. Farooq (Eds.), ACM SIGKDD, 2018, pp. 1406–1415.
[16] Z. Zhang, Y. Zeng, L. Bai, Y. Hu, M. Wu, S. Wang, E.R. Hancock, Spectral bounding: Strictly satisfying the 1-lipschitz property for generative adversarial networks, Pattern Recognit. 105 (2020) 107179.
[17] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: ACM SIGKDD, 2016, pp. 1225–1234.
[18] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, W. Zhu, Arbitrary-order proximity preserved network embedding, in: ACM SIGKDD, 2018, pp. 2778–2786.
[19] C. Zhou, Y. Liu, X. Liu, Z. Liu, J. Gao, Scalable graph embedding for asymmetric proximity, in: AAAI, 2017, pp. 2942–2948.
[20] A. Tsitsulin, D. Mottin, P. Karras, E. Müller, VERSE: versatile graph embeddings from similarity measures, in: WWW, 2018, pp. 539–548.
[21] Y. Shi, M. Lei, H. Yang, L. Niu, Diffusion network embedding, Pattern Recognit. 88 (2019) 518–531.
[22] C. Donnat, M. Zitnik, D. Hallac, J. Leskovec, Learning structural node embeddings via diffusion wavelets, in: ACM SIGKDD, 2018, pp. 1320–1329.
[23] C. Plant, S. Biedermann, C. Böhm, Data compression as a comprehensive framework for graph drawing and representation learning, in: ACM SIGKDD, 2020, pp. 1212–1222.
[24] C. Böhm, C. Plant, Massively parallel graph drawing and representation learning, in: 2020 IEEE International Conference on Big Data (Big Data), IEEE, 2020, pp. 609–616.
[25] Z. Zhang, D. Chen, Z. Wang, H. Li, L. Bai, E.R. Hancock, Depth-based subgraph convolutional auto-encoder for network representation learning, Pattern Recognit. 90 (2019) 363–376.
[26] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep graph contrastive representation learning, GRL+@ICML, 2020.
[27] X. Liu, J. Tang, Network representation learning: A macro and micro view, AI Open 2 (2021) 43–64.
[28] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: bayesian personalized ranking from implicit feedback, in: UAI, 2009, pp. 452–461.
[29] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: ICDM, 2008, pp. 263–272.
[30] W. Zhang, T. Chen, J. Wang, Y. Yu, Optimizing top-n collaborative filtering via dynamic negative item sampling, in: SIGIR, 2013, pp. 785–788.
[31] T. Zhao, J.J. McAuley, I. King, Improving latent factor models via personalized feature projection for one class recommendation, in: CIKM, 2015, pp. 821–830.
[32] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: ACM SIGKDD, 2018, pp. 974–983.
[33] L. Cai, W.Y. Wang, KBGAN: adversarial learning for knowledge graph embeddings, in: NAACL-HLT, 2018, pp. 1470–1480.
[34] J. Xie, X. Zhan, Z. Liu, Y.S. Ong, C.C. Loy, Delving into inter-image invariance for unsupervised visual representations, arXiv preprint arXiv:2008.11702 (2020).
[35] M. Wu, M. Mosse, C. Zhuang, D. Yamins, N.D. Goodman, Conditional negative sampling for contrastive learning of visual representations, ICLR, 2021.
[36] A. Mnih, K. Kavukcuoglu, Learning word embeddings efficiently with noise-contrastive estimation, in: NeurIPS, 2013, pp. 2265–2273.
[37] S. Prithviraj, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (3) (2008) 93–106.

[38] A. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of internet portals with machine learning, Inf. Retr. 3 (2) (2000) 127–163.
[39] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M.S. Livstone, R. Oughtred, D.H. Lackner, J. Bähler, V. Wood, K. Dolinski, M. Tyers, The bioGRID interaction database: 2008 update, Nucleic Acids Res. 36 (Database-Issue) (2008) 637–640.
[40] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: NeurIPS, 2012, pp. 1106–1114.
[41] J. Chen, T. Ma, C. Xiao, FastGCN: Fast learning with graph convolutional networks via importance sampling, ICLR, 2018.
[42] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, D. Koutra, Beyond homophily in graph neural networks: Current limitations and effective designs, NeurIPS, 2020.
[43] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of machine learning research 9 (11) (2008).

Yu Wang received the B.E degree and the M.S. degree from Jilin University in 2015 and 2018, where he is currently pursuing the Ph.D. degree. His research interests include data mining and machine learning.

Liang Hu received his PhD degree in the College of Computer Science from Jilin University in 1999. Currently, he is a professor at the College of Computer Science, Jilin University, China. His research areas are artificial intelligence and distributed computing.

Wanfu Gao received his M.S. and Ph.D. degrees in the College of Computer Science from Jilin University in 2016 and 2019. He is doing post-doctoral research in the College of Chemistry in Jilin University. His research interests include machine learning and feature selection.

Xiaofeng Cao received his Ph.D. degree at Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. He is currently an Associate Professor at the School of Artificial Intelligence, Jilin University, China and leading a Machine Perceptron Research Group with more than 15 PhD and Master students. He has published more than 10 technical papers in top tier journals and conferences, such as IEEE T-PAMI, IEEE TNNLS, IEEE T-CYB, CVPR, IJCAI. His research interests include PAC learning theory, agnostic learning algorithm, generalization analysis, and hyperbolic geometry.

Yi Chang received the Ph.D. degree in computer science from the University of Southern California, Los Angeles, CA, USA, in 2016. He is currently the Dean of the School of Artificial Intelligence, Jilin University, Changchun, China. He has published more than 100 research papers in premium conferences or journals. He has broad research interests on information retrieval, data mining, machine learning, and natural language processing. Dr. Chang is an Associate Editor of the IEEE Transactions on Knowledge and Data Engineering.