Graph Representation Learning via Adversarial Variational Bayes

Yunhe Li
University of Montreal
Montreal, Quebec, Canada
yunheli.cs@gmail.com

Yaochen Hu
Huawei Noah's Ark Lab
Montreal, Quebec, Canada
yaochen.hu@huawei.com

Yingxue Zhang
Huawei Noah's Ark Lab
Montreal, Quebec, Canada
yingxue.zhang@huawei.com

ABSTRACT

Methods that learn representations of nodes in a graph play an important role in network analysis. Most of the existing methods of graph representation learning have focused on embedding each node in a graph as a single vector in a low-dimensional continuous space. However, these methods have a crucial limitation: the lack of modeling the uncertainty about the representation. In this work, inspired by Adversarial Variational Bayes (AVB) [22], we propose GraphAVB, a probabilistic generative model to learn node representations that preserve connectivity patterns and capture the uncertainties in the graph. Unlike Graph2Gauss [3] which embeds each node as a Gaussian distribution, we represent each node as an implicit distribution parameterized by a neural network in the latent space, which is more flexible and expressive to capture the complex uncertainties in real-world graph-structured datasets. To perform the designed variational inference algorithm with neural samplers, we introduce an auxiliary discriminative network that is used to infer the log probability ratio terms in the objective function and allows us to cast maximizing the objective function as a two-player game. Experimental results on multiple real-world graph datasets demonstrate the effectiveness of our proposed method GraphAVB, outperforming many competitive baselines on the task of link prediction. The superior performances of our proposed method GraphAVB also demonstrate that the downstream tasks can benefit from the captured uncertainty.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Learning latent representations**.

KEYWORDS

Graph Representation Learning; Adversarial Variational Bayes

ACM Reference Format:
Yunhe Li, Yaochen Hu, and Yingxue Zhang. 2021. Graph Representation Learning via Adversarial Variational Bayes. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482116

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

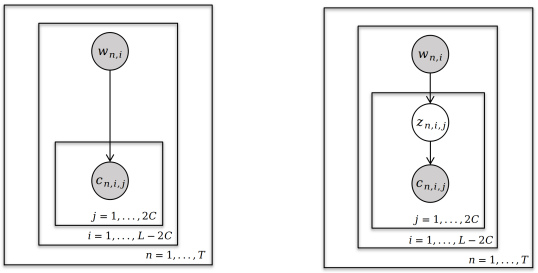
ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482116

1 INTRODUCTION

Graph representation learning plays a critical role in network analysis. In recent years, many methods have been proposed to learn graph representation, such as DeepWalk [27], LINE [33], node2vec [12] and struc2vec [29]. They represent each node as a single low-dimensional vector in the latent space and aim to preserve specific properties of a graph during the learning process, such as local and global structures, first-order and second-order proximities, diverse neighborhoods, and structural identity. However, most existing graph representation learning approaches suffer from a crucial issue that is the lack of modeling uncertainty. Uncertainty is ubiquitous and inherent when representing a node in a complex graph and should be captured during the representation learning process.

Inspired by Gaussian word embeddings [36], Bojchevski and Günnemann [3] proposed a graph representation learning method, Graph2Gauss, to model the uncertainties in a graph. Graph2Gauss represents each node as a Gaussian distribution and learns the representations via an energy-based learning framework. However, Graph2Gauss is not enough flexible to capture the uncertainties in a complex graph. For example, in an academic co-authorship network, a researcher may have multiple research interests and collaborate with diversified partners from different research fields. As a Gaussian distribution only have one mode, Graph2Gauss will fail to capture the truly complex multimodal distributions of the representations in the latent space.

In this paper, inspired by Adversarial Variational Bayes (AVB) [22], we propose GraphAVB, a novel probabilistic generative model for learning node representations which preserve connectivity patterns and capture the uncertainties in the graph. Unlike Graph2Gauss [3] which embeds each node as a Gaussian distribution, we represent each node as an implicit probability distribution in the latent space, which is called node-specific prior distribution. We can also obtain context-aware node representations which correspond to the approximating posterior distributions in our model. The node-specific prior distributions and approximating posterior distributions are parameterized by neural networks, so that they can be arbitrarily flexible and can recover the true distributions in the nonparametric limit. To perform the designed variational inference algorithm with neural samplers, we introduce an auxiliary discriminative network which is used to inference the log probability ratio terms in the objective function and allows us to cast maximizing the objective function as a two-player game. The objective function is composed of the variational lower bound of the pseudo-log-likelihood of the sampled source-context node pairs and a regularization term. We use a random walk approach to generate the source-context node pairs. Experimental results on multiple real-world graph datasets demonstrate the effectiveness of our proposed



(a) DeepWalk and node2vec (b) GraphAVB

Figure 1: Graphical model representations of DeepWalk, node2vec and GraphAVB. For GraphAVB, the outer plate represents node sequences generated by random walks in the graph where T is the total number of sampled node sequences, while the inner plates represent the repeated choices of source nodes, their latent representations and their context nodes within a node sequence where L is the length of each node sequence and C is the window size.

method GraphAVB, outperforming many competitive baselines on the task of link prediction. The superior performances of our proposed method GraphAVB also demonstrate that the downstream tasks can benefit from the captured uncertainty.

2 METHODOLOGY

2.1 Problem Definition

Let $G = (V, E)$ be a given graph, where $V = \{v_1, ..., v_N\}$ is the set of all nodes and E is the set of edges between these nodes, i.e. $E \subset V \times V$. Unlike most existing graph representation learning methods which try to learn a low-dimensional vector for each node, we aim to learn a implicit distribution in the low-dimensional latent space to represent each node which preserves connectivity patterns and captures the uncertainties in the graph. We define the node-specific prior distribution $p_\psi(z|w)$ as the representation of a node w. Also, we define the approximating posterior distribution $q_\phi(z|w, c)$ as the context-aware representation for a source node w appearing with a context node c. The node-specific prior distribution $p_\psi(z|w)$ and the approximating posterior distribution $q_\phi(z|w, c)$ are parameterized as neural samplers with parameters ψ and ϕ respectively, so that they can be enough flexible to capture the complicated uncertainty in real-world graph-structured datasets.

2.2 GraphAVB

GraphAVB models the generation of source-context node pairs. We assume that the choice of a context node c for a source node w is dependent on the latent representation z, and the latent representation z is dependent on the source node w. The representation z is modeled as a latent random vector, which can capture the uncertainty. The generative process for each source-context node pair (w, c) is the following: for the source node w, we first draw a latent representation $z \sim p(z|w)$. Then, the context node c is generated based on the latent representation z by drawing $c \sim p(c|z)$.

In order to preserve both local and global structures, we employ truncated random walks in the graph to efficiently generate the source-context node pairs. Specifically, for each node $v_i \in V$, we sample D node sequences with length of L and v_i as the starting point. In every sampling step, the next node is sampled uniformly from the neighbors of the current node. We also can use biased random walks [12], in which the node sequences sampling strategy can approximately interpolate between Breadth-First Search (BFS) and Depth-First Search (DFS) and hence reflects the preference of exploration between local and global structures. Then the source-context node pairs are constructed based on the sampled node sequences. Denote a sampled node sequences as $S_i, i \in \{1, ..., T\}$ and its j-th node as $s_{ij}, j \in \{1, ..., L\}$ where $T = D \times |V|$ is the number of sampled node sequences. The set of source-context node pairs based on the set of sampled node sequences $S = \{S_1, ..., S_T\}$ is defined as $\Omega_S = \{(s_{ij}, s_{ik}) | i \in \{1, ..., T\}, j \in \{C + 1, ..., L - C\}, k \in \{j - C, ..., j - 1, j + 1, ..., j + C\}\}$, where C is the window size.

The GraphAVB model is represented as a probabilistic graphical model in Figure 1(b). Given the set of source-context node pairs Ω_S, we want to maximize the pseudo-log-likelihood as follows:

$$\sum_{n=1}^{T} \sum_{i=1}^{L-2C} \sum_{j=1}^{2C} \log p_{\theta,\psi}(c_{n,i,j}|w_{n,i}). \tag{1}$$

Each term in Eq.(1) can be rewritten as follows (we omitted the subscripts for simplicity):

$$\log p_{\theta,\psi}(c|w) = \log \int p_\theta(c|z) p_\psi(z|w) \mathrm{d}z. \tag{2}$$

Because the integral in Eq.(2) is intractable, we maximize its variational lower bound instead. To derive the variational lower bound, we rewrite Eq.(2) as follows:

$$\begin{aligned} \log p_{\theta,\psi}(c|w) &= \log \int p_\theta(c|z) p_\psi(z|w) \frac{q_\phi(z|w,c)}{q_\phi(z|w,c)} \mathrm{d}z \\ &\geq \mathbb{E}_{q_\phi(z|w,c)} [\log p_\theta(c|z) + \log p_\psi(z|w) - \log q_\phi(z|w,c)], \end{aligned} \tag{3}$$

where the inequality is derived through Jensen's inequality. As $p_\psi(z|w)$ and $q_\phi(z|w, c)$ are parameterized as neural samplers, they can be enough flexible to learn complex probability distributions. However, the log probability ratio term $\log p_\psi(z|w) - \log q_\phi(z|w, c)$ in the variational lower bound cannot be computed directly. Inspired by Adversarial Variational Bayes (AVB) [22], we introduce an auxiliary discriminative network $T(w, c, z)$ to inference the log probability ratio term $\log p_\psi(z|w) - \log q_\phi(z|w, c)$. Specifically, the term $\log q_\phi(z|w, c) - \log p_\psi(z|w)$ is the optimal value of the discriminative network $T(w, c, z)$ in the following optimization problem with fixed $q_\phi(z|w, c)$ and $p_\psi(z|w)$:

$$\max_T \mathbb{E}_{q_\phi(z|w,c)} \log \sigma(T(w, c, z)) + \mathbb{E}_{p_\psi(z|w)} \log(1 - \sigma(T(w, c, z))) \tag{4}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

To present the theoretical analysis, we assume that the discriminative network $T(w, c, z)$ has enough capacity to represent any function, i.e., in the non-parametric limit which is the same assumption in GAN [11] and AVB [22].

Proposition 1. *For $q_\phi(z|w,c)$ and $p_\psi(z|w)$ fixed, the optimal discriminator T^* in the optimization problem (4) is*

$$T^*(w,c,z) = \log q_\phi(z|w,c) - \log p_\psi(z|w).$$

Proof. The proof is similar with the proof of Proposition 1 in GAN [11] and AVB [22]. We rewrite the objective function in the optimization problem (4) as the following:

$$\int_z q_\phi(z|w,c)\log\sigma(T(w,c,z)) + p_\psi(z|w)\log(1-\sigma(T(w,c,z)))dz.$$

For any $(a,b) \in \mathbb{R}^2\backslash\{0,0\}$, the function $t \to a\log t + b\log(1-t)$ attains its maximum at $t = \frac{a}{a+b}$. So that we have

$$T^*(w,c,z) = \log q_\phi(z|w,c) - \log p_\psi(z|w).$$

□

To better learn the node-specific prior distribution $p_\psi(z|w)$, we add a regularization term $-\lambda KL(p_\psi(z|w), q_\phi(z|w,c))$ to the variational lower bound in Eq.(3), where λ is a tunable hyper-parameter and $KL(\cdot,\cdot)$ denotes the Kullback–Leibler divergence. As a result, the objective function we want to maximize is the following:

$$\begin{aligned}&\mathbb{E}_{q_\phi(z|w,c)}\left[\log p_\theta(c|z) + \log p_\psi(z|w) - \log q_\phi(z|w,c)\right]\\&- \lambda\mathbb{E}_{p_\psi(z|w)}\left[\log p_\psi(z|w) - \log q_\phi(z|w,c)\right].\end{aligned} \tag{5}$$

Leveraging Proposition 1, the process of maximizing the objective function in Eq.(5) is cast into the following two-player game:

$$\max_\eta \mathbb{E}_{q_\phi(z|w,c)}\left[\log\sigma(T_\eta(w,c,z))\right] + \mathbb{E}_{p_\psi(z|w)}\left[\log(1-\sigma(T_\eta(w,c,z)))\right] \tag{6}$$

$$\max_{\theta,\phi,\psi} \mathbb{E}_{q_\phi(z|w,c)}\left[\log p_\theta(c|z) - T_\eta(w,c,z)\right] + \lambda\mathbb{E}_{p_\psi(z|w)}\left[T_\eta(w,c,z)\right] \tag{7}$$

Proposition 1 shows that any Nash-equilibrium of this two-player game yields a stationary point of the objective in Eq.(5).

Intuitively, given an observed source-context node pair (w,c), the discriminative network $T(w,c,z)$ trys to distinguish z sampled from the node-specific prior distribution $p_\psi(z|w)$ from that sampled from the context-aware approximating posterior distributions $q_\phi(z|w,c)$ in the Eq.(6). In the Eq.(7), minimizing the term $\mathbb{E}_{q_\phi(z|w,c)}[T_\eta(w,c,z)]$ tries to make $q_\phi(z|w,c)$ close to $p_\psi(z|w)$, which can be regarded as a regularization. Maximizing the term $\lambda\mathbb{E}_{p_\psi(z|w)}[T_\eta(w,c,z)]$ tries to make $p_\psi(z|w)$ close to $q_\phi(z|w,c)$ and confuse the discriminative network $T(w,c,z)$.

Using the reparameterization trick [14], the two-player game in Eq.(6) and Eq.(7) can be rewritten as follows:

$$\begin{aligned}\max_\eta\ &\mathbb{E}_\epsilon[\log\sigma(T_\eta(w,c,z_\phi(w,c,\epsilon)))\\&+\log(1-\sigma(T_\eta(w,c,z_\psi(w,c,\epsilon))))]\end{aligned} \tag{8}$$

$$\begin{aligned}\max_{\theta,\phi,\psi}\ &\mathbb{E}_\epsilon[\log p_\theta(c|z_\phi(w,c,\epsilon)) - T_\eta(w,c,z_\phi(w,c,\epsilon))\\&+\lambda T_\eta(w,c,z_\psi(w,c,\epsilon))]\end{aligned} \tag{9}$$

where $\epsilon \sim p(\epsilon)$, $z_\phi(w,c,\epsilon)$ and $z_\psi(w,\epsilon)$ are suitable functions which can be parameterized by neural networks. $p_\theta(c|z_\phi(w,c,\epsilon))$ is a categorical distribution parameterized as a softmax model as the following:

$$p_\theta(c|z_\phi(w,c,\epsilon)) = \frac{\exp(\theta_c^T z_\phi(w,c,\epsilon))}{\sum_{c'\in V}\exp(\theta_{c'}^T z_\phi(w,c,\epsilon))}. \tag{10}$$

The cost of computing the summation in the denominator of Eq.(10) is proportional to the number of nodes, which can be expensive. So for large datasets, we can employ negative sampling [24] which replaces every $\log p_\theta(c|z_\phi(w,c,\epsilon))$ term in the objective functions with the following terms:

$$\log\sigma(\theta_c^T z_\phi(w,c,\epsilon)) + \sum_{i=1}^{K}\mathbb{E}_{v\sim P_n(v)}\left[\log\sigma(-\theta_v^T z_\phi(w,c,\epsilon))\right]$$

where K is the number of negative samples and $P_n(v)$ is a noise distribution for sampling negative context nodes. We set $P_n(v) \propto d_v^{\frac{3}{4}}$ as done in LINE[33], where d_v is the out-degree of node v.

2.3 Optimization

In practice, we try to find a Nash-equilibrium by applying Stochastic Gradient Descent (SGD) jointly to Eq.(8) and Eq.(9). It is crucial to keep the discriminative T network close to optimality while optimizing Eq.(9). Hence we can perform several SGD-updates to Eq.(8) for one SGD-update to Eq.(9).

We employ an effective optimization strategy which decomposes the process of optimization into two stages. In the first stage, we fix the node-specific prior distribution $p_\psi(z|w)$ as a multivariate Gaussian distribution $N(0,\beta\mathrm{I})$ and optimize objectives over η,θ,ϕ. The node-specific prior distribution $p_\psi(z|w) = N(0,\beta\mathrm{I})$ regularizes the model and encourages the global smoothness of the latent representation distribution [7, 8] where β is a hyper-parameter to control the regularization. In the second stage, we fix the learned context-aware approximating posterior distributions $q_{\phi^*}(z|w,c)$ and the learned categorical distribution $p_{\theta^*}(c|z)$, and optimize objectives over η,ψ. The node-specific prior distribution $p_\psi(z|w)$ is parameterized as a neural sampler in the second stage, so that it can be enough flexible to model the complex uncertainty. Intuitively, $p_\psi(z|w)$ tries to summarize $q_{\phi^*}(z|w,c)$ for a source node w appearing with different context nodes c during the second stage. This strategy can prevent the model from overfitting as well.

3 EXPERIMENTS

In this section, we empirically evaluate our proposed method through the task of link prediction on 5 benchmark datasets.

There are several reasons why we focus on the fundamental task of link prediction. Firstly, we are interested in modeling the complicated uncertainty and interactions between nodes in the real-world graph datasets. Link prediction is especially suitable for testing whether the models capture these properties. Secondly, link prediction is suggested as the primary task to evaluate the performance of unsupervised graph representation learning methods by recent works [9, 41]. Contrastively, node classification is a secondary task because the process of labeling is not necessarily closely related with the graph itself [9]. Finally, there are many kinds of important applications of link prediction in the real world such as friends recommendation in social networks.

Table 1: The statistics of our datasets.

Dataset	$\|V\|$	$\|E\|$	Average Degree
Cora	2708	5278	1.95
Citeseer	3264	4552	1.39
Wiki	2363	11824	5.00
CA-GrQc	5241	14484	2.76
CA-HepTh	9875	25973	2.63

3.1 Datasets and Baseline Methods

We conduct experiments on 5 real-world datasets including Cora [21], Citeseer [21], Wiki [30], CA-GrQc [17], and CA-HepTh [17]. Cora [21] and Citeseer [21] are paper citation networks. Wiki [30] is a web page link network. CA-GrQc [17] and CA-HepTh [17] are author collaboration networks. The statistics of the datasets are provided in Table 1. These datasets are preprocessed by deleting self-loops and nodes with zero degree.

We compare GraphAVB with the competitive baseline methods including Graph Factorization (GF) [1], GraRep [5], DeepWalk [27], LINE [33], node2vec [12], Graph2Gauss [3], and AdvT4NE [8].

3.2 Experiment Configuration

For GraphAVB, DeepWalk, node2vec, Graph2Gauss, and AdvT4NE, the walk length, walks per node, window size and the number of negative samples per node are set to 40, 1, 5 and 5, respectively. For GraphAVB, we use uniform random walks to sample the node sequences and simply set the regularization strength $\lambda = 1$. For GraphAVB, we use the RMSProp optimizer and set the learning rate to 0.001. Besides, the dimension of node representations is set to 128 for all methods.

3.3 Link Prediction

We follow the methodology introduced in [8] to conduct the task of link prediction. In detail, we firstly perform the graph representation learning methods on a sub-graph, which contains 80% of edges of the original graph, to learn node representations. During the sub-sampling process, the degree of each node is ensured to be greater than or equal to 1 to avoid meaningless node representations. Then we treat link prediction as a classification problem. We train a L_2-SVM classifier whose inputs are the edge features obtained by the Hadamard product of embedding vectors of the two endpoints for baseline methods. The positive training examples are the observed edges of the sub-graph used to learn node representations and the same number of negative training examples are randomly sampled from the node pairs without an edge connection in the original graph. The positive testing examples are the removed 20% edges of the original graph and the two times number of negative testing examples are randomly sampled from the node pairs which are not in the set of negative training examples and without an edge connection in the original graph. We report the AUC score as the performance measure. For Graph2Gauss, we use the mean vector of each Gaussian distribution as the embedding vector of each node.

To leverage the advantages of GraphAVB for link prediction, we use the following method to construct edge features. As GraphAVB uses the dot product to encode the similarity, we use the Hadamard product of the two vectors sampled from node-specific prior distributions of the two endpoints respectively, whose dot product are the maximum among the pairs of the sampled vectors. More detailed, for $v_i, v_j \in V$, the edge feature e_{ij} is constructed as follows:

$$e_{ij} = z_\psi(v_i, \epsilon_{im}) \odot z_\psi(v_j, \epsilon_{jm}),$$

where $im, jm = \arg\max_{u,v} z_\psi(v_i, \epsilon_u) \cdot z_\psi(v_j, \epsilon_v)$, $\epsilon_u, \epsilon_v \sim p(\epsilon)$, $u, v = 1, ..., M$, $\odot$ denotes the Hadamard product and $\cdot$ denotes the dot product. We set $M = 5$ in our experiments.

In our implementation, the noise distribution $p(\epsilon)$ used in the reparameterization trick is a standard multivariate Gaussian distribution. $z_\phi(w, c, \epsilon)$ is parameterized as a 2-layer neural network whose inputs are the noise vector ϵ and the one-hot vectors of the source node w and the context node c. We assign each node w a node-specific prior network, i.e. $z_\psi(w, \epsilon) = z_{w,\psi}(\epsilon)$, which is parameterized as a 3-layer fully connected neural network whose input is the noise vector ϵ. Besides, we assign each node w a discriminative network $T_{w,\eta}(z)$ parameterized as a 3-layer fully connected neural network with skip connections, which is an approximation to $T_\eta(w, c, z)$ [20, 22].

Table 2: The AUC score for link prediction.

Method/Dataset	Cora	Citeseer	Wiki	CA-GrQc	CA-HepTh
GF	0.6093	0.6090	0.7294	0.8583	0.7929
GraRep	0.7367	0.7391	0.7778	0.8535	0.8361
DeepWalk	0.7479	0.7516	0.7819	0.8949	0.8454
LINE	0.7228	0.7317	0.7886	0.8891	0.8250
node2vec	0.7591	0.7783	0.7923	0.8805	0.8428
Graph2Gauss	0.7527	0.7497	0.7783	0.8819	0.8267
AdvT4NE	0.7623	0.7728	0.7522	0.8855	0.8210
GraphAVB	**0.7785**	**0.7829**	**0.8089**	**0.9027**	**0.8515**

The experimental results for link prediction on 5 real-world graph datasets are summarized in Table 2. It can be easily observed that our proposed method GraphAVB outperforms all baseline methods on all 5 graph datasets in terms of the AUC score. One of the reasons is that the Graph2Gauss model is not enough flexible to capture the uncertainty in real-world graph datasets and the other baseline methods are short of modeling the uncertainty. The experimental results also demonstrate that using a flexible and expressive model to capture the uncertainty and leveraging the advantage of the captured uncertainty can help improve the performance of downstream tasks such as link prediction.

4 CONCLUSION

In this paper, we propose GraphAVB, a novel probabilistic generative model to learn node representations which preserve connectivity patterns and capture the uncertainties in the graph. Inspired by AVB [22], we introduce an auxiliary discriminator that is used to infer the log probability ratio terms in the objective function and allows us to cast maximizing the objective function as a two-player game. Empirical evaluation on 5 real-world graph datasets demonstrates the effectiveness of GraphAVB and the benefits from the captured uncertainty on the task of link prediction.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*. 37–48.

[2] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *J. Amer. Statist. Assoc.* 112, 518 (2017), 859–877.

[3] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[4] Arthur Bražinskas, Serhii Havrylov, and Ivan Titov. 2017. Embedding words as distributions with a Bayesian skip-gram model. *arXiv preprint arXiv:1711.11027* (2017).

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 891–900.

[6] Koby Crammer and Yoram Singer. 2002. On the learnability and design of output codes for multiclass problems. *Machine learning* 47, 2 (2002), 201–233.

[7] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2017. Adversarial Network Embedding. *arXiv preprint arXiv:1711.07838* (2017).

[8] Quanyu Dai, Xiao Shen, Liang Zhang, Qiang Li, and Dan Wang. 2019. Adversarial training methods for network embedding. In *The World Wide Web Conference*. 329–339.

[9] Alessandro Epasto and Bryan Perozzi. 2019. Is a single embedding enough? learning node representations that capture multiple social contexts. In *The World Wide Web Conference*. 394–404.

[10] Hongchang Gao, Jian Pei, and Heng Huang. 2019. ProGAN: Network Embedding via Proximity Generative Adversarial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1308–1316.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[13] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 623–632.

[14] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[15] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[16] Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh, and Shou-De Lin. 2017. PRUNE: Preserving Proximity and Global Ranking for Network Embedding. In *Advances in Neural Information Processing Systems*. 5263–5272.

[17] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.

[18] Juzheng Li, Jun Zhu, and Bo Zhang. 2016. Discriminative Deep Random Walk for Network Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 1004–1013.

[19] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding.. In *IJCAI*. 1774–1780.

[20] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).

[21] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[22] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *arXiv preprint arXiv:1701.04722* (2017).

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[25] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 1895–1901.

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[28] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*. 1278–1286.

[29] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *KDD*.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[31] Fan-Yun Sun, Meng Qu, Jordan Hoffmann, Chin-Wei Huang, and Jian Tang. 2019. vGraph: A Generative Model for Joint Community Detection and Node Representation Learning. In *Advances in Neural Information Processing Systems*. 512–522.

[32] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 287–297.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 1067–1077.

[34] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1722–1731.

[35] Cunchao Tu, Hao Wang, Xiangkai Zeng, Zhiyuan Liu, and Maosong Sun. 2016. Community-enhanced network representation learning for network analysis. *arXiv preprint arXiv:1611.06645* (2016).

[36] Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623* (2014).

[37] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1225–1234.

[38] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2017. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. *arXiv preprint arXiv:1711.08267* (2017).

[39] David Warde-Farley and Ian Goodfellow. 2016. Adversarial Perturbations of Deep Neural Networks. *Perturbations, Optimization, and Statistics* (2016), 311.

[40] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*. 40–48.

[41] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2778–2786.

[42] Jun Zhu and Bei Chen. 2017. Latent feature models for large-scale link prediction. *Big Data Analytics* 2, 1 (2017), 3.