ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200203

Adversarial Learning for Overlapping Community Detection and Network Embedding

Junyang Chen, Zhiguo Gong [1], Quanyu Dai [2], Chunyuan Yuan [3] and Weiwen Liu [4]

Abstract. Network Embedding (NE) aims at modeling network graph by encoding vertices and edges into a low-dimensional space. These learned vectors which preserve proximities can be used for subsequent applications, such as vertex classification and link prediction. Skip-gram with negative sampling is the most widely used method for existing NE models to approximate their objective functions. However, this method only focuses on learning representation from the local connectivity of vertices (i.e., neighbors). In real-world scenarios, a vertex may have multifaceted aspects and should belong to overlapping communities. For example, in a social network, a user may subscribe to political, economic and sports channels simultaneously, but the politics share more common attributes with the economy and less with the sports. In this paper, we propose an adversarial learning approach for modeling overlapping communities of vertices. Each community and vertex are mapped into an embedding space, while we also learn the association between each pair of community and vertex. The experimental results show that our proposed model not only can outperform the state-of-the-art (including GANs-based) models on vertex classification tasks but also can achieve superior performances on overlapping community detection.

1 Introduction

Graph structures are ubiquitous in real-world applications, such as citation networks and social networks. Network embedding (NE) can map the semantic similarity of graph vertices into a low-dimensional space where the similar vertices are assigned to the nearby areas [5]. The learned embeddings are useful for the subsequent applications, such as link prediction [14] and vertex classification [24]. Lots of previous works have been devoted to NE for preserving the proximities in networks. For example, DeepWalk [24] performs truncated random walks to explore the networks. Line [29] extends DeepWalk by using depth-first search (DFS) and breadth-first search (BFS) strategies. Then, Node2vec [14] is proposed to take both BFS and DFS into consideration and designs a biased random walk procedure to explore diverse neighborhoods. After searching out neighbors of vertices, these methods adopt Skip-gram with negative sampling [20, 21], a language model that maximizes the probability of word co-occurrences (corresponding to vertex neighbors in graphs) within a sliding window, to learn vertex representations. The major idea of NE is to encourage a target vertex to be close to its neighbors and meanwhile be far from its negative samples.

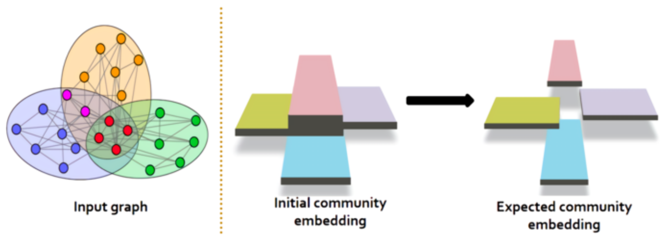


Figure 1: An example of overlapping communities within a graph, which is built from an online video website where users are denoted as vertices and edges represents the relation of subscribing common video genres (note that this figure is referred to [25].). The objective is to learn the representations which preserve the vertex and community proximities in the embedding space.

Nevertheless, most exiting NE models only focus on learning the local connectivity of neighbor vertices but ignore global patterns which are known as communities in many complex networks. As shown in many real-world scenarios, entities may contain disparate aspects [23, 26]. In networks, different paths stretching out from a vertex to its n-step neighbors may result from the expression of its aspects (i.e., communities). For example, a social network of an online video website is shown in Figure 1 where users are denoted as vertices and edges represent the relations of subscribing to common video genres. Since a user may simultaneously subscribe to political, economic and sports channels (represented as communities in the followings), if we neglect community structure information in network embedding, the learned representations of the user and its communities have to be close to each other in the embedding space, while the politics community share more common users with the economy and less with the sports. Therefore, the community structure is an important pattern of vertices and expected to benefit network embedding as well as overlapping community detection.

The challenges of developing overlapped community-aware network embedding modes are three-fold: (1) How to determine the communities of vertices; (2) How to map the community assignments from discrete space to continuous embedding space; (3) How to customize the objective function to make the embedding vectors of vertices and their assigned communities be close to each other while be far away from irrelevant communities. Recently, generative adversarial networks (GANs) [12] have received a lot of attention and achieved success in various applications [36]. One important effect of GANs is to learn a map of an input from a simple distribution to a complicated distribution (e.g., embedding space) [11]. Some previous works have introduced GANs into NE. For instance, GraphGAN [33] firstly unifies generative and discriminative models of vertices to boost NE performance. A-RNE [7] employs triplet ranking

[1] University of Macau, China, email: {yb77403, fstzgg}@umac.mo
[2] The Hong Kong Polytechnic University, quanyu.dai@connect.polyu.hk
[3] University of Chinese Academy of Sciences, yuanchunyuan@iie.ac.cn
[4] The Chinese University of Hong Kong, wwliu@cse.cuhk.edu.hk

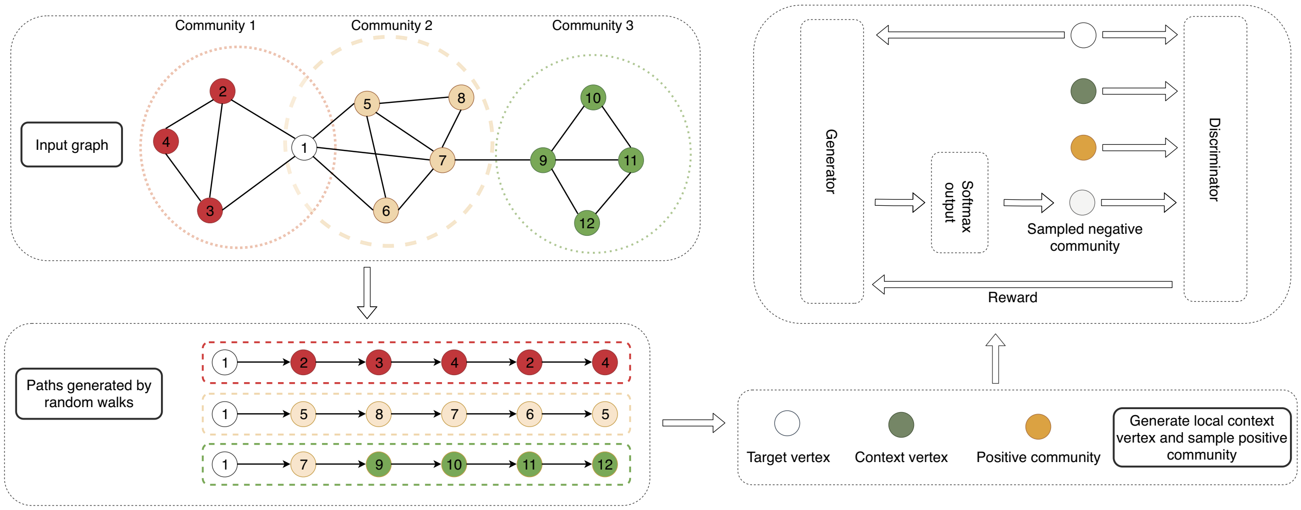


Figure 2: The training framework of ACNE.

loss to generate high-quality negative nodes and leverages GANs in the training. ProGAN [11] discovers ordinary underlying proximities with GANs to benefit NE. AIDW [6] proposes an improved version of DeepWalk [24] with GAN-based regularization method. However, all of these methods adopt standard GANs to generate samples of vertices rather than communities. Thus, how to incorporate community information into GANs is still challenging.

To address the aforementioned problems, in this paper, we propose an adversarial learning method (ACNE) for overlapping community detection and network embedding. Specifically, communities and vertices are represented as embedding vectors. For each vertex, we firstly sample a community from its random walking path. Then, to map the relation of a discrete vertex community assignment into a continuous vertex-community embedding space, we utilize a discriminator to learn the vertex representation by jointly maximizing the probabilities of predicting its context vertices and assigned community. In this way, the correlation of vertex-vertex from local connectivity and vertex-community from network structure can be uniformly preserved in our model. Furthermore, to capture the underlying community proximity and obtain distinguishable embedding vectors, ACNE employs a softmax generator to construct high-quality negative communities instead of a simple uniform sampling method, i.e., the pairs of targeted vertex and negative communities with higher probabilities from discriminator will be encouraged to be generated. At last, to evaluate whether considering community structure and vertex connectivity could actually benefit NE, we conduct experiments of overlapping community detection and vertex classification on a variety of real-world networks to compare the performance between our proposed ACNE and state-of-the-art models. Our major contributions are as follows:

- We propose an adversarial learning method (ACNE) for incorporating overlapping community information into network representation learning. We exploit the GANs technique to learn a map between discrete vertex community assignments and vertex-community embeddings. Moreover, ACNE is able to learn the correlations of vertex-vertex from local connectivity and vertex-community from network structure jointly.
- We leverage a softmax generator for generating high-quality negative communities with respect to a given target, and thus can obtain more discriminative community representations in the embedding space.
- We empirically evaluate ACNE through several network analysis tasks, including overlapping community detection and vertex classification. Experimental results show that ACNE achieves significant and consistent improvements than state-of-the-art models.

The code and datasets are at https://github.com/junyachen/ACNE. The rest of this paper is organized as follows. In Section 2, we introduce the core idea of our proposed model and present the ACNE algorithm. We discuss experimental results in Section 3 and introduce the related work in Section 4. Section 5 concludes our work.

2 Proposed ACNE Model

In this section, we will first introduce the problem formulation and notations. Then, we will present an overview of the training framework of ACNE shown in Figure 2, followed by detailed descriptions of each component.

2.1 Problem Formulation and Notations

Since we aim to detect overlapping communities and learn network embedding. The problems can be formulated as follows:

Network embedding. We denote a network as $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ denotes the set of edges. For each vertex $v \in V$, NE aims to learn a low-dimensional embedding $\mathbf{v} \in \mathbb{R}^d$ which preserves the network proximity. Here $d \ll |V|$ represents the dimension of representation space.

Community embedding. The number of communities is regarded as prior knowledge denoted as C. For each community c, we also want to learn a low-dimensional embedding $\mathbf{c} \in \mathbb{R}^d$ which has the same dimension as vertex representations. In general, mapping vertex-vertex and vertex-community relations into the same embedding space can help to integrate the information of local connectivity and network structure, which also benefits overlapping community detection. In this way, we can measure the similarity of vertices and communities by calculating the inner product of their vectors.

Random walk. It is a widely used method for exploring the neighborhood of vertices in a given network [6, 8, 24, 31]. Then, the network can be transformed into vertex sequences which contain the semantic relations between vertices. The set of walk sequences is denoted as $S = \{s_1, ..., s_N\}$ and each sequence is $s = \{v_1, ..., v_{|s|}\}$. N denotes the total number of walk sequences.

2.2 Context-aware Community Assignment

As shown in the left side of Figure 2, to explore the neighborhood of vertices, we firstly generate paths from a given network G. For example, start with vertex 1, multiple paths can be generated with random walks. As mentioned before, in network structures, different paths stretching out from a vertex to its n-step neighbors may result from the expression of its communities. Therefore, the community assignment of each vertex in a sequence should be related to its context vertices. Then, for each given vertex and its associated sentence, following the Gibbs Sampling method [13], the conditional probability of community assignments can be estimated as:

$$p(c|v, s) \propto p(c|s_{\neg v}) \prod_{\hat{v} \in s, \hat{v} \neq v} p(c|\hat{v}) \tag{1}$$

where $p(c|s_{\neg v})$ represents the community distribution in the sequence s except vertex v, $p(c|\hat{v})$ denotes the vertex-community distribution learned from global network structures which we assume it as prior knowledge (we will discuss how to obtain this distribution later). Then, we can further formulate $p(c|s_{\neg v})$, the conditional probability of a community c given a sequence s, as follows:

$$p(c|s_{\neg v}) = \frac{N(c, s_{\neg v})}{\sum_{\hat{c}}^{C} N(\hat{c}, s_{\neg v})} \tag{2}$$

where C denotes the set of communities, $N(c, s_{\neg v})$ represents the number of vertices assigned to community c in sequence s except current vertex v. Then, we discuss how we obtain the vertex-community distribution. Let $\mathbf{M} \in \mathbb{R}^{V \times V}$ be the asymmetric adjacency matrix of a network, non-negative matrix factorization (NMF) can be performed for learning community distributions [16, 38] by solving:

$$\min_{\mathbf{W} \geq 0} ||\mathbf{M} - \mathbf{W} \cdot \mathbf{W}^T||_F^2 + \alpha ||\mathbf{W}||_F^2 \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{V \times C}$ indicates the vertex-community distribution which encodes the global understanding of the network states, $||\cdot||_F$ is Frobenius norm of the matrix and α is a harmonic factor to balance two components. The probability of vertex $\hat{v}$ assigned to community c can be calculated as follows:

$$p(c|\hat{v}) = \frac{\mathbf{W}_{\hat{v},c}}{\sum_{\hat{c}}^{C} \mathbf{W}_{\hat{v},\hat{c}}} \tag{4}$$

In general, the reason why not simply using $\mathbf{W}$ for community detection or network embedding is that we want to preserve the correlation of vertex-vertex from local connectivity and vertex-community from global network structure into the same embedding space when learning the vertex and community representations. More concretely, for each vertex in a specific sequence, we firstly sample a community from its context vertices in the walking path. Then, we utilize a discriminator to learn the vertex representation by jointly maximizing the probabilities of predicting its context vertices and discrete assigned community. Experimental results also indicate that this jointly learning method outperforms the separate learning way (we represent the general network embedding and the jointly modeling for comparison). All details will be introduced in the following sections.

2.3 Adversarial Learning of Vertices and Communities

Inspired by the development of the GANs technique, we can map the discrete results learned from generative models into a continuous embedding space. However, the standard GANs is designed to generate samples of vertices rather than communities. Hence, how to generate the underlying community is still challenging.

Generator G. The purpose of the generator in ACNE is to construct high-quality negative communities with respect to a given target vertex. We employ softmax function over a set of negative candidates in the generator which is defined as follows:

$$G(c_n|v_t; \theta_G) = \frac{\exp(\mathbf{c}_n \cdot \mathbf{v}_t^T)}{\sum_{\hat{c} \in C} \exp(\hat{\mathbf{c}} \cdot \mathbf{v}_t^T)} \tag{5}$$

where v_t is the target vertex, c_n is the generated negative community, θ_G is the union of all vertex and community embeddings in the generator, C is the set of communities and its size is a predefined parameter usually set $|C| \ll |V|$. Thus, the summation term inside Eq. (5) only takes the slight expense of computation. Moreover, the loss function of the generator can be defined as:

$$\mathcal{L}_G = \sum_{v_t \in \mathcal{B}} \mathbb{E}_{c_n \sim G(\cdot|v_t;\theta_G)} D(c_n, v_t; \theta_D) \tag{6}$$

where $\mathcal{B}$ denotes a batch in the training process, θ_D is the union of all vertex and community embeddings in the discriminator, and $D(\cdot)$ indicates sigmoid function, i.e., $D(c_n, v_t; \theta_D) = \sigma(\mathbf{c}_n \cdot \mathbf{v_t}^T) = \frac{1}{1+\exp(-\mathbf{c}_n \cdot \mathbf{v_t}^T)}$. In summary, this formulated generator aims to sample high-quality negative communities from the softmax probability distribution $G(c_n|v_t; \theta_G)$ instead of uniform sampling which may generate totally unrelated communities. However, the output of the generator is a discrete index of the communities. Therefore, stochastic gradient descent (SGD) method can not be directly used for optimization. According to [27, 41], we can use policy gradient based reinforcement learning method to optimize the generator loss as:

$$\begin{aligned} \nabla_{\theta_G} \mathcal{L}_G &= \nabla_{\theta_G} \sum_{v_t \in \mathcal{B}} \mathbb{E}_{c_n \sim G(\cdot|v_t;\theta_G)} D(c_n, v_t; \theta_D) \\ &= \sum_{v_t \in \mathcal{B}} \mathbb{E}_{c_n \sim G(\cdot|v_t;\theta_G)} D(c_n, v_t; \theta_D) \nabla_{\theta_G} \log G(c_n|v_t; \theta_G) \end{aligned} \tag{7}$$

where the gradient of $\mathcal{L}_G$ is an expected summation of ∇_{θ_G}log $G(c_n|v_t; \theta_G)$ weighted by $D(\cdot)$ which is calculated with the discriminator. In the field of reinforcement learning, $D(\cdot)$ in Eq. (7) can be regarded as a reward function and the generator is trained to maximize the expected reward. In order to achieve a higher reward, for each negative pair (c_n, v_t), the policy used by the generator network would punish trivial negative communities by lowering down their corresponding probability and encourage the discriminator network to distribute high-quality negative communities, i.e., pair (c_n, v_t) with higher similarity from discriminator parameterized by θ_D will be encouraged to be generated. Moreover, in practice, the reinforcement-based algorithms may suffer from unstable performance and achieve high variance results[37]. According to [28], this problem can be alleviated by adding a baseline function to the reward term in the gradient loss. Then, $D(\cdot)$ can be replaced by:

$$D(c_n, v_t; \theta_D) + \frac{\sum_{\mathcal{B} \in \mathcal{P}} \sum_{v_t \in \mathcal{B}} \mathbb{E}_{c_n \sim G(\cdot|v_t;\theta_G)} D(c_n, v_t; \theta_D)}{|\mathcal{P}|} \tag{8}$$

where $\mathcal{P}$ denotes the whole batches in the training set, and the baseline function is the average reward obtained in the training process.

Discriminator D. The discriminator of our proposed ACNE aims to complete two goals. The first one is to distinguish the context vertices and the negative vertices, which is the same as the Skip-gram model [20, 21]. The second goal is to determine whether the generated triplet satisfies $D(c_p, v_t) \gg D(c_n, v_t)$, where c_p, c_n, v_t indicate positive community, negative community, and target vertex, respectively. To achieve these goals, as shown in the right side of Figure 2, for each target vertex v_t and its generated tuple $\{v_c, v_n, c_p, c_n\}$ in a given sequence $s \in \mathcal{B}$, we will learn their representations by jointly maximizing the probabilities of predicting its context vertices and assigned community, which can be formalized as:

$$\begin{aligned}\mathcal{L}_D = &\sum_{s\in\mathcal{B}}\sum_{v_t\in s}\mathbb{E}_{c_n\sim G(\cdot|v_t;\theta_G)}[\log D(v_t, v_c) \\ &+ \log D(-v_t, v_n) + \log D(v_c, c_p) + \log D(-v_t, c_n) \\ &+ \lambda(||\mathbf{v}_t||_F^2 + ||\mathbf{v}_c||_F^2 + ||\mathbf{v}_n||_F^2 + ||\mathbf{c}_p||_F^2 + ||\mathbf{c}_n||_F^2)]\end{aligned} \tag{9}$$

where v_c denotes the context vertex of v_t, v_n is the negative vertex generated by negative sampling [20], the generation of v_c and v_n is a common practice in NE literature [24, 29], c_p is the positive community sampled by using Eq. (1), c_n is the negative community generated with Eq. (5), $||\cdot||_F$ is Frobenius norm of vectors, $\{\mathbf{v}_t, \mathbf{v}_c, \mathbf{v}_n, \mathbf{c}_p, \mathbf{c}_n\}$ are the embedding vectors in θ_D, and λ is a harmonic factor for regularization. The discriminator D can be optimized with gradient descent technique.

2.4 Training Process of ACNE

The overall training process of our proposed ACNE is summarized in Algorithm 1. To be specific, similar to the learning process in [2, 33, 34], we iterate the whole training set in mini-batch to train the generator while the parameters of the discriminator are fixed. Then, we train the discriminator while fixing the parameters of the generator. The GAN-based training process of ACNE aims to make the generator search for high-quality negative communities as the inputs of training the discriminator. In addition, both the discriminator and generator are trained with Adam gradient descent [15] and L_2 regularization is applied to the parameters. When ACNE converges, we take the parameters learned by the discriminator as our final representations for vertices and communities. More details of experimental settings will be introduced in the next section.

Algorithm 1: Training Process of ACNE

Input: Graph $G = (V, E)$, number of community $|C|$, batch size $|\mathcal{B}|$, dimension d
Result: Parameters of Discriminator θ_D and Generator θ_G

begin
  Initialize θ_D and θ_G randomly;
  Generate walk sequences S via random walk based method [24];
  while *not converge* **do**
    Sample a batch $\mathcal{B}$ of walk sequences from S;
    for *G-steps* **do**
      Use G to sample negative communities for each target vertex v_t according to Eq. (5);
      Update the parameters of θ_G via policy gradient in Eq. (7);
    end
    for *D-steps* **do**
      For each target vertex v_t in sequence s of batch $\mathcal{B}$, calculate D loss w.r.t Eq. (9);
      Update the parameters of θ_D via gradient descent;
    end
  end
end

3 Experiments

In experiments, we evaluate the performance of vertex and community representations on real-world datasets with the tasks of vertex classification and overlapping community detection.

3.1 Datasets

We conduct experiments on four widely used network datasets with the statistics listed in Table 1.

Cora[5] is a research citation network constructed by [19]. It contains 2708 machine learning papers with 7 labels.

Citeseer[6] is another extensively adopted research paper set which contains 3264 publications and 6 labels.

Wiki[7] is a language network which contains 2405 web pages from 19 groups and 12761 edges between them. This dataset is firstly published from LBC[8] project and has been widely used for evaluating vertex classification tasks [39].

Table 1: Statistics of datasets

Datasets	$\|V\|$	$\|E\|$	$\|L\|$
Cora	2708	5278	7
Citeseer	3264	4551	6
Wiki	2405	12761	19
DBLP_C4	17725	52914	4

DBLP_C4[9] consists of bibliography data in computer science constructed by [30]. In the experiments, we select a list of conference papers from 4 research fields: database, data mining, AI, and CV.

3.2 Baseline Models

The descriptions of the baseline models can be divided into four groups as follows:

General network embedding. **DeepWalk** [24] is an efficient representation learning model by performing random walks on networks to generate vertex sequences and using the Skip-gram model [20] to learn vertex embeddings. **Line** [29] preserves the first-order and second-order proximity among vertices in networks. **Node2vec** [14] is an extension of DeepWalk by designing a biased random walk to explore the network structures. **SDNE** [32] firstly proposes a deep neural network to learn network embedding. **GraRep** [3] applies the SVD technique on k-step probability matrices to learn vertex embeddings and concatenates them as the global representations.

GAN-based network embedding. **AIDW** [6] is an inductive version of DeepWalk with GAN-based regularization method. **GraphGAN** [33] unifies the generative models and discriminative models of network embedding to boost the performance. **ARNE** [7] focuses on sampling high-quality negative vertices to achieve better results.

General community detection. **SCP** [17] detects communities by searching for adjacent cliques. **LC** [1] proposes to find link communities instead of vertices. **MDL** [22] proposes a minimum description length method to perform clustering. **BigCLAM** [40] uses non-

[5] https://people.cs.umass.edu/∼mccallum/data.html
[6] https://github.com/wonniu/AdvT4NE_WWW2019
[7] https://github.com/albertyang33/TADW
[8] https://linqs.soe.ucsc.edu/

[9] http://arnetminer.org/citation (V4 version is used)

Table 2: Accuracy (%) of vertex classification on Cora

% Label Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	64.60	69.85	74.21	76.68	77.59	77.68	78.63	79.35	79.23
Line	66.59	66.06	72.25	73.94	74.03	74.65	75.12	75.30	75.76
Node2vec	73.96	78.04	80.07	81.62	82.16	82.25	82.85	84.02	84.91
SDNE	70.97	75.08	76.90	77.82	78.26	79.11	79.37	79.46	79.37
GraRep	74.98	77.48	78.57	79.38	79.53	79.68	79.75	80.89	80.74
AIDW	73.83	77.93	79.43	81.16	81.79	82.27	82.93	84.11	83.69
GraphGAN	76.43	79.14	81.62	81.91	82.12	82.83	83.28	84.65	84.93
ARNE	68.09	72.86	75.14	75.83	76.97	77.30	79.22	78.90	78.43
MNMF	75.08	77.85	79.05	79.53	79.82	80.21	79.98	80.11	79.41
ComE	76.72	79.25	80.73	80.97	81.53	82.10	82.19	82.42	82.65
PolyDeepwalk	76.00	79.51	80.49	81.06	81.46	81.73	84.02	84.76	83.66
CNE	78.17	81.67	82.70	82.89	83.97	**84.68**	**85.91**	85.60	85.97
ACNE	**78.84**	**81.82**	**83.49**	**83.69**	**84.05**	84.59	85.85	**86.35**	**88.19**

Table 3: Accuracy (%) of vertex classification on Citeseer

% Label Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	45.53	50.98	53.79	55.25	56.05	56.84	57.36	58.15	59.11
LINE	47.03	50.09	52.71	53.52	54.20	55.42	55.87	55.93	57.22
node2vec	50.78	55.89	57.93	58.60	59.44	59.97	60.32	60.75	61.04
SDNE	47.35	51.10	52.45	53.20	53.70	54.20	54.79	55.26	54.46
GraRep	50.60	53.56	54.63	55.44	55.20	55.07	56.04	55.48	56.39
AIDW	50.77	54.82	56.96	58.04	59.65	60.03	60.99	61.18	62.84
GraphGAN	53.68	56.28	57.77	59.52	59.74	59.34	58.06	57.58	56.27
ARNE	54.12	56.09	56.46	56.92	56.99	57.81	57.55	56.97	52.60
MNMF	51.62	53.80	55.47	56.94	56.81	57.04	57.05	57.00	57.22
ComE	54.71	57.70	58.84	59.67	59.93	60.30	61.12	61.62	61.11
PolyDeepwalk	52.25	53.75	56.46	56.92	57.48	57.19	58.43	58.66	58.98
CNE	**56.33**	**59.57**	**62.88**	63.75	65.01	66.84	67.02	67.32	64.22
ACNE	55.34	58.46	62.80	**65.08**	**65.63**	**67.08**	**67.14**	**67.99**	**68.50**

negative matrix factorization to detect overlapping and hierarchically nested communities in massive networks. **NMF** [16] exploits non-negative matrix factorization to obtain vertex-community distribution in a global understanding of networks. We include NMF as one of the baseline models because we apply it to gaining prior knowledge for our model.

Jointly modeling. **MNMF** [35] employs a matrix factorization technique for jointly detecting non-overlapping communities and learning network representations. **ComE** [4] adopts multivariate Gaussian distributions to represent communities, which aims to learn network representation and overlapping communities jointly. **PolyDeepwalk** [18] proposes a polysemous embedding approach for modeling multiple facets of vertices by mapping each facet of a node into an embedding vector.

Besides, for ablation study, we take **CNE** as a variant of **ACNE** which omits the generator component in adversarial training and samples negative communities from a uniform distribution.

3.3 Parameter Settings and Evaluation Metrics

For the models requiring random walk preprocessing, we uniformly set the window size, the walk length, and the number of walks as 10, 30, and 50, respectively. Since the desirable representation dimension settings are not the same in different models, we adopt grid-search for searching their best performance by varying $d \in \{128, 200, 256, 300, 400\}$. For other parameters of models, we follow the preferred settings in their corresponding papers. In addition, in ACNE, we use Adam optimizer [15] with initial learning rate 1e-3. And λ in Eq. (9) is set to 1e-5. For vertex classification, we adopt Liblinear package [9] with default settings to build the classifier, and employ classification Accuracy [6] as metrics. For community detection, we use modified Modularity [42] which is specially designed for overlapping community detection tasks to evaluate the results.

3.4 Evaluation on Vertex Classification

As shown in Table 2, Table 3, and Table 4, we evaluate the classification accuracies of various models under different training ratios. For each ratio, we randomly select vertices as a training set and the remaining ones as a test set. Note that we exclude the baseline models in *general community detection* group because they are not designed for network embedding (we also omit NMF since its accuracy scores are much lower than other models). The highest scores

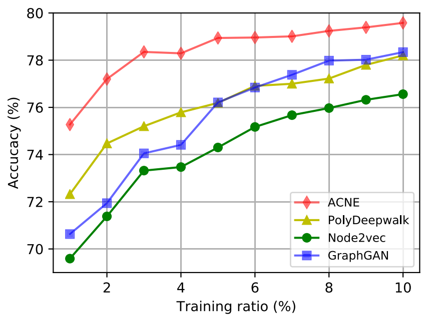


Figure 3: Vertex classification evaluation on DBLP_C4

Table 4: Accuracy (%) of vertex classification on Wiki

% Label Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	46.60	54.48	59.05	62.70	64.66	65.95	66.98	68.37	68.78
LINE	57.88	61.08	63.50	64.68	66.29	66.91	67.43	67.46	68.61
node2vec	55.94	59.67	61.11	64.21	65.08	65.58	66.76	67.19	68.73
SDNE	52.42	57.34	60.15	62.35	63.18	64.21	64.71	65.63	65.60
GraRep	58.57	61.91	63.58	63.77	64.68	65.39	65.92	65.18	67.05
AIDW	57.32	61.84	63.54	64.90	65.58	66.54	65.59	66.58	68.02
GraphGAN	57.97	62.57	63.79	65.39	66.01	66.67	67.83	68.23	68.87
ARNE	58.43	60.45	62.23	62.44	62.59	62.89	62.47	62.79	62.66
MNMF	54.76	58.82	60.43	61.66	62.74	63.23	63.46	63.45	64.77
ComE	59.11	62.46	64.38	65.45	65.98	67.38	67.49	67.92	67.89
PolyDeepwalk	56.44	61.90	62.59	63.76	64.09	64.35	64.96	65.90	66.49
CNE	58.89	63.38	**65.20**	**66.87**	67.31	67.85	68.28	69.23	68.46
ACNE	**59.68**	**63.41**	64.45	66.11	**67.41**	**68.09**	**68.98**	**69.65**	**69.71**

Table 5: Modularity of community detection results

Datasets	SCP	LC	MDL	BigCLAM	NMF	MNMF	ComE	PolyDeepwalk	**CNE**	**ACNE**
Cora	0.142	0.544	0.771	0.796	0.759	0.832	0.912	0.964	1.317	**1.334**
Citeseer	0.068	0.457	0.469	0.649	1.037	1.123	1.192	1.151	1.242	**1.276**
Wiki	0.113	0.480	0.550	0.522	1.303	1.413	1.614	1.576	1.705	**1.734**
DBLP_C4	0.067	0.586	0.685	0.616	0.651	0.687	0.701	0.693	0.757	**0.779**

are highlighted in boldface. From the above tables, we have the following observations:

(1) Our proposed models, ACNE and CNE, consistently outperform the state-of-the-art models on all datasets with different training ratios, which demonstrates the effectiveness of our adversarial training for incorporating overlapping community information into network representation learning.

(2) Specifically, ACNE achieves better performance than MNMF, ComE and PolyDeepwalk, although they jointly learn community structures and network embedding. It indicates that our ACNE can learn more discriminative representations with our proposed adversarial learning targets. Moreover, we also can see that ACNE has significant improvements compared with the GAN-based network embedding models (AIDW, GraphGAN, and ARNE) because they neglect community information in the learning process.

In addition, we utilize smaller training ratios on DBLP_C4 to accelerate the training speed of classifiers and evaluate the performance of ACNE under sparse scenes. The classification results are shown in Figure 3. Note that, for easy presentation, we only keep three models that have the best performance in DBLP_C4 from each baseline group. From Figure 3, we can see that ACNE still outperforms the baseline methods in the sparse situation.

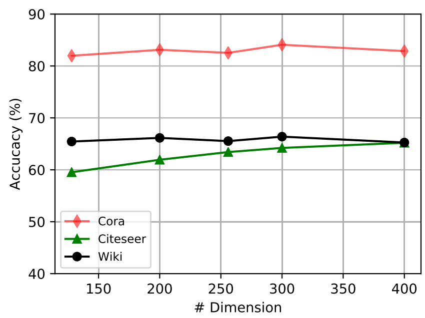
3.5 Evaluation on Community Detection

We evaluate the community detection performance of baseline models from *general community detection* group and *jointly modeling* group. From Table 5, we can obtain the following observations:

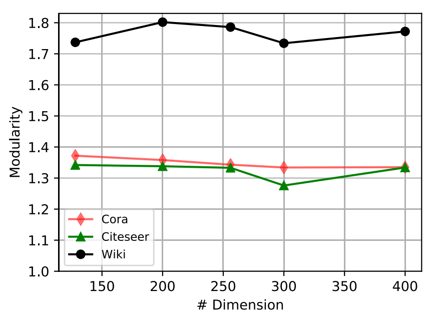
(1) ACNE significantly outperforms the baselines. It indicates that ACNE can learn more desirable community embeddings and detect meaningful communities, which also verifies the effectiveness of our proposed generator for generating high-quality negative communities in the adversarial learning.

(2) Moreover, ACNE performs better than NMF, which demonstrates that the superiority of ACNE comes beyond the prior knowledge obtained from NMF (we applied it to estimate the global communities contained in networks). The performance of ACNE also conforms to our assumption that the community assignment of each vertex is related to its context. Details are mentioned in Section 2.2.

In summary, all the network analysis tasks, including overlapping community detection and vertex classification, demonstrate the effectiveness of ACNE for incorporating GANs technique to jointly modeling community structures and vertex connectivity. Our model achieves consistent improvements comparing with the baselines.



(a) Influence of dimension setting on accuracy



(b) Influence of dimension setting on modularity

Figure 4: Parameter sensitivity analysis of dimension in ACNE

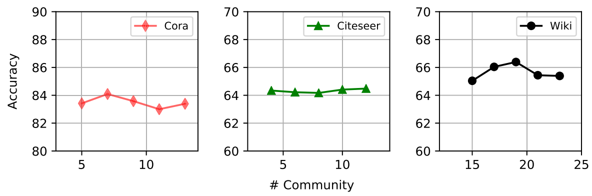


Figure 5: Influence of community setting on accuracy

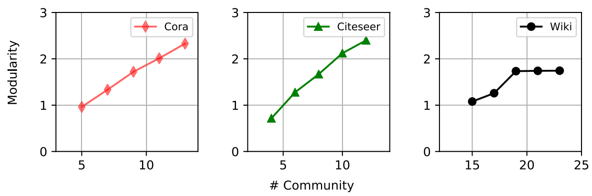


Figure 6: Influence of community setting on modularity

3.6 Parameter Sensitivity

In this part, we analyze the sensitivity of ACNE to the key parameters when conducting vertex classification and community detection tasks. Firstly, we try to investigate how the representation dimension d affects the performance of ACNE by varying its numbers in $\{128, 200, 256, 300, 400\}$. The experiment results are reported in Figure 4. For classification accuracy, as shown in Figure 4a, we can see that ACNE can achieve relatively stable performance on Cora, Citeseer, and Wiki datasets, respectively. Here, we report the average accuracy with training ratios from 10% to 90%. Note that the curve of Citeseer grows with dimension size in the beginning and becomes relatively smooth after 250. For modularity shown in Figure 4b, there are slight fluctuations in the curves of datasets (around 0.03 points in Cora, 0.05 points in Citeseer and 0.08 points in Wiki).

Then, we want to estimate how the setting of community number C affects the performance of ACNE. We test C in $\{5, 7, 9, 11, 13\}$, $\{4, 6, 8, 10, 12\}$, and $\{15, 17, 19, 21, 23\}$ on Cora, Citeseer, and Wiki, respectively. From Figure 5, we can see that ACNE can achieve the best performance when the community numbers set to 7 and 19 on Cora and Wiki. And ACNE can obtain stable performance on Citeseer when the number is set to around 7. These detected numbers are matched the ground truth of total numbers of communities in the datasets, which indicates that ACNE can dynamically detect the community number in the network. Besides, from Figure 6, we can see that the modularity curves increase with community number. One possible reason is that we can discover more fine-grained communities when increasing C. However, the accuracy results will decrease as shown in Figure 5, because ACNE may fall into local optimum when learning the vertex representations. Therefore, it is a trade-off between detecting more fine-grained communities and learning better network embedding.

4 Related Work

Since we propose an adversarial learning method for incorporating overlapping community information into network embedding, our work can be categorized into network representation learning and overlapping community detection.

Network representation learning (NRL), i.e., network embedding, has received a lot of attention in recent years. Specifically, inspired by word2vec [21], DeepWalk proposed by Perozzi et al. [24] performs truncated random walks to generate node sequences which are treated as sentences and fed into Skip-gram model [20] to learn representations. Then, Tang et al. propose LINE [29] which extends DeepWalk by employing breadth-first and depth-first graph search strategies to optimize proximities. After that, Node2vec [14] is proposed to take these two strategies into consideration and designs a biased random walk procedure to explore diverse neighborhoods. More recently, generative adversarial networks (GANs) [12] have presented promising performances in a wide variety of tasks in various applications [36]. Inspired by the development of GANs, some GAN-based models are proposed for NRL. GraphGAN [33] firstly unifies generative and discriminative models of vertices to boost embedding performance. AIDW [6] proposes an inductive version of DeepWalk which utilizes the adversarial technique to regularize the learned representation. ARNE [7] leverages GANs to sample high-quality negative vertices to facilitate network embedding. However, all these models only focus on learning the local connectivity of neighbor vertices but ignore global patterns which are known as communities in many complex networks.

Community detection is a critical task in social science [10]. Traditional detection only focuses on detecting non-overlapping communities, which may not conform to real-world scenarios. For example, each vertex may belong to different communities when it plays multiple roles. Therefore, models for overlapping community detection are proposed. SCP [17] proposes a sequential algorithm for fast overlapping community detection. LC [1] exploits a link clustering based detection algorithm by partitioning the links instead of vertices. MDL [22] utilizes a minimum description length method to detect overlapping groups. BigCLAM [40] and NMF [16] both employ non-negative matrix factorization to obtain vertex-community strength vectors and assign communities to vertices according to the learned vectors. Nevertheless, these methods neglect the local connectivity of vertices and are not designed for NRL.

Jointly modeling aims to perform the above two tasks in one model. For instance, community affiliation based algorithms, such as MNMF [35], employ a matrix factorization technique for jointly detecting communities and learning network representations. Besides, ComE [4] adopts multivariate Gaussian distributions to represent communities for unified learning. PolyDeepwalk [18] proposes a polysemous embedding approach for modeling multiple facets (i.e., communities) of vertices by mapping each node facet into a vector.

5 Conclusion

In this paper, we propose an adversarial training method called ACNE for overlapping community detection and network embedding. Current GAN-based NRL methods only adopt standard GANs to generate samples of vertices instead of communities. How to incorporate community information into GANs is still a challenge. In ACNE, we firstly obtain different paths expanding from a vertex to its n-step neighbors which may represent the expression of its communities. Then, we sample a community for each vertex with a context-aware community assignment method. Meanwhile, we leverage a softmax generator for generating high-quality negative communities. Lastly, we design a discriminator to jointly learn the vertex and community representations. Experimental results demonstrate that ACNE can achieve better performance than state-of-the-art models.

ACKNOWLEDGEMENTS

MOST (2019YFB1600704), FDCT (SKL-IOTSC-2018-2020, FDCT/0045/2019/A1), GSTIC (EF005/FST-GZG/2019/GSTIC), University of Macau (MYRG2017-00212-FST, MYRG2018-00129-FST).

REFERENCES

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann, 'Link communities reveal multiscale complexity in networks', *nature*, **466**(7307), 761, (2010).

[2] Liwei Cai and William Yang Wang, 'Kbgan: Adversarial learning for knowledge graph embeddings', *arXiv preprint arXiv:1711.04071*, (2017).

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

[4] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria, 'Learning community embedding with community detection and node embedding on graphs', in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 377–386. ACM, (2017).

[5] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu, 'A survey on network embedding', *IEEE Transactions on Knowledge and Data Engineering*, **31**(5), 833–852, (2018).

[6] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang, 'Adversarial network embedding', in *Thirty-Second AAAI Conference on Artificial Intelligence*, (2018).

[7] Quanyu Dai, Qiang Li, Liang Zhang, and Dan Wang, 'Ranking network embedding via adversarial learning', in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 27–39. Springer, (2019).

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

[9] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, 'Liblinear: A library for large linear classification', *Journal of machine learning research*, **9**(Aug), 1871–1874, (2008).

[10] Santo Fortunato, 'Community detection in graphs', *Physics reports*, **486**(3-5), 75–174, (2010).

[11] Hongchang Gao, Jian Pei, and Heng Huang, 'Progan: Network embedding via proximity generative adversarial network', in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1308–1316. ACM, (2019).

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

[13] Thomas L Griffiths and Mark Steyvers, 'Finding scientific topics', *Proceedings of the National academy of Sciences*, **101**(suppl 1), 5228–5235, (2004).

[14] Aditya Grover and Jure Leskovec, 'node2vec: Scalable feature learning for networks', in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, (2016).

[15] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, (2014).

[16] Da Kuang, Chris Ding, and Haesun Park, 'Symmetric nonnegative matrix factorization for graph clustering', in *Proceedings of the 2012 SIAM international conference on data mining*, pp. 106–117. SIAM, (2012).

[17] Jussi M Kumpula, Mikko Kivelä, Kimmo Kaski, and Jari Saramäki, 'Sequential algorithm for fast clique percolation', *Physical Review E*, **78**(2), 026109, (2008).

[18] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu, 'Is a single vector enough? exploring node polysemy for network embedding', *arXiv preprint arXiv:1905.10668*, (2019).

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

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781*, (2013).

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

[22] Tiago P Peixoto, 'Model selection and hypothesis testing for large-scale network models with overlapping groups', *Physical Review X*, **5**(1), 011033, (2015).

[23] Adam Perer, Ido Guy, Erel Uziel, Inbal Ronen, and Michal Jacovi, 'Visual social network analytics for relationship discovery in the enterprise', in *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 71–79. IEEE, (2011).

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

[25] Anand Rajaraman and Jeffrey David Ullman, *Mining of massive datasets*, Cambridge University Press, 2011.

[26] Joseph Reisinger and Raymond J Mooney, 'Multi-prototype vector-space models of word meaning', in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 109–117. Association for Computational Linguistics, (2010).

[27] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel, 'Gradient estimation using stochastic computation graphs', in *Advances in Neural Information Processing Systems*, pp. 3528–3536, (2015).

[28] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour, 'Policy gradient methods for reinforcement learning with function approximation', in *Advances in neural information processing systems*, pp. 1057–1063, (2000).

[29] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei, 'Line: Large-scale information network embedding', in *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, (2015).

[30] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su, 'Arnetminer: extraction and mining of academic social networks', in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998. ACM, (2008).

[31] Cunchao Tu, Xiangkai Zeng, Hao Wang, Zhengyan Zhang, Zhiyuan Liu, Maosong Sun, Bo Zhang, and Leyu Lin, 'A unified framework for community detection and network representation learning', *IEEE Transactions on Knowledge and Data Engineering*, (2018).

[32] Daixin Wang, Peng Cui, and Wenwu Zhu, 'Structural deep network embedding', in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234. ACM, (2016).

[33] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo, 'Graphgan: Graph representation learning with generative adversarial nets', in *Thirty-Second AAAI Conference on Artificial Intelligence*, (2018).

[34] Peifeng Wang, Shuangyin Li, and Rong Pan, 'Incorporating gan for negative sampling in knowledge representation learning', in *Thirty-Second AAAI Conference on Artificial Intelligence*, (2018).

[35] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang, 'Community preserving network embedding', in *Thirty-First AAAI Conference on Artificial Intelligence*, (2017).

[36] Zhengwei Wang, Qi She, and Tomas E Ward, 'Generative adversarial networks: A survey and taxonomy', *arXiv preprint arXiv:1906.01529*, (2019).

[37] Ronald J Williams, 'Simple statistical gradient-following algorithms for connectionist reinforcement learning', *Machine learning*, **8**(3-4), 229–256, (1992).

[38] Wei Xu, Xin Liu, and Yihong Gong, 'Document clustering based on non-negative matrix factorization', in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 267–273. ACM, (2003).

[39] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang, 'Network representation learning with rich text information', in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, (2015).

[40] Jaewon Yang and Jure Leskovec, 'Community-affiliation graph model for overlapping network community detection', in *2012 IEEE 12th international conference on data mining*, pp. 1170–1175. IEEE, (2012).

[41] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu, 'Seqgan: Sequence generative adversarial nets with policy gradient', in *Thirty-First AAAI Conference on Artificial Intelligence*, (2017).

[42] Hongyi Zhang, Irwin King, and Michael R Lyu, 'Incorporating implicit link preference into overlapping community detection', in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, (2015).