Graph Game Embedding

Xiaobin Hong[1*], Tong Zhang[1*], Zhen Cui[1†], Yuge Huang[2], Pengcheng Shen[2], Shaoxin Li[2], Jian Yang[1]

School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China.
[2]Youtu Lab, Tencent
{xbhong, tong.zhang, zhen.cui}@njust.edu.cn, {yugehuang, quantshen, darwinli}@tencent.com, csjyang@njust.edu.cn

Abstract

Graph embedding aims to encode nodes/edges into low-dimensional continuous features, and has become a crucial tool for graph analysis including graph/node classification, link prediction, etc. In this paper we propose a novel graph learning framework, named graph game embedding, to learn discriminative node representation as well as encode graph structures. Inspired by the spirit of game learning, node embedding is converted to the selection/searching process of player strategies, where each node corresponds to one player and each edge corresponds to the interaction of two players. Then, a utility function, which theoretically satisfies the Nash Equilibrium, is defined to measure the benefit/loss of players during graph evolution. Furthermore, a collaboration and competition mechanism is introduced to increase the discriminant learning ability. Under this graph game embedding framework, considering different interaction manners of nodes, we propose two specific models, named paired game embedding for paired nodes and group game embedding for group interaction. Comparing with existing graph embedding methods, our algorithm possesses two advantages: (1) the designed utility function ensures the stable graph evolution with theoretical convergence and Nash Equilibrium satisfaction; (2) the introduced collaboration and competition mechanism endows the graph game embedding framework with discriminative feature leaning ability by guiding each node to learn an optimal strategy distinguished from others. We test the proposed method on three public datasets about citation networks, and the experimental results verify the effectiveness of our method.

Introduction

In recent years, graph has drawn increasing attention due to its potential applications to ubiquitous irregular data such as protein data (Fout et al. 2017; Borgwardt et al. 2007) and social network (Cai, Zheng, and Chang 2018; Silva et al. 2010; Orsini, Baracchi, and Frasconi 2018; Wang, Cui, and Zhu 2016; Yu et al. 2018). A graph is composed of a set of nodes and a set of edges. The nodes often represent entities in real world while the edges indicate connection relationship between node pairs. In many graph-related tasks, graph embedding is one of most fundamental topics. It aims to learn effective representation of nodes by using graph topological structures or other characteristics (e.g., node attributes). Nowadays, graph embedding has become a crucial graph analysis technology in multiple practical applications, including user recommendation system (Silva et al. 2010), social network analysis (Orsini, Baracchi, and Frasconi 2018), face recognition (Cheng et al. 2019; Jiang et al. 2017) and protein function prediction (Borgwardt et al. 2007).

Various graph embedding methods have been proposed in previous literatures, including supervised methods(Kipf and Welling 2017; Veličković et al. 2018; Zhuang and Ma 2018) and unsupervised methods(Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Veličković et al. 2019). The unsupervised methods attempt to learn the embeddings of nodes so that topological structure of graph is preserved as much as possible in the embedding space. As the learning process is task-independent, the embedding features can well favor many downstream tasks, e.g., often used as input for the supervision learning case. Recently, graph convolution network (GCN) (Kipf and Welling 2017) arouses researchers' enthusiasm for graph embedding especially in the semi-supervised case. Although considerable progress has been achieved in both unsupervised and semi-supervised cases, there still lacks an efficient mechanism to mine potential cues on those unlabelled nodes or fully unlabelled graph. Most methods aim to preserve the connection relationship of edges during the embedding process. But the discriminability of node embeddings is as important as structure preservation in many popular pattern classification tasks.

Inspired by the spirit of game learning (Roughgarden 2010), in this work, we propose a novel graph embedding framework named graph game embedding to learn more discriminative embeddings as well as encode graph structures. Each node is treated as one game player, and one edge correlates the interaction of two players during game learning. Thus node embedding learning is converted to the selection/searching process of player strategies. To optimize node embedding, we construct a utility function to measure the benefit-loss of nodes. This utility function is designed to satisfy the Nash Equilibriums condition, which well guides graph evolution to good convergence with a theoretical guarantee. Further, we introduce the collaboration and competition modes to increase the discriminant learning ability. In view of different interaction manners, we propose two spe-

[*]Equal contribution
[†]Corresponding author
Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

cific embedding models under the graph game embedding framework, named paired game embedding for paired nodes and group game embedding for group interaction. The advantages of our proposed game strategy are two folds: (1) theoretical guarantee of convergence: our designed utility function theoretically satisfies the Nash Equilibrium, which ensures the stable graph evolution with good convergence; (2) discriminative feature leaning ability: the introduced collaboration and competition mechanism in game may guide each node to evolve into an optimal strategy distinguished from others, which results in discriminative node representation. To evaluate the performance, we test our proposed framework on three public graph structural datasets of citation networks. In this process, our proposed framework is applied to fulfill node classification tasks in both unsupervised and semi-supervised manners, and the performances are comprehensively compared with existing state-of-the-art methods. The experimental results verify the effectiveness of our method. To summarize, the main contributions of this paper are as follows:

- We proposed a novel and effective graph game embedding framework for both unsupervised and semi-supervised graph embedding by introducing game theory into graphs, which may open up a new perspective for graph embedding technology.
- The designed utility function satisfies the Nash Equilibrium with theoretical guarantee, which well guides the graph to evolve stably until a good convergence.
- The introduced collaboration and competition mechanism makes the strategy of each node distinguished from others, which results in discriminative node embedding.
- We comprehensively evaluate our algorithm on multiple graph-structural datasets, and the results demonstrate the state-of-the-art performance.

Related Work

In this section, we briefly overview existing works related to our proposed framework, including graph embedding learning methods and game theory based algorithms.

Graph Embedding Learning

Graph embedding aims to map a D-dimensional graph into a d-dimensional continuous space by considering local graph structure, where generally $d \ll D$. The motivation of graph embedding learning is to make those originally connected nodes be still close in the embedded space. Thus, graph embedding learning may provide robust node embeddings for downstream machine learning tasks such as node classification. According to different embedding manners, graph embedding algorithms can be roughly divided into three categories: factorization based, random walk based and deep learning based methods. Factorization based methods obtain graph embedding by factorizing the adjacent matrices. For instance, Laplacian Eigenmaps(Belkin and Niyogi 2002) aims to keep the embedding of two nodes close if their connection strength is strong. Besides, Graph Factorization(Ahmed et al. 2013), GraRep(Cao, Lu, and Xu 2015) and HOPE(Ou et al. 2016) are also representative factorization based methods. Random walk based methods obtain the graph embedding by learning from generated paths where different walk strategies have been employed, e.g. the first order random walk in DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and the second order partial random walk in node2vec (Grover and Leskovec 2016). On the other hand, the growing research on deep learning has led a deluge of deep neural networks based methods to be applied to graphs. For example, SDNE(Wang, Cui, and Zhu 2016) and SDGR(Cao, Lu, and Xu 2016) utilize deep auto-encoder to generate the embedding models that can capture non-linearity in graphs, while GCN (Kipf and Welling 2017) defines a convolution operator with powerful feature representation ability by diffusing information among connected nodes on graphs.

Game Theory

In recent years, game theory has been widely used in biology (Hauert, Holmes, and Doebeli 2006), economics (Magurran and Dugatkin 1998), international relations (Ekman 2001), computer science (Van Rooij 2007), political science (Philippe 1994), military strategy (Eguiluz and Tessone 2009), and many other fields. Generally speaking, game theory can be seen as a theory of quantitative analysis for the study of conditions and mediums, which results in the unification and harmonization of conflicts(Myerson 2013). Classical game theory can be classified into two categories: non-cooperative and cooperative games(Akkarajitsakul et al. 2011). Due to the simplicity in modeling and solving, non-cooperative games have often been used in resource competition and detection of attackers(Oulaourf et al. 2017; Moura and Hutchison 2017). On the other hand, cooperative games have been applied to resource allocation or issue sharing in virtual networks(Manshaei et al. 2013; Akkarajitsakul et al. 2011). Nash Equilibrium (NE)(Nash 1950) is a very important concept in game theory in which condition no player can benefit by changing strategies if the other players keep theirs unchanged. And it has also been applied in steady-state searching of network evolution (Santos, Rodrigues, and Pacheco 2005). The study of network evolutionary games begins with the study of Prisoner's Dilemma game on two-dimensional squares of Nowak and May (Nowak and May 1992). In recent years, with the booming of complex networks, game research on networks has attracted extensive attention and made huge progress (Szabo and Fath 2007; Perc and Szolnoki 2010).

Besides, our method belongs to unsupervised learning, in which those contrastive learning (Hadsell, Chopra, and LeCun 2006; He et al. 2020; Arora et al. 2019; Chen et al. 2020; Kipf, van der Pol, and Welling 2019) methods are related to ours. Contrastive learning methods usually learn representation by encoding what makes two things similar or different. There are **two main differences** between graph game learning and contrastive learning. First, contrastive learning does not consider the structural relationship between samples, but only focuses on the similarities and differences of the samples themselves. Graph game learning makes full use of topological relationship between nodes during representation learning. Second, graph game learning can reach the Nash

Equilibrium point (please see the section named Theoretical Analysis), which thus has a good theoretical guarantee, but contrastive learning only learns representation by simply optimizing reconstruction error, which lacks a strong support of theory. From the perspective of information theory, game learning has a broader category than contrastive learning, so our method is more general for unsupervised learning.

Problem Description

A graph can be denoted with a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of N nodes (i.e., $|\mathcal{V}| = N$) and $\mathcal{E}$ is a set of M edges (i.e., $|\mathcal{E}| = M$). Each edge indicates the connection of a pair of nodes. Here we study undirected graphs whose edges are undirected, but our method could be extended to directed graphs. In an undirected graph, each node might be endowed with an attribute vector or an initial state vector. If exists, we denote the initial state vector of node v_i with $\widetilde{\mathbf{x}}_i \in \mathbb{R}^d$. All edge connection relationships can be represented with an adjacent matrix $\mathbf{A} \in \{0,1\}^{N\times N}$. If the two nodes v_i and v_j are connected, $A_{ij} = 1$, otherwise 0. Sometimes one edge might be given a connection strength or connection probability. At this time, the adjacent relationship becomes $A_{ij} \in \mathbb{R}$, i.e., $\mathbf{A} \in \mathbb{R}^{N\times N}$. Graph embedding is to learn node representations by preserving topological structures as well as taking advantage of other properties.

Inspired by the spirit of game learning, we treat one node v_i of a graph as one player, and the embedding learning on nodes as the strategy selection for players. At one stage, one player chooses a strategy for next action based on the current situation. All possible strategies of the i-th player are collected to form a strategy set/space $\mathcal{S}_i$, which may be discrete or continuous. One player may definitely choose an certain element $s_i \in \mathcal{S}_i$, named pure strategy, or probabilistically choose each strategy in $\mathcal{S}_i$, name mixed strategy. In game learning, one player determines the next strategy according to the state of the player itself as well as the opponent states. For graph embedding learning, analogically, one node should update its representation based on the node's state itself and its contextual states. Viewed as game players, graph nodes collaborate and compete with each other for optimal potentials. It means that the states of nodes can be evolved into an balance situation through game learning. Hence, graph embedding can be framed into game learning on graph, where nodes update their states through collaboration and competition on each other. In the following section, we introduce our proposed method from a new perspective of game learning.

Graph Game Embedding

In this section, we first introduce how to infer states/strategies[1] of nodes, next pose two embedding frameworks including paired game case and group game case, and a parameterized method for graph embedding, finally summarize into algorithms and analyze the convergence as well as complexity.

[1]Due to the clear context, we sometimes do not distinguish some concepts such as "node" versus "player" and "state" versus "strategy" in the presentation.

Strategy Inference

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define a latent embedding space $\mathcal{S}_i \subseteq \mathbb{R}^d$ (i.e., a strategy space) for each node v_i. When one strategy is chosen, there returns a positive/negative reward, which could be used for constraining graph learning. To characterize rewards, we may define a utility function U_i for each node v_i. Thereafter, we can represent one game graph with a triple tuple $\Gamma_{\mathcal{G}} = (\mathcal{G}, \{\mathcal{S}_i | v_i \in \mathcal{V}\}, \{U_i | v_i \in \mathcal{V}\})$, where i) $\mathcal{G}$ is a plain graph defined above, and each node corresponds to one player, ii) $\mathcal{S}_i$ is the strategy/state space of the i-th node, and iii) U_i is a utility function to map a strategy list into the corresponding pay-off of the i-th player, defined as $U_i : \Pi_{v_j \in \mathcal{V}_{-i}} \mathcal{S}_j \times \mathcal{S}_i \to \mathbb{R}$ with $\mathcal{V}_{-i} = \mathcal{V} - \{v_i\}$.

Suppose the graph $\mathcal{G}$ is embedded in a d-dimensional space, each node v_i has its strategy space $\mathcal{S}_i \in \mathbb{R}^d$. In graph embedding, the strategy spaces of all nodes are often assumed to be equal, i.e., $\mathcal{S}_1 = \mathcal{S}_2 = \cdots = \mathcal{S}_N \in \mathbb{R}^d$. Our aim is to find a strictly dominant strategy list $\mathbf{s}^* = (\mathbf{s}_1^*, \mathbf{s}_2^*, \cdots, \mathbf{s}_N^*)$, where $\mathbf{s}_i^* \in \mathcal{S}_i,\ i = 1, \cdots, N$. A strategy list is strictly dominant, if and only if this strategy list satisfies the following condition on node utilities,

$$U_i(\mathbf{s}_i^*, \mathbf{s}_{-i}^*) \geq U_i(\mathbf{s}_i', \mathbf{s}_{-i}^*), \quad \forall \mathbf{s}_i' \in \mathcal{S}_i,\ \forall v_i \in \mathcal{V}. \tag{1}$$

where the symbol $-i$ denotes a set of all indices except the index i, i.e., $-i = \{1, 2, \cdots, i-1, i+1, \cdots, N\}$. At this time, the strategy list $\mathbf{s}^*$ is a Nash Equilibrium point during game learning. To obtain a Nash Equilibrium point, we can use the best-response method that maximizes the utility of each node. The best-response strategy is defined as a set mapping $\mathcal{B}_i(\mathbf{s}_{-i}) : \mathcal{S}_{-i} \to \mathcal{S}_i$, formally,

$$\mathcal{B}_i(\mathbf{s}_{-i}) = \{\mathbf{s}_i^* | \mathbf{s}_i^* = \arg\max_{\mathbf{s}_i \in \mathcal{S}_i} U_i(\mathbf{s}_i, \mathbf{s}_{-i})\}, \forall v_i \in \mathcal{V}. \tag{2}$$

In this case, the strategy selected by each node should be given the best response to all possible strategies of other nodes. The solution of the above formula is a fixed point in Nash Equilibrium. However, the best response strategy often takes high computational cost because all strategies need to be considered. Instead we use the better-response method:

$$\begin{aligned} &\mathcal{B}_i'(\mathbf{s}) = \{\mathbf{s}_i' | \mathbf{s}_i' \in \mathcal{S}_i, U_i(\mathbf{s}_i', \mathbf{s}_{-i}) \geq U_i(\mathbf{s}_i, \mathbf{s}_{-i})\}, \\ &\mathcal{B}_i'(\mathbf{s}) : \mathcal{S} \to \mathcal{S}_i,\ \forall v_i \in \mathcal{V}. \end{aligned} \tag{3}$$

At each time step, one node chooses a proper strategy to improve its utility, while the states of other nodes are preserved invariant. Thus node strategies could be iteratively optimized to reach a stable state. Formally, at the $(t+1)$-th step, we may infer a better-response strategy for each node v_i,

$$\mathbf{s}_i(t+1) \in \mathcal{B}_i'(\mathbf{s}(t)). \tag{4}$$

Paired Game Embedding

Definition 1 (Pair Graph Game) *For a graph game with the triple tuple $\Gamma_{\mathcal{G}} = (\mathcal{G}, \{\mathcal{S}_i | v_i \in \mathcal{V}\}, \{U_i | v_i \in \mathcal{V}\})$, the subject of the game is the node in the graph. For any $v_i \in \mathcal{V}$, its utility function satisfies: $U_i(s) = \sum_j u(s_i, s_j)$. The graph game called Pair Graph Game.*

In the game graph $\Gamma_{\mathcal{G}}$, each edge connects an interaction of two involved nodes. During the learning of graph embedding, pairwise nodes are required to collaborate/compete with each other to achieve a certain stability. We call such an paired-node game mode on graph as paired game embedding. In order to conduct cooperative and competitive games in graphs, the nodes in one graph are divided into different groups. In view of graph topological structures, we partition the node set $\mathcal{V}$ of one graph into two groups with respect to each reference node v_i: $\mathcal{N}(v_i)$ and $\overline{\mathcal{N}}(v_i)$. $\mathcal{N}(v_i)$ is a set of nodes adjacent to the node v_i, and $\overline{\mathcal{N}}(v_i) = \mathcal{V} - \{v_i\} - \mathcal{N}(v_i)$. We name those adjacent nodes $\mathcal{N}(v_i)$ as collaborative players, while the non-adjacent nodes $\overline{\mathcal{N}}(v_i)$ as competitive players. Accordingly, the utility function of each node consists of collaboration utility and competition utility, which are defined as follows

$$\begin{aligned} U_i^+(\mathbf{s}_i, \mathbf{s}_{\mathcal{N}(v_i)}) &= \sum_{v_j \in \mathcal{N}(v_i)} A_{ij} \times u(\mathbf{s}_i, \mathbf{s}_j), \\ U_i^-(\mathbf{s}_i, \mathbf{s}_{\overline{\mathcal{N}}(v_i)}) &= \sum_{v_k \in \overline{\mathcal{N}}(v_i)} \overline{A}_{ik} \times u(\mathbf{s}_i, \mathbf{s}_k). \end{aligned} \tag{5}$$

where A_{ij} denotes the weight of the edge bridging nodes i and j, $\overline{A}_{ij}$ denotes an unreachable probability between nodes i and j, and u is a paired-node utility function. The computation of unreachable matrix $\overline{A}_{ij} = 1 - \sum_{k=0}^{K-1} A_{ij}^{(k)}$, where $A^{(k)}$ is the reachable probability when randomly walking k steps, and K is the receptive field size. To reduce the affect of scales, the adjacent weights of each node are often normalized to $\sum_j A_{ij} = 1$. The cooperation utility of Eqn. 5 preserves topological structures of graph, while the competition utility increases the discriminability of embeddings. Concretely, we instantiate the utility function as follows

$$u(\mathbf{s}_i, \mathbf{s}_j) = \exp \frac{\langle \mathbf{s}_i, \mathbf{s}_j \rangle}{\|\mathbf{s}_i\| \|\mathbf{s}_j\|}, \quad \forall v_i \in \mathcal{V}. \tag{6}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product operation on two vectors.

To overcome the scale affect, we define the total utility function for node v_i as

$$U_i(\mathbf{s}) = U_i^+(\mathbf{s}_i, \mathbf{s}_{\mathcal{N}(v_i)}) / U_i^-(\mathbf{s}_i, \mathbf{s}_{\overline{\mathcal{N}}(v_i)}). \tag{7}$$

As the denominator might be close to zero, we formulate the object function as a relative logarithm ratio operation,

$$\begin{aligned} \arg\max_{\mathbf{s}_i} \quad & \ell(\mathbf{s}_i) = \log U_i^+(\mathbf{s}_i, \mathbf{s}_{\mathcal{N}(v_i)}) \\ & - \log(U_i^+(\mathbf{s}_i, \mathbf{s}_{\mathcal{N}(v_i)}) + U_i^-(\mathbf{s}_i, \mathbf{s}_{\overline{\mathcal{N}}(v_i)})). \end{aligned} \tag{8}$$

Group Game Embedding

Definition 2 (Group Graph Game) *For a graph game with the triple tuple $\Gamma_{\mathcal{G}} = (\mathcal{G}, \{\mathcal{S}_i | v_i \in \mathcal{V}\}, \{U_i | v_i \in \mathcal{V}\})$, if the game object of one node is the aggregation of its neighborhood, that is to say, its utility function satisfies: $U_i(s) = u(s_i, AGG_{j \in \mathcal{N}_i} s_j)$, The graph game called Group Graph Game.*

Different from paired game embedding, the group game method is to take a group of nodes as one game player. In contrast to Eqn. (6), the utility function can be redefined as

$$\begin{aligned} u(\mathbf{s}_i, \mathbf{s}_g) &= \exp \frac{\langle \mathbf{s}_i, \mathbf{s}_g \rangle}{\|\mathbf{s}_i\| \|\mathbf{s}_g\|}, \\ \mathbf{s}_g &= \sum_{v_j \in \mathcal{V}_g} W_{ij} \times \mathbf{s}_j. \end{aligned} \tag{9}$$

where $\mathcal{V}_g \subseteq \mathcal{V}$ is a specified group set, and W_{ij} is the weight coefficient between node i and j. Analogical to paired game embedding, we can construct a collaboration group set $\mathcal{N}(v_i)$ and a competition group set $\overline{\mathcal{N}}(v_i)$ for each node v_i. For the weight matrix, we may set $W_{ij} = A_{ij}$ in collaboration, and $W_{ij} = \overline{A}_{ij}$ in competition. The objective function becomes to maximize the following formula,

$$\begin{aligned} \arg\max_{\mathbf{s}_i} \quad & \ell(\mathbf{s}_i) = \log U_i(\mathbf{s}_i, \mathbf{s}_{g_1}) \\ & - \log(U_i(\mathbf{s}_i, \mathbf{s}_{g_1}) + U_i(\overline{\mathbf{s}}_i', \mathbf{s}_{g_2})), \\ \text{s.t.}, \quad & \mathbf{s}_{g_1} = \sum_{v_j \in \mathcal{V}(v_i)} A_{ij} \times \mathbf{s}_j, \\ & \mathbf{s}_{g_2} = \sum_{v_k \in \overline{\mathcal{V}}(v_i)} \overline{A}_{ik} \times \mathbf{s}_k. \end{aligned} \tag{10}$$

Parameterized Learning

For the given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each strategy $\mathbf{s}_i$ is derived through parameterized learning. Similar to graph convolution (Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017), we assume that each player makes decisions by considering the states of the neighbours. It means that the strategy is calculated by aggregating the neighbour information. Formally, for the i-th node, the strategy is computed by:

$$\mathbf{s}_i = \text{Agg}|_{k=1}^{K}(\sigma_1(\sum_{v_j \in \mathcal{N}(v_i)} A_{ij} \mathbf{W}^k \widetilde{\mathbf{x}}_j + \mathbf{b}^k)). \tag{11}$$

where $\widetilde{\mathbf{x}}_j$ is the initial state of node v_j, Agg is an aggregation operation (such as concatenation or average) on multiple responses, $\mathbf{W}^k \in \mathbb{R}^{d' \times d}, \mathbf{b}^k$ is the k-th filter and the corresponding bias, K denotes the number of filters, σ_1 is a non-linear activation function and assigned to Relu here.

Thus graph embedding is reduced to learn these filter parameters $\{\mathbf{W}^k, \mathbf{b}^k\}|_{k=1}^{K}$. It may be implemented with one module of network, and we can further stake multiple such modules to form deep network architecture.

Algorithms

Both algorithms of paired game embedding and group game embedding are summarized in Algorithm 1. Given one node, the number of competition nodes (i.e., negative examples) is huge. To reduce the computation cost, we randomly sample some of them as negative examples as illustrated in step 6. In step 8, we solve the object function in Eqn. (8) for paired game embedding, or Eqn. (10) for group game embedding by using the gradient descent method. In practice, due to the iterative batch processing, we need not to solve their optimal

Algorithm 1 Graph Game Embedding Algorithm

Input: Input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes.
Output: Graph embeddings $\mathbf{s}^*$.

1: **Initialization**: initialize network parameters and node states.
2: **for** batch processing **do**
3: Randomly sample a subset $\mathcal{V}' \subseteq \mathcal{V}$ of batch-size nodes to be updated.
4: **for** $v_i \in \mathcal{V}'$ **do**
5: Collect the adjacent node set $\mathcal{N}(v_i)$ for collaboration by using edge connections in $\mathcal{E}$.
6: Randomly sample a subset $\overline{\mathcal{V}}(v_i)$ from nonadjacent nodes as competition nodes.
7: Infer the current states of nodes in $\{v_i\} \cup \mathcal{N}(v_i) \cup \overline{\mathcal{V}}_i$ by parameterized models in subsection named Parameterized Learning.
8: Solve Eqn. (8) for paired game embedding or Eqn. (10) for group game embedding by using gradient descent method. i.e., $\Theta \leftarrow \Theta + \delta \nabla_\Theta \ell$, where $\Theta = \{\mathbf{W}^k, \mathbf{b}^k\}|_{k=1}^K$ is the model parameter set.
9: **end for**
10: **end for**

values for Eqn. (8) or Eqn. (10), but only need to update them by one-time gradient descent. The computation mainly costs in forward inference and gradient computation in the steps 7∼8. Considering one-time update of one node with one layer network, the computation complexity is $\mathcal{O}((|\mathcal{N}(v_i)| + |\overline{\mathcal{N}}(v_i)|)Kdd')$, where K is the number of filters, d, d' are the dimension numbers of input and output, $|\mathcal{N}(v_i)| \ll N$ and $|\overline{\mathcal{N}}(v_i)| \ll N$ due to the sparsity of edges and sampling strategy of negative examples respectively.

Convergence Analysis Here we only analyze the case of paired game embedding, because group game embedding has a similar derivation. At the t-th step, suppose the state $\mathbf{s}_i$ of node v_i will be updated. After the gradient descent step, $\Theta \leftarrow \Theta + \delta \nabla_\Theta \ell(\mathbf{s}_i)$, we can infer the new state $\mathbf{s}_i'$ for node v_i, and the loss should satisfy $\ell(\mathbf{s}_i') \geq \ell(\mathbf{s}_i)$. We can further expand the loss in Eqn. (8) into

$$\log \frac{U_i^+(\mathbf{s}_i', \mathbf{s}_{\mathcal{N}(v_i)})}{U_i^+(\mathbf{s}_i', \mathbf{s}_{\mathcal{N}(v_i)}) + U_i^-(\mathbf{s}_i', \mathbf{s}_{\overline{\mathcal{N}}(v_i)})} \geq \log \frac{U_i^+(\mathbf{s}_i, \mathbf{s}_{\mathcal{N}(v_i)})}{U_i^+(\mathbf{s}_i, \mathbf{s}_{\mathcal{N}(v_i)}) + U_i^-(\mathbf{s}_i, \mathbf{s}_{\overline{\mathcal{N}}(v_i)})}. \tag{12}$$

After a series of derivation, we can have the inequation $\frac{U_i^+(\mathbf{s}_i', \mathbf{s}_{\mathcal{N}(v_i)})}{U_i^-(\mathbf{s}_i', \mathbf{s}_{\overline{\mathcal{N}}(v_i)})} \geq \frac{U_i^+(\mathbf{s}_i, \mathbf{s}_{\mathcal{N}(v_i)})}{U_i^-(\mathbf{s}_i, \mathbf{s}_{\overline{\mathcal{N}}(v_i)})}$. According to the definition of the total utility in Eqn. (7), we further have $U_i(\mathbf{s}_i', \mathbf{s}_{-i}) \geq U_i(\mathbf{s}_i, \mathbf{s}_{-i})$. Thus, $\mathbf{s}_i'$ is a better-response strategy, i.e., $\mathbf{s}_i' \in \mathcal{B}_i'(\mathbf{s})$, based on Eqn. (3). Therefore, the algorithm is convergent and could reach a Nash Equilibrium point according to the literature (Young and Zamir 2014). For a detailed theoretical analysis of the existence of Nash Equilibrium in graph game, please refer to the section named Theoretical Analysis.

Extend to Semi-Supervision

Considering an extensive application to semi-supervised learning in graph-structural data, we further adapt the proposed method to the corresponding semi-supervised versions. In semi-supervised task, a part of node labels are accessible, and provide the optimal/target strategies for supervised learning. To further take advantage of the supervision information, we define the semi-supervised objective constraint on the basis of the unsupervised versions. In this case, for each labelled node, the semi-supervised utility comes from not only the competition incomes versus neighbours, but also the similarity between the predicted strategy and the target one. Accordingly, we can add a supervised cross-entropy loss term into the above loss function in Eqn. (8) and Eqn. (10) as semi-supervise graph game embedding.

Theoretical Analysis

In this section, we first give the definition of Nash Equilibrium in graph game. In order to prove that there must be a Nash Equilibrium state in the graph game, we introduce the Brouwer's fixed point theorem.

Definition 3 (Nash Equilibrium in Graph Game) *In the graph game $\Gamma_\mathcal{G} = (\mathcal{G}, \{\mathcal{S}_i | v_i \in \mathcal{V}\}, \{U_i | v_i \in \mathcal{V}\})$, if there exists a strategy set with $\mathbf{s}^* = (\mathbf{s}_1^*, \mathbf{s}_2^*, \cdots, \mathbf{s}_N^*)$, where $\mathbf{s}_i^* \in \mathcal{S}_i, i = 1, \cdots, N$. When other nodes fix the selected strategy, any node can not change its own strategy to obtain higher benefits. Which can be formalized as:*

$$U_i(\mathbf{s}_i^*, \mathbf{s}_{-i}^*) \geq U_i(\mathbf{s}_i', \mathbf{s}_{-i}^*), \quad \forall \mathbf{s}_i' \in \mathcal{S}_i, \ \forall v_i \in \mathcal{V}. \tag{13}$$

Theorem 1 (Brouwer's fixed point theorem) *Suppose $\mathcal{S} \subseteq \mathbb{R}^d$ is a nonempty, compact and convex set, $f : \mathcal{S} \to \mathcal{S}$ is a continuous mapping. Then in the set $\mathcal{S}$, there is at least one fixed point of f. Which means there is at least one point $\mathbf{x}^* \in \mathcal{S}$, satisfies $\mathbf{x}^* = f(\mathbf{x}^*)$. That's the Brouwer's fixed point theorem (Nikaido 1989).*

Theorem 2 *In graph game, mixed strategy equilibrium is allowed, and there must be Nash equilibrium in every limited strategic game.*

Proof *In graph game $\Gamma_\mathcal{G} = (\mathcal{G}, \{\mathcal{S}_i | v_i \in \mathcal{V}\}, \{U_i | v_i \in \mathcal{V}\})$, $\forall s \in \mathcal{S}, \mathbf{s} = \prod_{i=1}^n \mathbf{s}_i$, it is the map of $\mathcal{S} \to \mathcal{S}$. For any $\sigma \in \mathcal{S}$ and node v_i, set $U_i(\sigma_{-i}) = \arg\max_{\sigma_i \in \mathcal{S}_i} u_i(\sigma_i, \sigma_{-i})$ is the optimal response hybrid strategy of node v_i in $\mathcal{S}_i$ to the independent hybrid strategy combination σ_{-i} of other nodes.*

For any $\sigma_i \in U_i(\sigma_{-i}), \sigma_i^{'} \in U_i(\sigma_{-i}), \lambda \in [0, 1]$. Let

$$\sigma_i^{''} = \lambda \sigma_i + (1 - \lambda)\sigma_i$$

Obviously, $\sigma_i^{''} \in \mathcal{S}_i$.

$$\begin{aligned} u_i(\sigma_i^{''}, \sigma_{-i}) &= \sum_k \sigma_{ik}^{''} u_i(\mathbf{s}_{ik}, \sigma_{-i}) \\ &= \lambda \sum_k \sigma_{ik} u_i(\mathbf{s}_{ik}, \sigma_{-i}) + (1 - \lambda) \sum_k \sigma_{ik}^{'} u_i(\mathbf{s}_{ik}, \sigma_{-i}) \\ &\geq \lambda u_i(\tilde{\sigma_i}, \sigma_{-i}) + (1 - \lambda) u_i(\tilde{\sigma_i}, \sigma_{-i}) \\ &= u_i(\tilde{\sigma_i}, \sigma_{-i}), \qquad \forall \tilde{\sigma_i} \in U_i(\sigma_{-i}) \end{aligned} \tag{14}$$

*Therefore, $\sigma_i^{''} \in U_i(\sigma_{-i})$, and $U_i(\sigma_{-i})$ is **convex set**.*

*According to $u_i(\sigma_i, \sigma_{-i}) = \sum_k \sigma_{ik} u_i(\mathbf{s}_{ik}, \sigma_{-i})$, and $\mathcal{S}_i$ is a finite set. So there is a k that makes $u_i(\mathbf{s}_{ik}, \sigma_{-i}) = \max_l [u_i(\mathbf{s}_{il}, \sigma_{-i})]$ stand. That means $U_i(\sigma_{-i})$ is a **nonempty set**. Now we construct a function R, it maps the points from $\mathcal{S}$ to the subset of $\mathcal{S}$:*

$$R(\sigma) = \prod_{i=1}^{n} U_i(\sigma_{-i}), \sigma \in \mathcal{S}$$

For any $i = 1, \cdots, n$, $U_i(\sigma_{-i})$ always a nonempty convex set, so $R(\sigma)$ is also a nonempty convex set. Let's prove that R is upper half continuous.

Suppose $\{\sigma^k\}_{k=1}^{\infty}$ and $\{\tau^k\}_{k=1}^{\infty}$ are converge sequence.

$$\sigma^k \in \mathcal{S}, \tau^k \in R(\sigma^k), \quad k = 1, 2, \cdots$$

and $\sigma = \lim_{k\to\infty} \sigma^k, \tau = \lim_{k\to\infty} \tau^k$. In order to prove R is upper half continuous, we need to prove that $\tau \in R(\sigma)$. Because of:

$$u_i(\tau_i^k, \sigma_{-i}^k) \geq u_i(\sigma_i, \sigma_{-i}^k), \quad \forall \sigma_i \in \mathcal{S}_i, k = 1, 2, \cdots$$

Obviously, the expected utility function U_i is a continuous function on $\mathcal{S}$, and:

$$u_i(\tau_i, \sigma_{-i}) \geq u_i(\sigma_i, \sigma_{-i}), \qquad \forall \sigma_i \in \mathcal{S}_i$$

*Therefore, for every i, $\tau_i \in U_i(\sigma_{-i})$, which means $\tau \in R(\sigma)$. And R is the **upper half continue map** of $\mathcal{S}$ to itself.*

According to the Brouwer's fixed point theorem, there exists a hybrid strategy combination σ in strategy set $\mathcal{S}$ satisfies $\sigma \in R(\sigma)$. Which means for every $i \in [1, \cdots, n]$, $\sigma_i \in U_i(\sigma_{-i})$, and σ is the (hybrid) Nash Equilibrium of $\Gamma_{\mathcal{G}}$.

Experiments

In this section, we conduct comprehensive experiments to evaluate the proposed models. We first introduce the used datasets and experimental setup, then show the experimental results and compare them with state-of-the-art methods, and finally make some discussion about the convergence.

Dataset and Experimental Setup

Three citation networks, named Cora, Citeseer and Pubmed, are employed to evaluate our proposed method. Cora dataset consists of 2708 scientific publications of seven classes with 5429 existing links, while Citeseer dataset consists of 3312 scientific publications classified into one of six classes with totally 4732 links. As the largest dataset among the three, Pubmed citation network consists of 19717 scientific publications of three classes with 44338 links. To describe each publication, for either the Cora or Citeseer dataset, a dictionary is constructed to encode a publication into a 0/1-valued word vector indicating the absence/presence of the corresponding word, where the unique words in the dictionaries of Cora and Citeseer are 1433 and 3703, respectively. For Pubmed dataset, each publication is described with a TF/IDF weighted word vector by a dictionary containing 500 unique words.

Models	Cora	Citeseer	Pubmed
Raw_feature	47.9	49.3	69.1
LP	68.0	45.3	63.0
DeepWalk	67.2	43.2	65.3
AttentionWalk	67.9	51.5	-
GAE	77.3	58.2	74.6
DGE	73.9	62.8	75.6
DGI	76.9	62.9	73.6
CAN	79.2	63.2	75.8
U-PGE(Ours)	**80.1**	**63.8**	75.4
U-GGE(Ours)	79.6	62.7	**76.1**

Table 1: Results of unsupervised node classification on the Cora, Citeseer and Pubmed datasets.

Models	Cora	Citeseer	Pubmed
Planetoid	75.7	64.7	77.2
Chebyshev	81.2	69.8	74.4
GCN	81.5	70.3	79.0
GraphSAGE	74.7	63.0	78.9
GWNN	81.6	71.7	79.1
GraphStar	82.1	71.0	77.2
APPNP	82.2	70.0	79.4
GAT	83.0	72.5	79.0
DGCN	83.5	72.6	79.6
LGGCN	83.3	**73.0**	79.5
S-PGE(Ours)	83.6	71.9	79.3
S-GGE(Ours)	**84.0**	72.7	**80.0**

Table 2: Results of Semi-supervised node classification on the Cora, Citeseer and Pubmed datasets.

We perform two tasks on the three citation datasets including node classification and link prediction. For node classification task, we adopt the widely used protocol in (Yang, Cohen, and Salakhudinov 2016) for fair comparison. For all three datasets, there are 20 samples in each class for training, 500 samples for validation, and 1000 samples for testing. The accuracy metric is used for measurement, and the performance is calculated by averaging the results of 20 runs. We test both group and paired game embeddings in semi-supervised and unsupervised classification tasks. In this process, the architectures of the models are kept the same while slight difference exists in parameter settings and implementation details. For both tasks, in Eqn. (11) of parametrized learning, we stack the function $f(\cdot)$ twice in the model and set K to be 8. Specifically, the projection matrix $\mathbf{W}^k$ yields a 64-dimensional output vector in unsupervised learning while a 128-dimensional vector in semi-supervised learning. For node sampling, the number of negative samples is set twice of the positive, where the positive number is set as 5 for all experiments. Specifically, positive samples come from the neighbours while negative samples are chosen from the nodes that cannot reach the central node in two steps. For unsupervised classification in the testing stage, the cosine similarity

between strategies is calculated.

For the link prediction task, the parameter setting is set almost the same with the node classification task, and just different in the dataset split protocol. All experiments strictly follow the dataset split protocol of GAE (Kipf and Welling 2016). The results of the baseline methods, except CAN (Meng et al. 2019), all come from the public reported results in previous literature as they follow the same protocol. Specifically, for the CAN method, we run its source code on the unified protocol and report the results.

Experimental Results

Unsupervised Graph Embedding. In the unsupervised task, we compare our unsupervised group game embedding (U-GGE) and unsupervised pair game embedding (U-PGE) models with multiple unsupervised graph learning methods, including deep graph infomax (DGI), Deepwalk, attention work, etc. The experimental results are shown in Table 1. Different from the method of training a classifier by DGI to evaluate the embedding quality, we adopt a completely nonparametric method to evaluate the embedding quality. Specifically, we use the embedded nearest neighbor relationship between the testing set and the training set to classify the test samples (the label of the tested sample is the training sample label of the nearest neighbor). In order for the fairness of comparing, we use the same test method to test the embedding obtained by other unsupervised methods. In general, the proposed U-PGE model achieves the better result than the U-GGE model. Compared with other existing methods, our U-PGE model achieves the best performance on all three citation networks, and outperforms the recent state-of-the-art method, i.e. DGI, with the performance gain of 3.2%, 0.9% and 1.8% on Cora, Citeseer and Pubmed respectively.

Semi-supervised Graph Embedding. In the semi-supervised task, the results of our models named semi-supervised GGE (S-GGE) and semi-supervised PGE (S-PGE) are shown in Table 2, and also compared with other state-of-the-art methods including Planetoid(Yang, Cohen, and Salakhudinov 2016), Chebyshev(Defferrard, Bresson, and Vandergheynst 2016), GCN(Kipf and Welling 2017), graph attention networks (GAT)(Veličković et al. 2018), LGGCN(Gao, Wang, and Ji 2018), DGCN(Zhuang and Ma 2018), etc. Specifically, comparing the S-GGE model with S-PGE, we can see that S-GGE outperforms S-PGE on all three datasets, which is different from the result in the unsupervised task after additionally involving label in guiding the game evolution. Comparing with existing methods, our S-GGE model achieves the best performance on Cora and Pubmed datasets, and also obtains the comparable accuracy with the GAT on Citeseer dataset.

Link Prediction. In the Link prediction task, We compare our pair game embedding (PGE) models with multiple baseline methods, and the comparison results are summarized in Table 3. Our PGE achieves the best performance, and outperforms the recent state-of-the-art method, i.e. CAN, with the performs gain of 3.8%, 5.4% and 1.5% on Cora, Citeseer and Pubmed in AUC criteria respectively.

Models	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
GCNII	76.5	76.4	73.8	73.8	79.9	80.6
GraphSAGE	79.5	76.3	80.2	79.1	84.0	82.9
GAE	91.0	92.0	89.5	89.9	96.4	96.5
VGAE	91.4	92.6	90.8	92.0	94.4	94.7
CAN	93.4	94.2	93.2	93.9	96.2	96.1
DGE	90.8	-	95.5	-	84.3	-
PGE (Ours)	**97.2**	**96.7**	**98.6**	**98.9**	**97.7**	**97.4**

Table 3: Results of Link prediction on the Cora, Citeseer and Pubmed datasets.

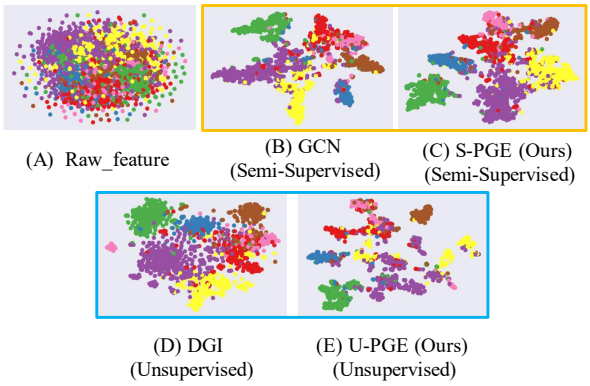

Figure 1: t-SNE embeddings of the nodes in Cora dataset from the Raw_feature (A) and features from two different supervision modes: semi-supervised (GCN (B) and S-PGE (C)) and unsupervised (DGI (D) and U-PGE (E)).

Discussion on Convergence

Besides performance evaluation, it's also interesting for us to discuss the model convergence, which is dominated by Nash Equilibrium in our framework. So, we conduct several additional experiments:

- Visualizing the features under convergence. The learned features of two pairs of methods, i.e. our S-PGE versus semi-supervised GCN and our U-PGE versus unsupervised DGI, are visualized after projected into 2D plane. Please see Fig. 1.
- Quantitative evaluation of convergence. The silhouette scores (Rousseeuw 1987) corresponding to the four methods in Fig. 1 are computed and compared in Table 4.
- Visualization of utility/MI (Mutual Information) curves in training process. The curves of our U-PGE, U-GGE and unsupervised DGI. Please see Fig. 2.

Based on the results above, we have the following observations:

- Our graph game embedding framework dominated by Nash Equilibrium possesses better convergence states. Fig. 1 intuitively shows both better clustering quality and

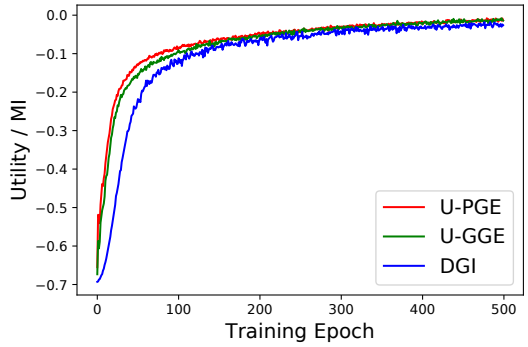

Figure 2: Convergence curves of U-PGE, U-GGE and DGI on Cora dataset.

Supervision mode	Models	Before t-SNE	After t-SNE
	Raw_feature	−0.002	0.375
Semi-supervised	GCN	0.346	0.490
	S-PGE (ours)	**0.417**	**0.506**
Unsupervised	DGI	0.124	0.430
	U-PGE (ours)	**0.234**	**0.463**

Table 4: Silhouette scores on Cora dataset. Before t-SNE means obtained from original features, and after t-SNE means calculated after projected into 2D space.

linear separability of our proposed models. And this is also quantitatively verified by the statistical silhouette scores in Table 4.

- Nash Equilibrium also promotes the convergence speed of our framework. According to Fig. 2, comparing to the recent state-of-the-art DGI, both the proposed U-PGE and U-GGE models converge to relatively stable points with higher speed, especially in the starting stage.

Conclusion

In this paper, a novel graph game embedding framework was proposed to learn discriminative node embeddings for given graphs. By leveraging game theory for graph inference, we converted the task of node embedding to strategy selection/searching processes of players. Then, a utility function satisfying Nash Equilibrium condition was designed to measure the benefit/loss of players. Besides, a collaboration and competition mechanism was further introduced to facilitate the discriminative feature leaning. Under this proposed framework, considering different interaction manners of nodes, we proposed two specific embedding models, named PGE for paired nodes and GGE for group interaction. In this process, we provided the theoretical proof of the Nash Equilibrium satisfaction, and analysed the complexity of the proposed models. The experimental results verified the effectiveness of our models in both unsupervised and semi-supervised tasks, and also demonstrated better convergence of our framework.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61906094, 62072244, 61972204), the Natural Science Foundation of Jiangsu Province (Grant Nos. BK20190019 and BK20190452), and the CCF-Tencent Open Research Fund.

References

Ahmed, A.; Shervashidze, N.; Narayanamurthy, S.; Josifovski, V.; and Smola, A. J. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, 37–48.

Akkarajitsakul, K.; Hossain, E.; Niyato, D.; and Kim, D. I. 2011. Game theoretic approaches for multiple access in wireless networks: A survey. *IEEE Communications Surveys & Tutorials* 13(3): 372–395.

Arora, S.; Khandeparkar, H.; Khodak, M.; Plevrakis, O.; and Saunshi, N. 2019. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229* .

Belkin, M.; and Niyogi, P. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, 585–591.

Borgwardt, K. M.; Kriegel, H.-P.; Vishwanathan, S.; and Schraudolph, N. N. 2007. Graph kernels for disease outcome prediction from protein-protein interaction networks. In *Biocomputing 2007*, 4–15. World Scientific.

Cai, H.; Zheng, V. W.; and Chang, K. C.-C. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30(9): 1616–1637.

Cao, S.; Lu, W.; and Xu, Q. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, 891–900.

Cao, S.; Lu, W.; and Xu, Q. 2016. Deep neural networks for learning graph representations. In *Thirtieth AAAI conference on artificial intelligence*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709* .

Cheng, Y.; Li, Y.; Liu, Q.; Yao, Y.; Sai Vijay Kumar Pedapudi, V.; Fan, X.; Su, C.; and Shen, S. 2019. A graph based unsupervised feature aggregation for face recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 0–0.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, 3844–3852.

Eguiluz, V. M.; and Tessone, C. J. 2009. Critical Behavior in an Evolutionary Ultimatum Game with Social Structure. *Advances in Complex Systems* 12(02): 221–232.

Ekman, J. 2001. Game Theory Evolving. A Problem-Centered Introduction to Modelling Strategic Interaction,: Herbert Gintis, Princeton University Press, 2000. ISBN 0-691-00942-2, 0-691-00943-0. pp. 532. *Ecological Economics* 39(3): 479–480.

Fout, A.; Byrd, J.; Shariat, B.; and Ben-Hur, A. 2017. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*, 6530–6539.

Gao, H.; Wang, Z.; and Ji, S. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1416–1424.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, 1735–1742. IEEE.

Hauert, C.; Holmes, M.; and Doebeli, M. 2006. Evolutionary games and population dynamics: maintenance of cooperation in public goods games. *Proceedings of The Royal Society B: Biological Sciences* 273(1600): 2565–2570.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9729–9738.

Jiang, R.; Ho, A. T.; Cheheb, I.; Al-Maadeed, N.; Al-Maadeed, S.; and Bouridane, A. 2017. Emotion recognition from scrambled facial images via many graph embedding. *Pattern Recognition* 67: 245–251.

Kipf, T.; van der Pol, E.; and Welling, M. 2019. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247* .

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* .

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Magurran, A. E.; and Dugatkin, L. A. 1998. Finding Ways to Live Together@@@Cooperation Among Animals: An Evolutionary Perspective. *Evolution* 52(3): 925.

Manshaei, M. H.; Zhu, Q.; Alpcan, T.; Bacşar, T.; and Hubaux, J.-P. 2013. Game theory meets network security and privacy. *ACM Computing Surveys (CSUR)* 45(3): 1–39.

Meng, Z.; Liang, S.; Bao, H.; and Zhang, X. 2019. Co-embedding attributed networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 393–401.

Moura, J.; and Hutchison, D. 2017. Survey of game theory and future trends with application to emerging wireless data communication networks. *Researchgate. net/publication* 315764798.

Myerson, R. B. 2013. *Game theory*. Harvard university press.

Nash, J. J. F. 1950. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America* 36(1): 48–49.

Nikaido, H. 1989. Fixed point theorems. In *General Equilibrium*, 139–144. Springer.

Nowak, M. A.; and May, R. M. 1992. Evolutionary games and spatial chaos. *Nature* 359(6398): 826–829.

Orsini, F.; Baracchi, D.; and Frasconi, P. 2018. Shift aggregate extract networks. *Frontiers in Robotics and AI* 5: 42.

Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 1105–1114.

Oulaourf, S.; Haidine, A.; Aqqal, A.; and Ouahmane, H. 2017. Review on radio resource allocation optimization in LTE/LTE-advanced using game theory. *International Journal of Communication Networks and Information Security (IJCNIS)* 9(1): 117–153.

Perc, M.; and Szolnoki, A. 2010. Coevolutionary Games - A Mini Review. *BioSystems* 99(2): 109–125.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Philippe, M. 1994. Playing fair game theory and the social contract. *Journal Des Economistes Et Des Etudes Humaines* 5: 429–436.

Roughgarden, T. 2010. Algorithmic game theory. *Communications of the ACM* 53(7): 78–86.

Rousseeuw, P. J. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20: 53–65.

Santos, F. C.; Rodrigues, J. F. M.; and Pacheco, J. M. 2005. Epidemic spreading and cooperation dynamics on homogeneous small-world networks. *Physical Review E* 72(5): 056128.

Silva, N. B.; Tsang, R.; Cavalcanti, G. D.; and Tsang, J. 2010. A graph-based friend recommendation system using genetic algorithm. In *IEEE congress on evolutionary computation*, 1–7. IEEE.

Szabo, G.; and Fath, G. 2007. Evolutionary games on graphs. *Physics Reports* 446(4): 97–216.

Van Rooij, R. 2007. The Stag Hunt and the Evolution of Social Structure. *Studia Logica* 85(1): 133–138.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *International Conference on Learning Representations* .

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=rklz9iAcKQ.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 1225–1234.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 40–48.

Young, P.; and Zamir, S. 2014. *Handbook of game theory*. Elsevier.

Yu, W.; Zheng, C.; Cheng, W.; Aggarwal, C. C.; Song, D.; Zong, B.; Chen, H.; and Wang, W. 2018. Learning deep network representations with adversarially regularized autoencoders. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2663–2671.

Zhuang, C.; and Ma, Q. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*, 499–508.