Network Embedding Attack: An Euclidean Distance Based Method

Shanqing Yu[1,2], Jun Zheng[2], Yongqi Wang[2], Jinyin Chen[1,2], Qi Xuan[1,2,3(✉)], and Qingpeng Zhang[4]

[1] Institue of Cyberspace Security, Zhejiang University of Technology, Hangzhou 310023, China
[2] College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China
{yushanqing,2111903062,chenjinyin,xuanqi}@zjut.edu.cn, zjun2878@gmail.com
[3] PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen 518000, China
[4] City University of Hong Kong, Hong Kong 999077, China
qingpeng.zhang@cityu.edu.hk

Abstract. Network embedding methods are widely used in graph data mining. This chapter proposes a Genetic Algorithm (GA) based Euclidean Distance Attack strategy (EDA) to attack the DeepWalk-based network embedding to prevent certain structural information from being discovered. EDA disrupts the Euclidean distance between pairs of nodes in the embedding space by making a minimal modification of the network structure, thereby rendering downstream network algorithms ineffective, because a large number of network embedding based downstream algorithms, such as community detection and node classification, evaluate the similarity based on the Euclidean distance between nodes. Different from traditional attack strategies, EDA is an unsupervised network embedding attack method, which does not need labeling information.

Experiments with a set of real networks demonstrate that the proposed EDA method can significantly reduce the performance of DeepWalk-based networking algorithms, outperforming other attack strategies in most cases. The results also indicate the transferability of the EDA method since it works well on attacking the network algorithms based on other network embedding methods such as High-Order Proximity preserved Embedding (HOPE) and non-embedding-based network algorithms such as Label Propagation Algorithm (LPA) and Eigenvectors of Matrices (EM).

Keywords: MDATA · Network embedding · Euclidean distance attack

1 Introduction

Real-world complex systems can be represented and analyzed as networks. Network embedding map the nodes of a network into vectors in an Euclidean space, which largely facilitates the application of machine learning methods in graph data mining [1,2]. In this chapter, we focus on attacking the network embedding process, rather than one particular network algorithm. Since most network

Y. Jia et al. (Eds.): MDATA: A New Knowledge Representation Model, LNCS 12647, pp. 131–151, 2021.
https://doi.org/10.1007/978-3-030-71590-8_8

algorithms are based on network embedding, attacking the embedding process (instead of particular algorithms) could be a more generic approach that can be easily applied to various attack tasks. In this consideration, here we propose an Euclidean Distance Attack (EDA), aiming to disturb the distances between vectors in the embedding space directly. We think it is the distance between vectors that determine the performance of many downstream algorithms based on network embedding. Since we do not need to know the labels of training data, it can be considered as an unsupervised learning method. In particular, the main contributions of this chapter are as below:

- We propose a novel unsupervised attack strategy on network embedding, namely EDA, using the Euclidean Distance between embedding vectors as the reference.
- We adopt the Genetic Algorithm (GA) to search for the optimal solution for EDA. As compared with state-of-the-art baseline attack strategies, EDA performed the best in reducing the prediction accuracy of downstream algorithms for community detection and node classification tasks.
- We validate the transferability of EDA in reducing the performance on many other network algorithms.

The rest of the chapter is organized as follows. The next section introduces the background. In Sect. 3, we introduce the problem and we present some related methods in Sect. 4. Our method is presented in Sect. 5 including the experimental results. We discuss about the connection with the MDATA model in Sect. 6 and we finally summarize the chapter in Sect. 7.

2 Background

During the past a few decades, network science has emerged as an essential interdisciplinary field aiming at using network and graph as a tool to characterize the structure and dynamics of complex systems, including social networks, citation networks, protein networks and transport networks [3]. Network embedding solves the problem of high dimensions and sparseness of the original network data. It builds a bridge between machine learning and network science, enabling many machine learning algorithms to be applied in network analysis.

The network algorithm has developed rapidly and has been recognized by many people, but its robustness has not been widely verified, which is indispensable before generally applied. Moreover, while providing convenience to researchers, such network analysis algorithms may also bring the risk of privacy leakage, i.e., our personal information in the social network may be easily inferred by adopting such algorithms [4–6].

Recent studies on deep learning have shown that deep neural networks, especially those in the area of computer vision, seem to be susceptible to small perturbations despite the remarkable performances in many tasks [7–10]. These adversarial attacks usually target at specific algorithms and make the prediction accuracy drop sharply. Quite recently, it was also found that network algorithms

in community detection [11], link prediction [12], and node classification [13–15] are also vulnerable to such adversarial attacks. Most of previous works focused on a specific task or particular algorithm which limit the practical significance of those research.

3 Problem

Network embedding attack, which aimed at making the node embedding representation as different as possible by slight perturbation on the original network, could be described mathematically as follows.

Given a undirected and unweighted network $G(V, E)$, where V denotes the set of nodes, and E denotes the set of links. The link between nodes v_i and v_j is denoted by $e_{ij} = (v_i, v_j) \in E$. In the process of perturbation, we denote the set of added links as $E^+ \subseteq \overline{E}$ and the set of deleted links as $E^- \subseteq E$, where $\overline{E}$ is the set of all pairs of unconnected nodes in G. Then, we get the adversarial network $\hat{G}(V, \hat{E})$ with the updated the set of links $\hat{E}$ satisfying the following formula.

$$\max \quad D(Emb(G), Emb(\hat{G})) \tag{1}$$

$$\text{s.t.} \quad \hat{E} = E \cup E^+ - E^- \tag{2}$$

$$|E^+| = |E^-| \tag{3}$$

$$|E^+| + |E^-| \ll |E| \tag{4}$$

where $D(Emb(G), Emb(\hat{G}))$ denotes the difference between the embedding representations of G and $\hat{G}$.

4 Related Methods

4.1 Network Embedding

With the development of machine learning technology, the feature learning for nodes in the network has become an emerging research task. The embedding algorithm transforms network information into the low-dimensional dense vector, which acts as the input for machine learning algorithms.

In the recent study, the first network embedding method was proposed in [16], namely DeepWalk, which introduced Natural Language Processing (NLP) model [17–19] into network and achieved great success in community detection [20–23] and node classification [24,25]. In [26], it developed Node2vec as an extension of DeepWalk. They utilized a biased random walk to combine BFS and DFS neighborhood exploration so as to reflect equivalence and homogeneity in the network structure. LINE was proposed in [27] that preserved both the first-order and second-order proximities to rich node representation. Moreover, it uses the adversarial framework for graph representation learning in [28].

For some specific tasks, such as community detection, a new Modularized Nonnegative Matrix Factorization (M-NMF) model os proposed in [29], which incorporates community structures into network embedding. Their approach is

to jointly optimize the NMF-based model and the module-based community model so that the learned representation of nodes can maintain both microscopic and community structures. It formulated a multi-task neural network structure(PRUNE) in [30] to combine embedding vectors and the global node ranking. This model can more abundantly retain the information from the network structure.

These network embedding methods show the following advantages. Firstly, embedding can well obtain the structural information of the network; secondly, many approaches can be easily parallel to decrease the time consumption; thirdly, the representation vectors of nodes, obtained by the network embedding algorithms, can be used to support the subsequent network analysis tasks, such as link prediction, community detection, and node classification.

4.2 Adversarial Attacks

Given the importance of graph algorithm analysis, an increasing number of works start to analyze the robustness of machine learning models on graph [31] and graph neural networks model [32]. For instance, due to the connectivity and cascading effects in networks, the robustness and vulnerability of complex networks are analyzed in [33]. In [34], it developed a heuristic method, namely Disconnect Internally, Connect Externally (DICE). They added perturbations to the original network by deleting the links within community while adding the links between communities. An attack strategy against community detection, namely Q-Attack, is proposed in [11], which uses modularity Q, under particular community detection algorithm, as the optimization objective, aiming to disturb the overall community structure. In [12], it introduced an evolutionary method against link prediction using Resource Allocation Index (RA) to protect link privacy.

As for node classification, NETTACK is proposed in [13] to fool Graph Convolutional Networks (GCN) by generating adversarial networks. It further proposed Fast Gradient Attack (FGA) [14], which utilized the gradient of algorithm to design loss function. FGA model can generate adversarial network faster and make the target nodes classified incorrectly with quite small perturbations. Moreover, a greedy algorithm to attack GCNs is designed in [15] by adding fake nodes. This method generated adjacency and feature matrices of fake nodes to minimize the classification accuracy of the existing nodes. Those kinds of attack strategies are always effective in some cases since they are strongly targeted at specific algorithms.

Most current attack strategies belong to supervised learning, i.e., attackers can get actual labels of nodes or communities, and further utilize this information to design attack strategies. However, in many cases, it is difficult to get such information, making those supervised attack strategies less effective. To solve these problems, it analyzes adversarial vulnerability based on the structure of network only and derive efficient adversarial perturbations that poison the network [35]. What's more, it proposes an method based on projected gradient descent to attack unsupervised node embedding algorithms in [36].

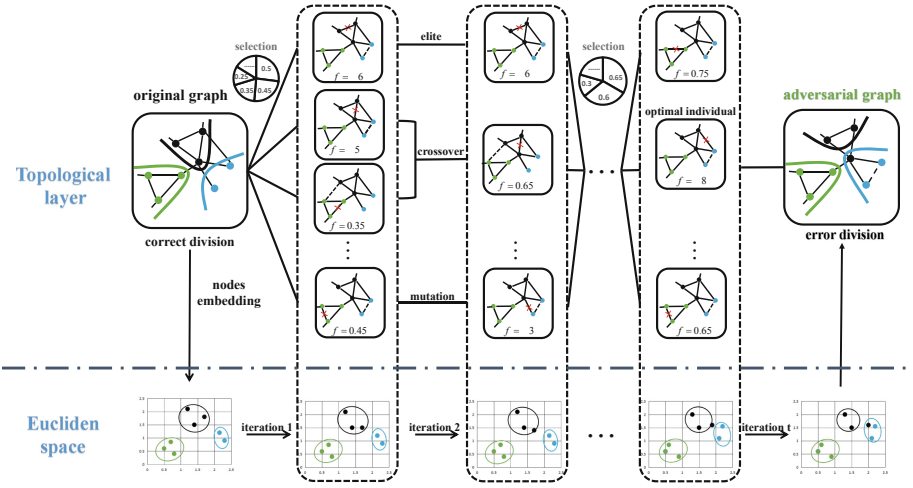


Fig. 1. The framework of EDA on a network. With the evolution of perturbation, the positions of nodes in the Euclidean space constantly changes, leading to the changing prediction results of node classification.

5 Our Method

5.1 Euclidean Distance Attack

In this section, we introduce *DeepWalk* briefly, based on which we propose the Euclidean distance attack (EDA) method. In particular, we turn the attack problem to a combinatorial optimization problem and then use Genetic Algorithm (GA) to solve it. Here, we choose DeepWalk because it is one of the most widely-used unsupervised network embedding methods and it can have good mathematical properties rooted at matrix factorizations [37].

DeepWalk. This chapter mainly focuses on undirected and unweighted networks. A network is represented by $G(V, E)$, where V denotes the set of nodes and, E denotes the set of links. The link between nodes v_i and v_j is denoted by $e_{ij} = (v_i, v_j) \in E$. The adjacency matrix of the network then is defined as A, with the element denoted by

$$a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E. \end{cases} \tag{5}$$

Real-world networks are often sparse and high-dimensional, preventing the broad application of machine learning algorithms in graph data. To address these problems, network embedding is a family of methods to encode the topological properties in the graph data into low-dimensional features.

DeepWalk trains the vectors of nodes $R^{|V|\times n}$ by calculating the probability of generating the nodes on both sides from the center node, with the loss function represented by

$$\min_{R} \sum_{k=-w,k\neq 0}^{w} -\log P(v_{i+k} \mid v_i), \tag{6}$$

where the sequence $\{v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}\}$ is obtained by random walk within the window w around the center node v_i, and the probability $P(v_{i+k}|v_i)$ can be transformed by the following softmax function [38,39]:

$$P(v_{i+k} \mid v_i) = \frac{\exp(r_i r_{i+k}^T)}{\sum_{n=1}^{|V|} \exp(r_i r_n^T)}, \tag{7}$$

where r_i is the representation vector of node v_i.

EDA on Network Embedding. Many machine learning methods are based on the relative, rather than absolute, positions of samples in Euclidean space. Thus the Euclidean distance between samples playing a vital role in these methods. Moreover, due to the randomness of many network embedding algorithms, embedding vectors generated for the same node in different rounds might be different from each other. Regardless of such differences, the Euclidean distance between the same pair of nodes in the embedding space is approximately consistent. Those are the key motivation that drives us to propose the Euclidean Distance Attack (EDA): *Disturbing the Euclidean distance between pairwise nodes in the embedding space as much as possible with the minimal changes of the network structure.*

In particular, we calculate the distance between a pair of nodes v_i and v_j in the embedding space as follows:

$$d_{ij} = \mathbf{dist}(r_i, r_j) = ||r_i - r_j||_2, \tag{8}$$

based on which we can get the Euclidean distance matrix $D = [d_{ij}]_{|V|\times|V|}$ for the whole network, with each row denoted by D_i representing the Euclidean distances between node v_i and all the other nodes in the network.

Denote the adversarial network after our EDA as $\hat{G}$, and its corresponding Euclidean distance matrix in the embedding space as $\hat{D}$. We calculate the Pearson correlation coefficient between the distance vectors of the corresponding nodes in the original network and adversarial network:

$$\varphi(G, \hat{G}) = \sum_{i=1}^{|v|} |\rho(D_i, \hat{D}_i)| \tag{9}$$

Then, we focus on minimizing φ by changing a certain number of links in the network with the following objective function:

$$\min \varphi(G, \hat{G}). \tag{10}$$

Process of Perturbation. we perform network perturbation by adding and removing links. Furthermore, the perturbation is always excepted to be imperceptible when the attack methods are applied in real world scenarios. Thus EDA is designed to lower the performance of network embedding algorithms at a minimum cost, which means the perturbation should be as small as possible.

In the process of perturbation, we denote the set of added links as $E^+ \subseteq \overline{E}$ and the set of deleted links as $E^- \subseteq E$, where $\overline{E}$ is the set of all pairs of unconnected nodes in G. Then, we get the adversarial network $\hat{G}(V, \hat{E})$ with the updated the set of links $\hat{E}$ satisfying

$$\hat{E} = E \cup E^+ - E^- \tag{11}$$

Suppose u is the number of flipped links in the attack, the total complexity of instances in the searching space is equal to

$$O(u) = \mathcal{C}^u_{|E|} * \mathcal{C}^u_{|\overline{E}|}, \tag{12}$$

which could become huge as the size of the network or the number of flipping links grows. The search for the optimal solution is NP-hard, and thus we adopt the Genetic Algorithm (GA) to search for the optimal solution. The detailed procedure of EDA is presented in Algorithm 1.

Algorithm 1: Method of EDA

Input: *The original network* $G = (V, E)$
Output: *Adversarial network* $\hat{G}^*$

Initialize *the node vectors* $R^{|V|\times n}$ *and the distance matrix* D, with $d_{ij} = \mathbf{dist}(r_i, r_j)$;
while *not converged* **do**
  $\hat{R}^{|V|\times n} = \mathbf{DeepWalk}(\hat{A}, n)$;
  for $i = 1; i \leq |V|; i++$ **do**
    for $j = 1; j < i; j++$ **do**
      $\hat{d_{ij}} = \sqrt{\sum_{q=1}^{n} (\hat{r_i^q} - \hat{r_j^q})^2}$;
    end
  end
  for $i = 1; i \leq |V|; i++$ **do**
    $\rho_\tau += \rho(D_i, \hat{D}_i)$;
  end
  $fitness = 1 - \frac{\rho_\tau}{|V|}$;
  $\hat{G} = \mathbf{Genetic\ Algorithm}(G, fitness)$
end
return *Adversarial network* $\hat{G}^*$;

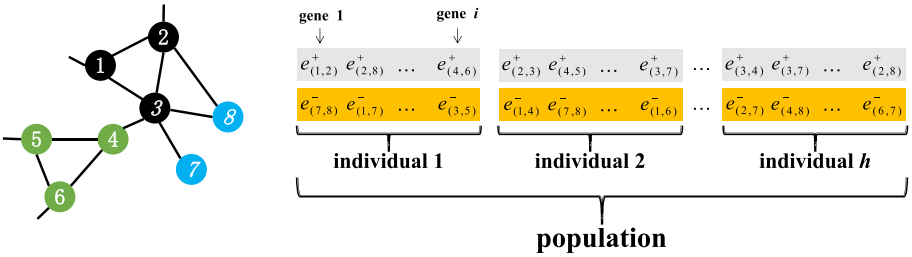


Fig. 2. Encoding of network. The individual solution includes i genes formed by adding and deleting links, and a population consists of h individuals. The sets of added and deleted links are represented in gray and yellow, respectively.

The Evolution of EDA. Here, we use the GA to find the optimal set of flipping links for EDA. Typically, the algorithm consists of three parts: encoding of the network, selection by fitness, crossover and mutation operation:

- **Encoding of network:** We directly choose the flipped links as genes, including the set of removed links E^- and the set of added links E^+. The length of each chromosome is equal to the number of added or deleted links. Figure 2 shows an overview of network encoding. Individuals are combinations of flipping links, representing different solutions of adversarial perturbations, and a population consists of h individuals.
- **Selection by fitness:** We use Eq. (13) as the fitness function of individual k in GA, capturing the relative changes of vector distances in the embedding space by the attack:

$$f(k) = 1 - \frac{\varphi(G, \hat{G})}{|V|}. \tag{13}$$

  Then, the probability that individual k is selected to be the parent genes in the the next generation is proportional to its fitness:

$$p(i) = \frac{f(i)}{\sum_{j=1}^{h} f(j)}. \tag{14}$$

- **Crossover and mutation:** We then use the selected individuals of higher fitness as the parents to generate new individuals by adopting crossover and mutation operations, assuming that those better genes can be inherited in the process. In particular, single-point crossover between two individuals is used, with probability p_c, as shown in Fig. 3; while for mutation, we randomly select individuals from a population and randomly change their genes, with probability p_m, as shown in Fig. 4.

Overview of EDA. In Fig. 1, we divide the whole framework into two parts, the topological layer and the Euclidean space layer. In the topological layer,

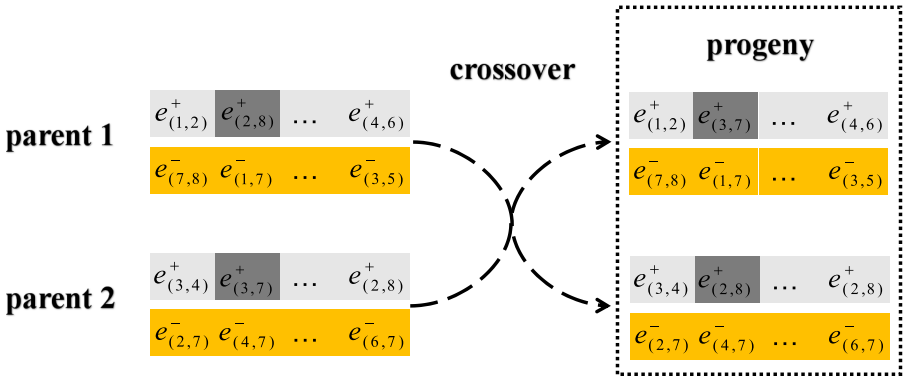


Fig. 3. An example of crossover operation, where link (2, 8) in parent 1 and link (3, 7) in parent 2 are exchanged to produce progeny.

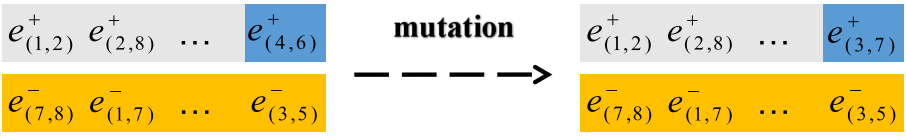


Fig. 4. An example of crossover operation, where a new link (3, 7) is generated by mutation.

we perturb the original network by adding or removing links, which can be observed directly in the figure. In the Euclidean space layer, the location of nodes are obtained by graph embedding algorithm. EDA method uses the Genetic Algorithm to iterate the initially generated perturbation, and finally blind the prediction algorithms.

For example, as shown in Fig. 1, one member (the node in black) in the original network that be divided into the black community from network embedding algorithm. However, after t times of evolutionary iterations, by flipping some links, this node is incorrectly divided into the blue community. Our strategy utilizes Genetic Algorithm to generate the optimal adversarial network through selection, crossover and mutation.

5.2 Experiments

To evaluate the effectiveness of EDA, we compare it with four baseline methods by performing multi-task experiments on several real-world networks.

Datasets. Here, we choose four commonly-used benchmark social networks to evaluate the performance of the attack strategies.

– **Zachary Karate Club (Karate)** [40]. The karate network is constructed by observing a university karate club, which consists of 34 nodes and 78 links.

Algorithm 2: Genetic Algorithm (GA)

Input: *The fitness of individuals* $f(k),\ k = 1, 2, \ldots, h$ *and the parameters* $h,\ p_c,\ p_m$;
Output: *Adversarial network* $\hat{G}$
1 Initialize *individuals in population*;
2 $Elites = \textbf{Retain}(f(k), population)$; $Selection = \textbf{Selection}(f(k), individual)$; $Crossover = \textbf{Crossover}(p_c, Section)$; $Mutation = \textbf{Mutation}(p_m, Section)$;
3 population $= Elites \cup Crossover \cup Mutation$;
4 *Reconstruct network.*
5 return *Evolutionary network* $\hat{G}$;

Table 1. Basic structural features of three networks. $|V|$ and $|E|$ are the numbers of nodes and links, respectively; $\langle k \rangle$ is the average degree; C is the clustering coefficient and $\langle d \rangle$ is the average distance.

	Karate	Game	Citeseer	Cora
$\|V\|$	34	107	3312	2708
$\|E\|$	78	352	4732	5429
$\langle k \rangle$	6.686	6.597	3.453	5.680
C	0.448	0.259	0.128	0.184
$\langle d \rangle$	2.106	2.904	7.420	5.271

- **Game of Thrones (Game)** [41]. The TV sensation *Game of Thrones* is based on George Martin's epic fantasy novel series *A Song of Ice and Fire*. It is a network of character interactions in the novel, which contains 353 links connecting 107 characters.
- **The Citeseer dataset (Citeseer)** [42]. It is a paper citation network consisting of 3,312 papers and 4,732 citation links. These papers are divided into six classes.
- **The Cora dataset (Cora)** [42]. This dataset contains seven classes by combining several machine learning papers, which contains 2,708 papers and 5,429 links in total. The links between papers represent the citation relationships.

The basic topological properties of these networks are presented in Table 1.

Baseline Methods. We perform experiments on community detection and node classification to see how the proposed EDA degrades their performances. In particular, we compare the performance of EDA with that of the following baseline methods.

- **Randomly Attack (RA).** RA randomly deletes the existent links, while randomly adds the same number of nonexistent links. This attack strategy does not require any prior information about the network.
- **Disconnect Internally, Connect Externally (DICE).** DICE is a heuristic attack algorithm for community detection [34]. The attacker needs to

know the correct node labels in the network and then delete the links within community and add the links between communities.

- **Degree-Based Attack (DBA).** It has been found that many real-world networks follow the power-law distribution [43], in which a small fraction of nodes (usually named as hubs) have most connections. Since it is generally recognized that these hub nodes often have a huge impact on the connectivity of the network, here we also adopt another heuristic attack strategy named degree-based attack (DBA) [11]. In each iteration, we select the node of the highest degree and delete one of its links or add a link to the node with the smallest degree. Then, we update the degrees of these nodes. DBA is only based on the structure of the network, but not on the labels and attributes of the nodes.
- **Greedy Attack (GDA).** GDA is a method based on greedy algorithm. GDA calculates the fitness of each link from the candidate set, and selects the Top-K links as perturbation instead of using genetic algorithm.
- **Node Embedding Attack (NEA).** NEA is an attack strategy for network embedding proposed by Bojcheski et al. [35]. This method generates adversarial networks by maximizing the loss function of the adjacency matrix $\hat{A}$ and the matrix Z^* embedded from the adversarial network.

Parameter Setting and Convergence of GA. For DeepWalk and GA, there are many parameters. In our experiments, the parameter setting is empirically determined through balancing the performance and convergence speed shown in Table 2. Note that different parameter settings may lead to various performances of these network algorithms, but our attack strategies will be still effective in degrading them.

Figure 5 verifies the convergence of EDA. It can be seen from the experimental results of the genetic algorithm that iteration can be converged after 500 generations in most instances.

Attack on Community Detection. Community detection is one of the most common unsupervised learning problems in network science, aiming to identify the communities (a group of nodes that are closely connected) in a whole network. There are many community detection algorithms. Here, to validate the effectiveness of different attack strategies on network embedding, we would prefer to transfer nodes into vectors by DeepWalk and then realize the community detection by clustering these vectors in the embedding space by using K-means algorithm.

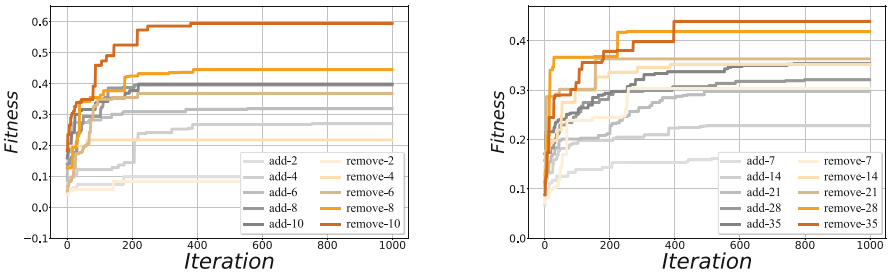
For each attack strategy, we flip the same number of links and then use the above community detection method to identify communities. We use the Normalized Mutual Information (NMI) to measure the performance [40].

NMI is used to evaluate the accuracy of a detected community. For two different categories of prediction C_p and reality C_t, it is defined as

$$\mathrm{NMI}(C_p, C_t) = \frac{\mathrm{MI}(C_p, C_t)}{\sqrt{H(C_p)H(C_t)}}, \tag{15}$$

Table 2. Parameters setting for DeepWalk and GA.

Item	Meaning	Value
l	Number of random walk	10
r	Size of window	5
w	Length of random walk	40
n	Size of representation	4/8/16/24
h	Number of population size	20
$n_{iteration}$	Number of iterations	1000
n_{elite}	Number of retained elites	4
$n_{crossover}$	Number of chromosomes for crossover	16
$n_{mutation}$	Number of chromosomes for mutation	16
p_c	Crossover rate	0.6
p_m	Mutation rate	0.08



(a) Karate network (b) Game network

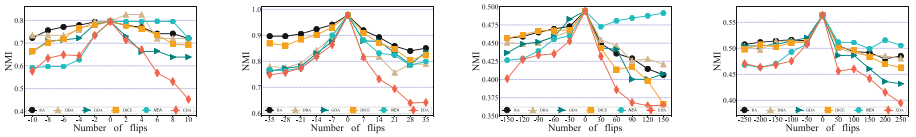
Fig. 5. The above two figures are the EDA iteration diagrams of the karate network and the game network. The gray color represents the addition of the links, and the orange color represents the deletion of the links. (Color figure online)

where MI and H represent the Mutual Information and entropy, respectively, which are defined as

$$\mathrm{MI}(C_p, C_t) = \sum_{i=1}^{|C_p|} \sum_{j=1}^{|C_t|} P(i,j) log(\frac{P(i,j)}{P(i)P'(j)}), \tag{16}$$

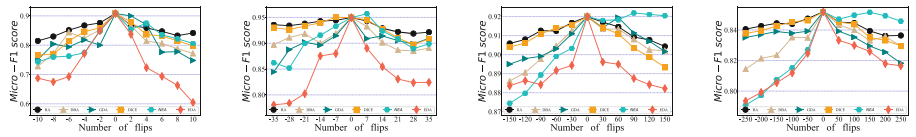
$$H(C_p) = \sum_{i=1}^{|C_p|} P(i) log(P(i)), \tag{17}$$

$$H(C_t) = \sum_{j=1}^{|C_t|} P'(j) log(P'(j)), \tag{18}$$



(a) Karate (b) Game (c) Citeseer (d) Cora

Fig. 6. NMI as the functions of the percentage of attacked links for different attack strategies on community detection.



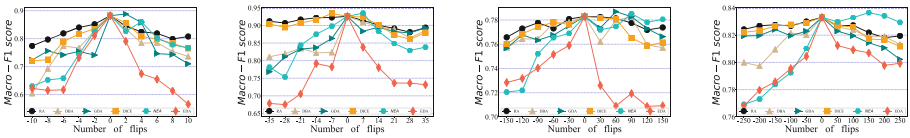
(a) Karate (b) Game (c) Citeseer (d) Cora

Fig. 7. Micro-F1 score as the functions of the percentage of attacked links for different attack strategies on node classification.

respectively, where $|C_p|$, $|C_t|$ are the number of categories in the division of prediction and that of truth, respectively, $P(i) = |C_p^i|/|V|$, $P'(j) = |C_t^j|/|V|$, and $P(i,j) = |C_p^i \cap C_t^j|/|V|$. The value of NMI indicates the similarity between $|C_p|$ and $|C_t|$, thus, the larger value, the more similar the prediction and the truth are.

For baseline attack strategies, we carry out the experiments for 100 times and present the average result in Fig. 6. In each line chart, the highest value in the middle represented is the result without suffering from attack (flip−0). The left half is the result of removing links, and the right half is the result of adding links. In general, all proposed attack strategy can effectively reduce the accuracy of community detection. More specifically, heuristic attack strategies, such as DICE and DBA, are more effective than RA. GDA always has a good effect when there are fewer links flipping. In some cases, the performance of the NEA may be better than EDA. But in the majority of cases, our proposed EDA exhibits the best overall performance, with the lowest NMI in four networks.

Attack on Node Classification. Different from the community detection problem, node classification is a typical supervised learning in network science in which the label of some nodes are known. Again, here we would like to use DeepWalk to map nodes to vectors and then use Logistic Regression (LR) [44] to classify them, namely *DeepWalk+LR*. We use the same set of benchmark networks since their real communities are known beforehand. We randomly choose 70% of nodes as the training set and treat the rest as the test set, and use Micro-F1 and Macro-F1 to evaluate the classification results. We calculate the number of true positives (TP), false positives (FP), true negatives (TN), false negatives (FN) in the instances.



(a) Karate (b) Game (c) Citeseer (d) Cora

Fig. 8. Macro-F1 score as the functions of the percentage of attacked links for different attack strategies on node classification.

Macro-F1 and Micro-F1 are then defined as

$$\text{Macro} - \text{F1} = \frac{\sum_{C_P \in C_T} F1(C_P))}{|C_T|}, \tag{19}$$

$$\text{Micro} - \text{F1} = \frac{2 * Pr * R}{Pr + R}, \tag{20}$$

respectively, where C_P, C_T are the category of in the division of prediction and that of truth, $F1(C_P)$ is the F1-measure for the label C_P, and Pr and R are calculated by

$$Pr = \frac{\sum_{C_P \in C_T} TP}{\sum_{C_P \in C_T} (TP + FP)}, \tag{21}$$

$$R = \frac{\sum_{C_P \in C_T} TP}{\sum_{C_P \in C_T} (TP + FN)}, \tag{22}$$

For multi-classification problems, Micro-F1 and Macro-F1 are used to evaluate the performance of the classification model.

Similarly, for each attack strategy, we flip the same number of links and then use the above two indicators to evaluate. For each case, we carry out the experiments for 100 times and present the average of results in Fig. 7 and Fig. 8. We find that both Micro-F1 and Macro-F1 decrease after each attack, regardless of the choice of the downstream classification algorithm (LR or KNN). In most cases, EDA still performs best, leading to the most significant drop of Micro-F1 and Macro-F1. The heuristic attack strategies DICE is more effective than RA, consistent with the results in community detection. However, it seems that in some instances, when the percentage of flipping links is relatively big, baseline methods may be more effective than EDA. It might be due to: GA has been recognized to tend to be trapped in local optimum. But still, in the majority of cases, our proposed EDA is significantly more effective than the other attack strategies.

Transferability of EDA. Generally, disturbing the distance matrix between node vectors in embedding space is equivalent to altering the similarity between nodes in the network, which will naturally affect other algorithms. We also examine the transferability of the proposed DeepWalk-based EDA method for other network algorithms.

In particular, we choose another network embedding method High-Order Proximity preserved Embedding (HOPE) [45] and two classic network algorithms, including Label Propagation Algorithm (LPA) [46] and Eigenvectors of Matrices (EM) [47], which are not based on network embedding. HOPE utilizes the generalized SVD to handle the formulation of a class of high-order proximity measurements. HOPE is widely used due to its high effectiveness and efficiency. LPA sets the label of a node identical to most of its neighbors through an iterative process, while EM directly uses the modularity matrix to study the community structure. We choose the two relatively large networks, i.e., Game and Cora, to explore the transferability of EDA.

The results are shown in Table 3. Although EDA is based on DeepWalk, it is still valid on HOPE-based node classification and community detection algorithms, i.e., HOPE+LR and HOPE+K-Means. Moreover, it is also valid on LPA and EM, which are not based on any network embedding algorithm. EDA still outperforms the other baseline attack strategies in most cases, suggesting that it has relatively strong transferability, i.e., we can generate small perturbations on the target network by EDA and destroy many network algorithms, no matter whether it is based on a certain network embedding method.

Visualization and Explanation Statistics of Flipping Links. To better understand how EDA works in reality, we visualize the perturbations of the added and deleted links to see how many of them are within the same communities or between different communities. Taking the karate club network as an example, we present its original network and the adversarial network after one-link attack, as shown in Fig. 9, where we can see that the added link is between two different communities while the deleted one is within the same community. To give more comprehensive results, we consider all the flipping links in all the experiments for experimental networks, and count the percentages of added links and deleted links within or between communities, respectively, as shown in Fig. 10. We find that most of the added links are between communities, while most of the deleted links are within the same community. This rule is quite interesting since EDA focuses on perturbing the network from embedding space without any prior knowledge of communities. One reason may be that the community is a critical structural property that matters the embedding results, and this is why EDA behaves quite well on attacking community detection algorithms.

The Position of Node Vector. Furthermore, to show what EDA does to network embedding, we also visualize node embedding vectors by t-sne method after the attack. For different percentages of rewiring links ranging from 1% to 7% in Karate club network, we obtain the nodes vectors of original and adversarial networks using DeepWalk, and then display the results in a two-dimensional space, as shown in Fig. 11. There are four communities represented by different colors. When there is no attack, the node vectors of different communities are separated, as shown in Fig. 11(a). As the number of rewiring links increases, the node vectors of different communities are being mixed gradually, and finally

Table 3. Transferability results on different attack strategies.

Dataset	Model	Metric	Baseline	R-7	R-14	R-21	R-28	R-35	A-7	A-14	A-21	A-28	A-35
Game	HOPE+LR	Micro-F1	Unattack	0.840	0.840	0.840	0.840	0.840	0.840	0.840	0.840	0.840	0.840
			RA	0.796	0.779	0.760	0.761	0.754	0.823	0.811	0.797	0.792	0.802
			Degree attack	0.810	0.717	0.727	0.715	0.704	0.844	0.826	0.830	0.786	0.779
			Greedy attack	0.835	0.721	0.802	0.782	0.703	0.848	0.839	0.761	0.820	0.750
			DICE	0.783	0.767	0.755	0.751	0.746	0.799	0.783	0.779	0.773	0.763
			NEA	0.724	0.727	0.799	0.711	0.798	0.823	0.784	0.771	0.819	0.787
			EDA*	**0.670**	**0.596**	**0.615**	**0.600**	**0.607**	**0.733**	**0.667**	**0.630**	**0.630**	**0.659**
		Macro-F1	Unattack	0.717	0.717	0.717	0.717	0.717	0.717	0.717	0.717	0.717	0.717
			RA	0.654	0.625	0.598	0.607	0.580	0.693	0.672	0.643	0.643	0.662
			Degree attack	0.670	0.555	0.551	0.548	0.535	0.720	0.691	0.693	0.626	0.612
			Greedy attack	0.714	0.553	0.705	0.682	0.537	0.725	0.724	0.592	0.717	0.598
			DICE	0.639	0.614	0.598	0.589	0.584	0.658	0.631	0.631	0.616	0.605
			NEA	0.545	0.532	0.669	0.521	0.663	0.683	0.619	0.607	0.695	0.637
			EDA*	**0.481**	**0.393**	**0.427**	**0.393**	**0.410**	**0.566**	**0.511**	**0.435**	**0.436**	**0.479**
	HOPE+K-Means	NMI	Unattack	0.496	0.496	0.496	0.496	0.496	0.496	0.496	0.496	0.496	0.496
			RA	0.460	0.451	0.443	0.433	0.427	0.472	0.468	0.464	0.460	0.473
			Degree attack	0.472	0.463	0.456	0.450	0.454	0.470	0.462	0.462	0.452	0.457
			Greedy attack	0.463	0.447	0.467	0.450	0.433	0.475	0.473	0.457	0.506	0.460
			DICE	0.460	0.441	0.439	0.435	0.432	0.462	0.458	0.458	0.471	0.452
			NEA	0.441	0.434	0.450	0.413	0.437	0.515	0.510	0.512	0.515	0.508
			EDA*	**0.408**	**0.408**	**0.396**	**0.370**	**0.391**	**0.439**	**0.420**	**0.405**	**0.408**	**0.392**
	LPA	NMI	Unattack	0.673	0.673	0.673	0.673	0.673	0.673	0.673	0.673	0.673	0.673
			RA	0.663	0.656	0.652	0.643	0.622	0.651	0.627	0.591	0.560	0.586
			Degree attack	0.621	**0.602**	0.614	0.604	0.596	**0.581**	**0.545**	**0.507**	**0.506**	**0.504**
			Greedy attack	0.646	0.622	0.605	0.599	0.604	0.635	0.617	0.593	0.600	0.605
			DICE	0.620	0.653	0.619	**0.588**	0.582	0.619	0.583	0.541	0.518	0.547
			NEA	0.621	0.613	0.609	0.598	**0.564**	0.620	0.607	0.608	0.604	0.604
			EDA*	**0.606**	0.603	**0.604**	0.599	0.578	0.608	0.565	0.526	0.523	0.513
	EM	NMI	Unattack	0.723	0.723	0.723	0.723	0.723	0.723	0.723	0.723	0.723	0.723
			RA	0.723	0.723	0.724	0.720	0.698	0.701	0.676	0.669	0.655	0.664
			Degree attack	**0.662**	0.680	0.678	0.680	0.693	0.696	0.680	0.670	**0.654**	0.660
			Greedy attack	0.718	0.710	0.716	0.717	0.712	0.718	0.694	0.673	0.665	0.655
			DICE	0.738	0.713	0.710	0.704	0.690	0.710	0.672	0.666	0.671	0.662
			NEA	0.703	0.712	0.716	0.736	0.757	**0.672**	0.749	0.726	0.753	0.758
			EDA*	0.672	**0.673**	**0.668**	**0.661**	**0.659**	0.685	**0.661**	**0.648**	0.657	**0.632**
Cora	Model	Metric	Baseline	R-50	R-100	R-150	R-200	R-250	A-50	A-100	A-150	A-200	A-250
	HOPE+LR	Micro-F1	Unattack	0.663	0.663	0.663	0.663	0.663	0.663	0.663	0.663	0.663	0.663
			RA	0.663	0.658	0.656	0.657	0.654	0.658	0.657	0.657	0.655	0.654
			Degree attack	0.662	0.658	0.656	0.656	0.658	0.666	0.660	0.648	0.656	0.662
			Greedy attack	0.663	0.648	0.640	0.646	0.638	0.664	0.656	0.651	0.654	0.652
			DICE	0.657	0.659	0.655	0.654	0.649	0.659	0.654	0.654	0.650	0.650
			NEA	0.653	0.650	0.643	0.640	0.635	0.660	0.660	0.657	0.663	0.668
			EDA*	**0.640**	**0.641**	**0.630**	**0.636**	**0.627**	**0.637**	**0.635**	**0.635**	**0.636**	**0.629**
		Macro-F1	Unattack	0.608	0.608	0.608	0.608	0.608	0.608	0.608	0.608	0.608	0.608
			RA	0.607	0.601	0.601	0.601	0.600	0.602	0.598	0.599	0.597	0.598
			Degree attack	0.606	0.603	0.600	0.598	0.604	0.610	0.603	0.595	0.604	0.607
			Greedy attack	0.601	0.589	0.584	0.585	0.582	0.602	0.606	0.595	0.599	0.590
			DICE	0.601	0.604	0.600	0.599	0.591	0.603	0.594	0.595	0.594	0.591
			NEA	0.596	0.596	0.588	0.583	0.574	0.603	0.603	0.602	0.609	0.616
			EDA*	**0.583**	**0.587**	**0.575**	**0.575**	**0.570**	**0.581**	**0.582**	**0.584**	**0.579**	**0.570**
	HOPE+K-Means	NMI	Unattack	0.262	0.262	0.262	0.262	0.262	0.262	0.262	0.262	0.262	0.262
			RA	0.252	0.251	0.249	0.248	0.248	0.253	0.252	0.252	0.251	0.252
			Degree attack	0.251	0.252	0.253	0.252	0.252	0.251	0.252	0.251	0.252	0.252
			Greedy attack	**0.248**	0.251	0.251	0.250	0.249	**0.248**	0.251	0.251	**0.250**	**0.249**
			DICE	0.251	0.250	0.249	0.247	**0.242**	0.251	0.251	0.253	0.252	0.252
			NEA	0.251	0.254	0.251	0.249	0.246	0.251	0.252	0.252	0.252	0.251
			EDA*	**0.248**	**0.247**	**0.243**	**0.245**	0.244	0.250	**0.250**	**0.250**	**0.250**	**0.249**
	LPA	NMI	Unattack	0.488	0.488	0.488	0.488	0.488	0.488	0.488	0.488	0.488	0.488
			RA	0.481	0.481	0.479	0.480	0.480	0.477	0.473	0.468	0.466	0.463
			Degree attack	0.480	0.481	0.480	0.478	0.480	**0.470**	**0.461**	**0.452**	**0.443**	**0.435**
			Greedy attack	**0.478**	0.480	0.481	0.479	0.479	0.473	0.471	0.466	0.459	0.455
			DICE	0.479	0.479	0.477	0.476	0.476	0.475	0.471	0.465	0.462	0.455
			NEA	0.482	0.480	0.478	0.480	0.480	0.483	0.481	0.480	0.481	0.483
			EDA*	**0.478**	**0.474**	**0.476**	**0.472**	**0.473**	0.471	0.464	0.457	0.456	0.455
	EM	NMI	Unattack	0.445	0.445	0.445	0.445	0.445	0.445	0.445	0.445	0.445	0.445
			RA	0.447	0.440	0.438	0.431	0.436	0.445	0.437	0.428	0.415	0.404
			Degree attack	0.443	0.446	0.443	0.439	0.439	0.454	0.455	0.438	0.451	0.429
			Greedy attack	0.446	0.448	0.442	0.444	0.450	0.442	0.429	0.431	0.410	0.414
			DICE	0.443	0.444	0.444	0.440	0.435	0.443	0.430	**0.412**	0.404	0.405
			NEA	**0.418**	0.444	0.464	0.459	0.452	0.457	0.443	0.473	0.465	0.459
			EDA*	0.442	**0.436**	**0.437**	**0.429**	**0.417**	**0.437**	**0.425**	0.414	**0.380**	**0.402**

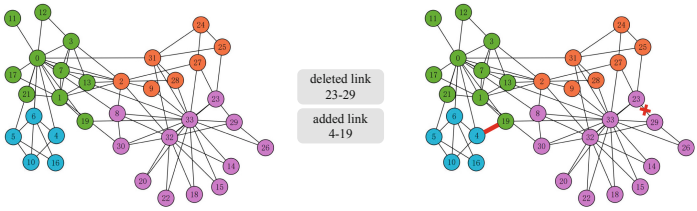
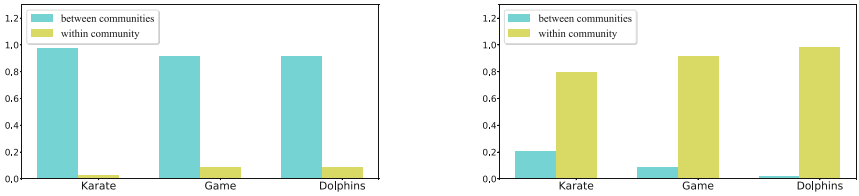


Fig. 9. The network visualization of EDA attack. Left is the original karate network, and the right is the adversarial network generated by EDA, which added a link between nodes 4 and 19, while deleted a link between nodes 23 and 29. Different colors represent different communities. (Color figure online)



(a) Added links

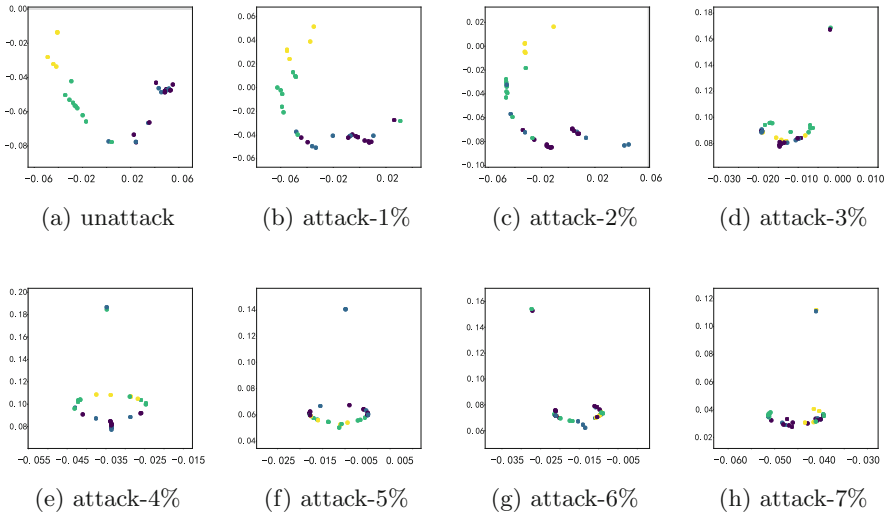
(b) Deleted links

Fig. 10. The percentages of (a) added links and (b) deleted links within community and between communities.

become inseparable, as shown in Fig. 11(h). This result demonstrates that EDA indeed has a significant effect on network embedding, and can further disturb the subsequent community detection or node classification effectively.

6 Connection with the MDATA Model

MDATA can be regarded as a special knowledge graph which expands spatio-temporal information. Thus representation learning is a common but effective strategy for knowledge construction and knowledge calculation. The network embedding attack algorithm proposed in this chapter can be migrated to MDATA for embedding reliability analysis, which could improve the robustness of MDATA in a targeted way, make the results of knowledge calculation more reliable.



(a) unattack (b) attack-1% (c) attack-2% (d) attack-3%

(e) attack-4% (f) attack-5% (g) attack-6% (h) attack-7%

Fig. 11. The visualization of node vectors in two-dimensional embedding space by attacking certain percentages of links. Different colors represent different communities. As more links are attacked by EDA, it is getting more difficult to distinguish the node vectors of different communities in the embedding space.

7 Chapter Summary

In this chapter, we propose the novel unsupervised attack strategy, namely EDA, on network embedding, with the focus on disturbing the Euclidean distance between nodes in the embedding space as much as possible with minimal changes of the network structure. Since network embedding is becoming the basis of many network algorithms, EDA can be considered as a universal attack strategy that can degenerate many downstream network algorithms.

We take the DeepWalk as the basis to design our EDA, and a number of experiments on real networks validate that EDA is more effective than a set of baseline attack strategies on community detection and node classification, no matter whether these algorithms are based on DeepWalk or other embedding methods, or even not based on embedding. Such results indicate the strong transferability of our strategy. Not that, our EDA is an attack on disturbing global network structure, and we may also focus on disturbing local structure around target nodes and links, to realize target attacks, which belongs to our future work.

Acknowledgments. This work was partially supported by the National Natural Science Foundation of China under Grant No. 61973273 and the Special Scientific Research Fund of Basic Public Welfare Profession of Zhejiang Province under Grant LGF20F020016

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
2. Hong, R. He, Y., Wu, L., Ge, Y., Wu, X.: Deep attributed network embedding by preserving structure and attribute information. IEEE Trans. Syst. Man Cybern.: Syst. (2019)
3. Barabási, A.-L., et al.: Network Science. Cambridge University Press, Cambridge (2016)
4. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inform. Sci. Technol. **58**(7), 1019–1031 (2007)
5. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: Null, pp. 475–486. IEEE (2006)
6. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
7. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
8. Athalye, A., Sutskever, I.: Synthesizing robust adversarial examples. arXiv preprint arXiv:1707.07397 (2017)
9. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE (2016)
10. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
11. Chen, J., et al.: GA based Q-attack on community detection. arXiv preprint arXiv:1811.00430 (2018)
12. Yu, S., et al.: Target defense against link-prediction-based attacks via evolutionary perturbations. arXiv preprint arXiv:1809.05912 (2018)
13. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2847–2856. ACM (2018)
14. Chen, J., Wu, Y., Xu, X., Chen, Y., Zheng, H., Xuan, Q.: Fast gradient attack on network embedding. arXiv preprint arXiv:1809.02797 (2018)
15. Wang, X., Eaton, J., Hsieh, C.-J., Wu, F.: Attack graph convolutional networks by adding fake nodes. arXiv preprint arXiv:1810.10751 (2018)
16. Perozzi, B., Al-Rfou, R., Skiena, S.: 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)
17. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ACM (2008)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
20. Fortunato, S., Hric, D.: Community detection in networks: a user guide. Phys. Rep. **659**, 1–44 (2016)
21. Guerrero, M., Montoya, F.G., Baños, R., Alcayde, A., Gil, C.: Adaptive community detection in complex networks using genetic algorithms. Neurocomputing **266**, 101–113 (2017)

22. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Disc. **23**(3), 447–478 (2011)
23. Liu, X., Shen, C., Guan, X., Zhou, Y.: We know who you are: discovering similar groups across multiple social networks. IEEE Trans. Syst. Man Cybern.: Syst. **99**, 1–12 (2018)
24. Tang, J., Aggarwal, C., Liu, H.: Node classification in signed social networks. In: Proceedings of the 2016 SIAM International Conference on Data Mining, pp. 54–62. SIAM (2016)
25. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Aggarwal, C. (ed.) Social Network Data Analytics, pp. 115–148. Springer, Boston (2011). https://doi.org/10.1007/978-1-4419-8462-3_5
26. Grover, A., Leskovec, J.: 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)
27. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077. International World Wide Web Conferences Steering Committee (2015)
28. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. arXiv preprint arXiv:1802.04407 (2018)
29. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
30. Lai, Y.-A., Hsu, C.-C., Chen, W.H., Yeh, M.-Y., Lin, S.-D.: Prune: preserving proximity and global ranking for network embedding. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5257–5266, Curran Associates Inc. (2017)
31. Dai, H., et al.: Adversarial attack on graph structured data. arXiv preprint arXiv:1806.02371 (2018)
32. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
33. Faramondi, L., et al.: Network structural vulnerability: a multiobjective attacker perspective. IEEE Trans. Syst. Man Cybern.: Syst. **99**, 1–14 (2018)
34. Waniek, M., Michalak, T.P., Wooldridge, M.J., Rahwan, T.: Hiding individuals and communities in a social network. Nat. Hum. Behav. **2**(2), 139 (2018)
35. Bojcheski, A., Günnemann, S.: Adversarial attacks on node embeddings. arXiv preprint arXiv:1809.01093 (2018)
36. Sun, M., et al.: Data poisoning attack against unsupervised node embedding methods. arXiv preprint arXiv:1810.12881 (2018)
37. Qiu, J., Dong, Y., Ma, H.. Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: unifying DeepWalk, LINE, PTE, and node2vec. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 459–467. ACM (2018)
38. Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. In: Advances in Neural Information Processing Systems, pp. 1081–1088 (2009)
39. Morin, F., Bengio, Y.: Hierarchical probabilistic neural network language model. In: Aistats, vol. 5, pp. 246–252. Citeseer (2005)
40. Ghosh, R., Lerman, K.: Community detection using a measure of global influence. In: Giles, L., Smith, M., Yen, J., Zhang, H. (eds.) SNAKDD 2008. LNCS, vol. 5498, pp. 20–35. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14929-0_2

41. Beveridge, A., Shan, J.: Network of thrones. Math Horizons **23**(4), 18–22 (2016)
42. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Inf. Retrieval **3**(2), 127–163 (2000)
43. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
44. Dreiseitl, S., Ohno-Machado, L.: Logistic regression and artificial neural network classification models: a methodology review. J. Biomed. Inform. **35**(5–6), 352–359 (2002)
45. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1105–1114. ACM (2016)
46. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**(3), 036106 (2007)
47. Newman, M.E.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. **103**(23), 8577–8582 (2006)