Data Poisoning Attack Against Graph Embedding Based Link Prediction

XINYU LU and RONGYI SUN
Shanghai Jiao Tong University

As an effective way to encode the structure of a network, node embedding has recently emerged as a powerful representation for many supervised learning tasks, such as classification and link prediction. However, the embeddings provided are not always trustworthy. Actually, there may exist malicious attackers in a network who will manipulate the edges for the purpose of changing the embedding results. Although much work has been studied on the vulnerability of supervised embedding method, little related work has analyzed the robustness of unsupervised embedding method. In this paper, we take the task of link prediction as an example, and we will give a formal formulation of attackers' behavior. Then, we design an intelligent attack mechanism to maximize the utility of an attacker. In the end, experiments based on real-world graph datasets are conducted to verify the desirable properties of the proposed mechanism. Experimental results show that our proposed mechanism can significantly affect the results of link prediction by slightly changing the graph structures.

Categories and Subject Descriptors:

General Terms: Attack, Embedding

Additional Key Words and Phrases: Data poisoning attack, node embedding, matrix factorization, approximation

1. INTRODUCTION

With the proliferation of deep neural network, node embedding has emerged as an effective tool for further learning tasks, such as classification [Perozzi et al. 2014] and link prediction [Grover and Leskovec 2016]. Recently, a large number of popular embedding methods have been proposed, including DeepWalk [Perozzi et al. 2014], LINE [Tang et al. 2015], and node2vec [Grover and Leskovec 2016]. In terms of a typical embedding task, such method analyzes the structure of a network and then represents each node with a low-dimensional vector (called embedding). As a result, the learned node embedding is general and can be potentially useful to multiple downstream tasks.

Although node embedding brings substantial advantages, the openness of the network structure offers both opportunities and incentives for malicious parties to launch attacks. In this paper, we investigate link prediction task with network embedding empowered in adversarial environments and study an important attack form, called data poisoning.

As shown in Figure 1, the network on the left hand side is its structure before data poisoning attack. And the probability of link (A,C) to emerge in the future is less than 0.5. Then the attacker controls those orange nodes (called malicious nodes) and manipulates their edges. After adding two edges from malicious nodes, the probability of link (A,C) to emerge in the future will be larger than 0.5. Such process of attack is known as data poisoning attack.

In our work, we assume that the attacker aims to maximize the error of the final prediction results and render the network structure through conducting data poisoning attack. This attack goal can be easily achieved if the attacker has the capability of controlling an overwhelming number of malicious nodes. However, in practice, the attacker usually has limited resources and he can only control a few malicious nodes. In such cases, the attack strategy plays an important role.

Here, we propose an intelligent data poisoning attack mechanism for two kinds of attacks, Target Attack(aims at specific node pair) and Availability Attack(aims at the whole network structure). Based on the mechanism, the attacker can conduct data poisoning attack intelligently. Towards this end, we formulate an optimization problem and the objective in the optimization problem is to maximize the attacker's utility. Since for availability attack, the number of the successfully attacked edges is discrete, it is hard to directly solve the optimization problem. To address this challenge, a continuous and differentiable sigmoid function is adopted to approximate the discrete component in the objective function. We solve the optimization problem by the projected gradient descent method. In summary, the main contributions of this paper are:

* We identify the pitfalls in link prediction task empowered with node embedding method.
* We propose an intelligent data poisoning attack mechanism, based on which the attacker can achieve the optimal attack goal.
* Experiments based on real-world network datasets are conducted to verify the advantages of the proposed mechanism.

2. RELATED WORK

Existing work [Dai et al. 2018] [Zugner et al. 2018] on adversarial attack against graph are limited to graph neural networks. While our work aims at the vulnerabilities of unsupervised methods on graph. Here we give previous work on graph embedding methods, adversarial attacks on graph and the related work on matrix factorization.

Unsupervised embedding methods on Graph The three most representative unsupervised methods on graph are DeepWalk [Perozzi et al. 2014], LINE [Tang et al. 2015] and Node2vec [Grover and Leskovec 2016].

Adversarial Attack on Graph There is a little work on adversarial attack on graph before. Test time attack on graph convolutional network has been investigated. [Dai et al. 2018]. Also, poisoning attack against graph is also studied. [Zugner et al. 2018] However, they only consider the attack against graph convolutional network.

Matrix Factorization Recently, it has been shown that most of the popular unsupervised methods for graph is doing implicit matrix factorization. [Qiu et al. 2017]. Moreover, poisoning attack has been demonstrated for matrix factorization problem. [Li et al. 2016] We will use this conclusion to get the gradient of loss function when solving the optimization problem.

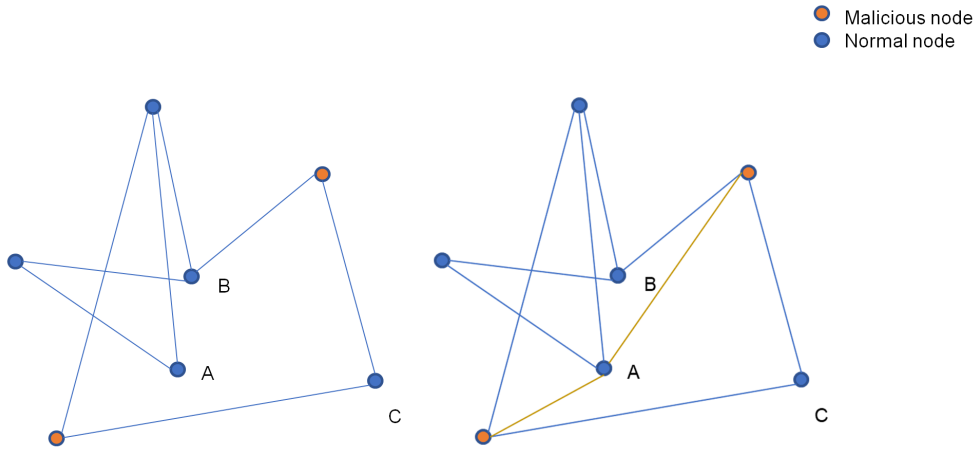


Fig. 1. Comparison of the network structure before attack and after attack. (left) Structure of the network before the data poisoning attack.(right) Structure of the network after the data poisoning attack. (blue line) The edges already exist before the attack. (Orange line) The edges added by the attacker. (Blue nodes) normal nodes. (Orange nodes) malicious nodes i.e. nodes controlled by the attacker.

3. PROBLEM SETTING

In this paper, we consider a scenario in which a graph is given with some nodes and links. The link prediction task is to predict the next likely(or unlikely) links to emerge in the network based on the structure of the graph.

The security threats considered in this paper mainly come from an attacker who aims to attack the graph structure for malicious purposes. *The goal of the attacker is to maximize the error of the derived prediction result*. We assume that the attacker can control limited number of nodes(called malicious nodes) and arbitrarily manipulate their links(add edges, remove edges or change their weights), but he cannot influence the links of the normal nodes. If there is no limitation on the ability of the attacker, he can achieve the attack goal easily through controlling a large number of malicious nodes. However, in practice, the attacker usually has limited resources and can only control a few nodes. In such cases, it is essential for the attacker to design a sophisticated attack strategy such that the attack goal can be maximally achieved. In order to assess the vulnerability of the node embedding method in the worst case, we also assume that the attacker has full knowledge of the prediction method and the structure of the network. This assumption is reasonable as it is possible for the attacker to learn the structure through eavesdropping the communications between the nodes.

Problem formulation. Suppose we are given a graph $G = (V, E)$, where V is the node set and E is the edge set. Assume that the attacker can control K malicious nodes represented as $V_m = \{v_{m1}, v_{m2}, ..., v_{mK}\}$. The edges of all the malicious nodes can be arbitrarily manipulated.

Our goal in this paper is to find an optimal attack strategy from the perspective of the attacker such that the attack goal can be maximally achieved.

4. PRELIMINARY

We first introduce the graph embedding problem and link prediction problem. Then we will give an overview of the existing algorithm for computing the embedding.

Graph embedding. Given a graph $G = (V, E)$, where V is the node set and E is the edge set, the goal of graph embedding methods is to learn a mapping from V to R_d which maps each node in the graph to a d-dimensional vector. Here, we use A to represent the adjacency matrix of the graph G. For each node v_i, we let X be the learnt node embedding matrix where X_i is the embedding of node v_i.

Link prediction. The goal of link prediction is to predict the missing edges or the edges that are most likely to emerge in the future. In this paper, given a set of node pairs $S \in V \times V$, the task is to predict a score for each node pair from the cosine similarity matrix XX^t , which means we calculate $P = XX^t$, and the value of P_{ij} represents the probability for the node pair (i, j) to emerge in the future. If $P_{ij} > 0.5$, it means (i, j) are more likely to emerge in the future, while $P_{ij} < 0.5$ means (i, j) are less likely to emerge in the future.

Now we briefly introduce a popular graph embedding method: **LINE [Tang et al. 2015]**. LINE learns the node embeddings by keeping both first-order proximity ($LINE_{1st}$) and second-order proximity ($LINE_{2nd}$) for sampled node pairs. For $LINE_{2nd}$, there is a context embedding matrix computed together with the node embedding matrix. We use Y to denote the context embedding matrix. Previous work [Qiu et al. 2017] has shown that $LINE_{2nd}$ is implicitly doing matrix factorization.

i.e. $LINE_{2nd}$ is solving the following matrix factorization problem:

$$\log(vol(G)D^{-1}AD^{-1}) - \log b = XY^t \tag{1}$$

where $vol(G) = \sum_{i,j} A_{ij}$ is the volume of graph G, D is the diagonal matrix in which each element represents the degree of the corresponding node and b is the number of negative samples. We use Z to denote the matrix that $LINE_{2nd}$ is factorizing(i.e. left-hand-side of the equation above). We denote $\Omega = \{(i, j)|Z_{ij} \neq 0\}$ as the observable elements in Z when solving matrix factorization and $\Omega_i = \{(i', j)|i' = i, Z_{i'j} \neq 0\}$ as the nonzero elements in row i of Z. With these notations defined, we now give a unified formulation for $LINE_{2nd}$:

$$\min_{X,Y} \left\| R_\Omega \left(Z - XY^t \right) \right\|_F^2 \tag{2}$$

where $[R_\Omega(M)]_{ij}$ is M_{ij} if $(i,j) \in \Omega$ and 0 otherwise, $||M||_F^2$ denotes the squared Frobenious norm of matrix M.

5. THE ATTACK MECHANISM

In this paper, the attacker conducts the data poisoning attacks for the purpose of maximizing the error of the predicted graph. In other words, the attacker aims to maximize the deviation between the outputs of the prediction before and after the data poisoning attacks. Here, we say a link is attacked successfully if the predicted result is changed from one to the other after the poisoning attack. In this section, we stand on the attackers position and discuss how to find an optimal attack strategy so that the goal of the attacker can be achieved as much as possible. Suppose the attacker is able to control K malicious nodes, and for each malicious node, the existing links are given. When conducting the data poisoning attacks to break the prediction, the attacker needs to find the optimal assignments for the links weight of malicious nodes' edges.

We assume that the attacker can manipulate the poisoned graph G by changing the weights of malicious nodes' edges. In this paper, we consider two types of adversarial goals.

Target attack: The attacker's goal is either to increase or decrease the probability (similarity score) of the target node pair to emerge in the future. For example, in social network, the attacker may be interested in increasing (or decreasing) the probability that a friendship occurs between two people. Specifically, the attacker aims to change the probability of the edge to emerge in the future whose vertices' embeddings are learnt from the poisoned graph.

Availability attack:The adversarial goals of availability attack is to reduce the performance over the whole graph. In other words, we want to maximize the number of successfully attacked links.

Based on mathematical understanding, two attacks above can be seen as a general optimization problem (i.e. optimize the error of the prediction result) and we can use projected gradient descent method to solve it. We use L to denote the loss function, then the update of the adjacent matrix A will be:

$$A^{(t+1)} \leftarrow Proj_{[0,1]}\left(A^{(t)} - s_t \cdot \nabla_A L\right) \tag{3}$$

Specific loss functions for two attack goals and their gradients will be shown in section 6. For further description, we use O to denote the similarity matrix after the attack, use R to denote the prediction result before the attack and R' to denote the prediction result after the attack:

* R_{ij}**(or** R'_{ij}**)=0**:link (i,j) already exists before the prediction.
* R_{ij}**(or** R'_{ij}**)=1**:link(i,j) is predicted to emerge in the future (i.e. $P_{ij} \geq 0.5$ or $O_{ij} \geq 0.5$)
* R_{ij}**(or** R'_{ij}**)=-1**:link(i,j) is predicted not to emerge in the future (i.e. $P_{ij} < 0.5$ or $O_{ij} < 0.5$)

6. ATTACK STRATEGY

We have introduced the general optimization problem before. In this section, we will show you the specific loss functions and their gradients for two kinds of attack.

6.1 Target Attack

In this section, the attacker's goal is to increase or decrease the probability (similarity score) of the target node pair to emerge in the future.

Thus, we can formulate the goal of the attacker as the following optimization problem, where (i,j) is the target pair (doesn't exist before the attack):

$$\max_A \pm [O_{ij}] \tag{4}$$

Here the + or - sign depends on whether the attacker wants to increase or decrease the score of the target edge. And $L_1 = \pm [O_{ij}]$

We note that:

$$\nabla_A L = \nabla_X L \cdot \nabla_A X \tag{5}$$

Based on the loss fuction above, we have:

$$\nabla_X L = 2X^t \tag{6}$$

Now to compute $\nabla_A X$, using the chain rule, we have: $\nabla_A X = \nabla_Z X \nabla_A Z$. First we show how to compute $\nabla_Z X$. For the matrix factorization problem defined in section 4, using the KKT condition, we have:

$$\sum_{j \in \Omega_i} \left(Z_{ij} - X_i Y_j^t\right) Y_j = 0 \tag{7}$$

$$\frac{\partial X_i}{\partial Z_{ij}} = Y_j \left(\sum_{j' \in \Omega_i} Y_{j'}^t Y_{j'}\right)^{-1} \tag{8}$$

Next we show how to compute $\nabla_A Z$ for LINE. In the derivation of $\nabla_A Z$, we view $vol(G)$ and D as constant. For $LINE_{2nd}$, $Z = \log\left(vol(G) D^{-1} A D^{-1}\right) - \log b$.

Since $Z_{ij} = \log\left(vol(G) d_i^{-1} A_{ij} d_j^{-1}\right) - \log b$, where $d_i = D_{ii}$. We have:

$$\frac{\nabla Z_{ij}}{\nabla A_{ij}} = \frac{1}{A_{ij}} \tag{9}$$

Once we have $\nabla_Z X$ and $\nabla_A Z$, we can compute $\nabla_A X$. Then we can compute $\nabla_A L$.

Here, we give the pseudo code for target attack in Algorithm 1:

Algorithm 1: Optimal target attack against graph embedding based link prediction

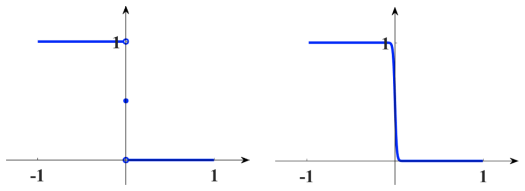
Input: The graph: $G = (V, E)$;the target pair: (i_0, j_0);the set of malicious nodes: S_m
Output: The optimal attack strategy: A
while *The convergence criterion is not satisfied* **do**
  get X and Y through conducting LINE;
  for *each* A_{ij} *such that* $i \in S_m$ *or* $j \in S_m$ **do**
    Caculate the gradient of L_1;
    Update A_{ij} according to PGD;
  end
end
return The optimal attack strategy A;

6.2 Availability Attack

In this section, the attacker conducts the data poisoning attacks for the purpose of maximizing the error of the total prediction results.

In order to address the challenge, we formulate the goal of the attacker as the following optimization problem:

$$\max_A \sum_{(i,j) \in E} \mathbf{1}\left(R_{ij} \neq R'_{ij}\right) \tag{10}$$



(a) $h_1(x)$ **(b)** $h_2(x)$ **with** $\theta = 100$

Fig. 2. (a) The function curve of $h_1(x)$. (b) The function curve of $h_2(x)$ with $\theta = 100$.

Since the value of R_{ij} and R'_{ij} is related to P_{ij} and O_{ij}, we can reformulate the optimization problem in (10) as follows:

$$\max_{A} \sum_{(i,j)\in E} \frac{1}{2}\left\{1 - sgn\left[(P_{ij} - 0.5)\cdot(O_{ij} - 0.5)\right]\right\} \tag{11}$$

Based on the method mentioned in [Miao et al. 2018], we approximate the objective function in problem (11) by the following one, where θ is a parameter to tune the steepness of the function:

$$\max_{A} \sum_{(i,j)\in E} \left\{1 - \frac{1}{1+\exp\left[-\theta\left(P_{ij} - 0.5\right)\cdot\left(O_{ij} - 0.5\right)\right]}\right\} \tag{12}$$

The basic idea behind the approximation is that function $h_1(x) = \frac{1}{2}(1 - sgnx)$ can be approximated by function $h_2(x) = 1 - \frac{1}{1+exp(-\theta x)}$ when $x \in (-1, 1)$. The parameter θ in $h_2(x)$ represents the steepness of the curve. The curves of the two functions when $\theta = 100$ are shown in Figure 2. We can see $h_2(x)$ is a good approximation of $h_1(x)$.

Thus, we have:

$$L_2 = \sum_{(i,j)\in E} \left\{1 - \frac{1}{1+\exp\left[-\theta\left(P_{ij} - 0.5\right)\cdot\left(O_{ij} - 0.5\right)\right]}\right\} \tag{13}$$

We have menioned that we adopt the projected gradient descent method to solve the problem. More specifically, in iteration t, we update A as follows:

$$A_{ij}^{(t+1)} \leftarrow Proj_{[0,1]}\left(A_{ij}^{(t)} + s_t \cdot \nabla_{A_{ij}} L_2\right) \tag{14}$$

Then we will show you how to calculate $\nabla_{A_{ij}} L$.

Based on the properties of gradient, we have

$$\nabla_{A_{ij}} L = \nabla_{X_{ij'}} L \cdot \nabla_{A_{ij}} X_{ij'} \tag{15}$$

It can be solved easily that:

$$\nabla_{X_{ij'}} L = -\frac{(0.5 - O_{ij'})\,\theta \exp\left[-\theta(P_{ij'} - 0.5)(O_{ij'} - 0.5)\right]}{\left(1 + \exp\left[-\theta(P_{ij'} - 0.5)(O_{ij'} - 0.5)\right]\right)^2} \cdot X_{j'j'} \tag{16}$$

Now to compute $\nabla_{A_{ij}} X_{ij'}$, using the chain rule, we have: $\nabla_{A_{ij}} X_{ij'} = \nabla_{Z_{ij}} X_{ij'} \nabla_{A_{ij}} Z_{ij}$. First we show how to compute $\nabla_{Z_{ij}} X_{ij'}$. For the matrix factorization problem defined in section 4, using the KKT condition, we have:

$$\sum_{j\in\Omega_i} \left(Z_{ij} - X_i Y_j^t\right) Y_j = 0 \tag{17}$$

$$\frac{\partial X_{ij'}}{\partial Z_{ij}} = \frac{Y_j}{\sum_{k\in\Omega_i} Y_{kj'} Y_k} \tag{18}$$

Next we give $\nabla_{A_{ij}} Z_{ij}$ for LINE. Same as the previous calculation, in the derivation of $\nabla_{A_{ij}} Z_{ij}$, we view $vol(G)$ and D as constant. For $LINE_{2nd}$, $Z = \log\left(vol(G)D^{-1}AD^{-1}\right) - \log b$.

Since $Z_{ij} = \log\left(vol(G)d_i^{-1}A_{ij}d_j^{-1}\right) - \log b$, where $d_i = D_{ii}$. We have:

$$\frac{\nabla Z_{ij}}{\nabla A_{ij}} = \frac{1}{A_{ij}} \tag{19}$$

Once we have $\nabla_Z X$ and $\nabla_A Z$, we can compute $\nabla_A X$. Then we can compute $\nabla_A L$.

The pseudo code of availability attack is shown in Algorithm 2.

Algorithm 2: Optimal availability attack against graph embedding based link prediction

Input: The graph: $G = (V, E)$;the set of malicious nodes: S_m
Output: The optimal attack strategy: A
Initialize the optimal strategy A;
while *The convergence criterion is not satisfed* **do**
  get X and Y through conducting LINE;
  for *each* A_{ij} *such that* $i \in S_m$ *or* $j \in S_m$ **do**
    Calculate the gradient of L_2;
    Update A_{ij} according to PGD;
  end
end
return The optimal attack strategy A;

7. EXPERIMENTS

We conduct experiments based on real-world graph dataset to verify the performance of the proposed attack mechanism.

7.1 Dataset and Baseline

In this section, we introduce the adopted real-world graph dataset, the baseline method which is compared with the proposed mechanism, and the performance measure.

7.1.1 *Dataset.* To verify the advantages of the proposed attack mechanism, we adopt the following real-world graph dataset.

Social circles: Facebook This dataset consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using Facebook app. The dataset includes node features (profiles), circles, and ego networks. The network contains 4039 nodes and 88234 edges.

7.1.2 *Baseline.* We compare the proposed attack mechanism with one baseline method: Rand_attack.

In the Rand_attack method, the attacker does not consider any strategy, and he just randomly sets the weights of each malicious node's edges. This method introduces less overhead to the attacker, as he does not need to take effort to analyze the structure of the graph.

7.1.3 *Performance Measure.* In order to evaluate the performance of the proposed attack mechanism, we compare the prediction results before and after the data poisoning attacks. For target data, we show the value of O_{ij}. For availability attack, we adopt the change rate as the measure metric. The change rate is defined as

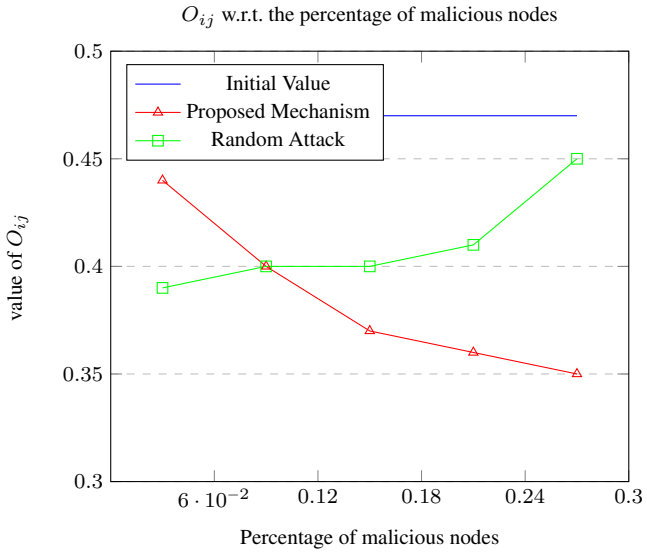


Fig. 3. The figure shows the value of O_{ij} w.r.t. the percentage of the malicious workers.

$\frac{\sum_{(i,j)\in E} 1(R_{ij} \neq R'_{ij})}{N}$, where R_{ij} and R'_{ij} are the prediction results before and after the data poisoning attacks. N is the total number of edges. Since the goal of the attacker is to maximize the error of the prediction result, thus, the larger the change rate, the better the method.

7.2 Experiment Results for Target Attack

When conducting the data poisoning attack, we assume that the attacker cannot manipulate the edges of normal nodes, but he can manipulate the edges of malicious nodes. Thus, the number of the malicious nodes controlled by the attacker plays an important role in the attack. If the attacker is able to control an overwhelming number of malicious nodes, the goal of the attacker can be easily achieved. However, in practice, the attacker can only control a limited number of malicious nodes due to the limitation of his ability. In this experiment, we consider the scenarios where the percentage of malicious nodes is low, and evaluate the performance of the proposed mechanism when the percentage is varying. We vary the percentage of the malicious nodes from 0.03 to 0.27 and set the objective as minimizing O_{ij}. All the experiments are conducted 3 times and we report the average results. The value of O_{ij} is shown in Figure 3.

From this figure, we can see the proposed attack mechanism performs better than the baseline method overall. When the percentage of the malicious nodes is very low (e.g., 3%), since the malicious nodes are too few to change the final prediction results much, the advantage of the proposed mechanism is small. However, when the percentage of the malicious nodes increases, the advantage of the proposed attack scheme becomes bigger.

7.3 Experiment Results for Availability Attack

In this experiment, we consider the scenarios where the percentage of malicious nodes is low (3%), and compare the performance of the proposed mechanism with the performance of Rand_Attack. The measure metric used is the change rate. The experiment is conducted 3 times and we report the average results. The results are shown in table 1.

Method	Proposed Mechanism	Random Attack
Chage Rate	0.302	0.289

Table 1

From this table, we can see the proposed attack mechanism performs better than the baseline method. Since the malicious nodes are too few to change the final prediction results much, the advantage of the proposed mechanism is small. However, we can predict that when the percentage of the malicious nodes increases, the advantage of the proposed attack scheme will become bigger.

8. CONCLUSION

In this paper, we investigate the node embedding in adversarial environments and study the data poisoning attack against link prediction task with the node embedding empowered. In order to find an effective attack strategy for the attacker, we design an intelligent attack mechanism by solving an optimization problem. With the derived optimal attack strategy, the attacker can maximize his attack utility. The experimental results based on real-world datasets demonstrate that the proposed attack mechanism can achieve higher attack utility with very few malicious workers.

REFERENCES

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research*.

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

Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in Neural Information Processing Systems (29)*. Curran Associates, 1885–1893.

Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. 2018. Atack under Disguise: An Intelligent Data Poisoning Atack Mechanism in Crowdsourcing. *WWW* (April 2018).

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. 710–710.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2017. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. *CoRR* (2017). http://arxiv.org/abs/1710.02971

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. *ACM* (2015).

Daniel Zugner, Amir Akbarnejad, and Stephan Gunnemanni. 2018. Adversarial attacks on neural networks for graph data. *KDD* (2018).