Adversarial Attacks on Node Embeddings via Graph Poisoning

Aleksandar Bojchevski [1] **Stephan Günnemann** [1]

Abstract

The goal of network representation learning is to learn low-dimensional node embeddings that capture the graph structure and are useful for solving downstream tasks. However, despite the proliferation of such methods, there is currently no study of their robustness to adversarial attacks. We provide the first adversarial vulnerability analysis on the widely used family of methods based on random walks. We derive efficient adversarial perturbations that poison the network structure and have a negative effect on both the quality of the embeddings and the downstream tasks. We further show that our attacks are transferable since they generalize to many models and are successful even when the attacker is restricted.

1. Introduction

Unsupervised node embedding (network representation learning) approaches are becoming increasingly popular and achieve state-of-the-art performance on many network learning tasks (Cai et al., 2018). The goal is to embed each node in a low-dimensional feature space such that the graph's structure is captured. The learned embeddings are subsequently used for downstream tasks such as link prediction, node classification, community detection, and visualization. Among the variety of proposed approaches, techniques based on random walks (RWs) (Perozzi et al., 2014; Grover & Leskovec, 2016) are often employed since they incorporate higher-order relational information. Given the increasing popularity of these methods there is a strong need for an analysis of their robustness. In particular, we aim to study the existence and effects of *adversarial perturbations*. A large body of research shows that both traditional and deep learning methods can easily be fooled/attacked: even slight deliberate data perturbations can lead to wrong results (Goodfellow et al., 2015; Mei & Zhu, 2015; Carlini & Wagner, 2017; Liang et al., 2018; Cissé et al., 2017; Lin et al., 2017; Chen et al., 2017a).

While adversarial attacks for graph models have been proposed recently (Zügner et al., 2018; Dai et al., 2018a; Zügner & Günnemann, 2019), they are all limited to the semi-supervised learning setting. In contrast, this is the first work that studies adversarial perturbations for *unsupervised* embeddings. This is critical, since especially in domains where graph embeddings are used (e.g. the web) adversaries are common and false data is *easy to inject*: e.g. spammers can easily create fake followers on social networks. Can we construct attacks that do no rely on a specific downstream task? Are node embedding methods just as easily fooled, since compared to semi-supervised models they do not incorporate a supervision signal that can be exploited?

Barring the few above-mentioned attacks on graphs most existing works on adversarial attacks perturb the features of individual instances. In our case however, since we are operating on plain graph data (no features are available) we perturb the interactions (edges) between instances instead. Manipulating the network structure (the graph) is a common scenario with link spam farms (Gyöngyi & Garcia-Molina, 2005) and Sybil attacks (Yu et al., 2006) as typical examples.

Moreover, since node embeddings are typically trained in an unsupervised and transductive fashion we cannot rely on a single end-task that our attack might exploit to find appropriate perturbations, and we have to handle a challenging poisoning attack where the model is learned *after* the attack. That is, the model cannot be assumed to be static as in most existing attacks. Lastly, since graphs are discrete, gradient-based approaches (Li et al., 2016; Mei & Zhu, 2015) for finding adversarial perturbations that were designed for continuous data are not well suited. In particular, for methods based on random walks the gradient computation is not directly possible since sampling random walks is not a differentiable operation. The question is how to design efficient algorithms that are able to find adversarial perturbations in such a challenging – discrete and combinatorial – graph domain?

We propose a principled strategy for adversarial attacks on unsupervised node embeddings. Exploiting results from eigenvalue perturbation theory (Stewart, 1990) we are able to efficiently solve a challenging bi-level optimization prob-

[1]Technical University of Munich, Germany. Correspondence to: Aleksandar Bojchvski <a.bojchevski@in.tum.de>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

lem associated with the poisoning attack. We assume an attacker with full knowledge about the data and the model, thus ensuring reliable vulnerability analysis in the worst case. Nonetheless, our experiments on transferability demonstrate that our strategy generalizes – attacks learned based on one model successfully fool other models as well. Additionally, we study the effect of restricting the attacker.

Overall, we shed light on an important problem that has not been studied so far. We show that node embeddings are sensitive to adversarial attacks. Relatively few changes are needed to significantly damage the quality of the embeddings even in the scenario where the attacker is restricted. Furthermore, our paper highlights that more work is needed to make node embeddings robust to adversarial perturbations and thus readily applicable in production systems.

2. Related Work

We focus on unsupervised node embedding approaches based on random walks (RWs) and further show how one can easily apply a similar analysis to attack spectral-based node embeddings. For a recent survey, also of other non-RW based approaches, we refer to Cai et al. (2018). Moreover, while many semi-supervised learning methods (Defferrard et al., 2016; Kipf & Welling, 2017; Klicpera et al., 2019) have been introduced, we focus on unsupervised methods since they are often used in practice due to their flexibility in simultaneously solving various downstream tasks.

Adversarial attacks. Attacking machine learning models has a long history, with seminal works on SVMs and logistic regression (Biggio et al., 2012; Mei & Zhu, 2015). Neural networks were also shown to be highly sensitive to small adversarial perturbations to the input (Szegedy et al., 2014; Goodfellow et al., 2015). While most works focus on image classification, recent works also study adversarial examples in other domains (Grosse et al., 2017; Liang et al., 2018).

Different taxonomies exist characterizing the adversaries based on their goals, knowledge, and capabilities (Papernot et al., 2016; Biggio et al., 2017; Muñoz-González et al., 2017). The two dominant attacks types are poisoning attacks targeting the training data (the model is trained *after* the attack) and evasion attacks targeting the test data/application phase (the learned model is assumed fixed). Compared to evasion attacks, poisoning attacks are far less studied (Mei & Zhu, 2015; Li et al., 2016; Koh & Liang, 2017; Muñoz-González et al., 2017; Chen et al., 2017a) since they usually require solving a challenging bi-level optimization problem.

Attacks on semi-supervised graph models. The robustness of semi-supervised graph classification methods to adversarial attacks has recently been analyzed (Zügner et al., 2018; Dai et al., 2018a; Zügner & Günnemann, 2019). The first work, introduced by Zügner et al. (2018), linearizes a graph convolutional network (GCN) (Kipf & Welling, 2017) to derive a closed-form expression for the change in class probabilities for a given edge/feature perturbation. They calculate a score for each possible edge flip based on the classification margin and greedily pick the top edge flips with highest scores. Later, Dai et al. (2018a) proposed a reinforcement (Q-)learning formulation where they decompose the selection of relevant edge flips into selecting the two end-points. Zügner & Günnemann (2019) develop a general attack on the training procedure of GCN using meta-gradients. All three approaches focus on the semi-supervised graph classification task and take advantage of the supervision signal to construct the attacks. In contrast, our work focuses on general attacks on *unsupervised* node embeddings applicable to many downstream tasks.

Manipulating graphs. There is an extensive literature on optimizing the graph structure to manipulate: information spread in a network (Khalil et al., 2014; Chen et al., 2016), user opinions (Chaoji et al., 2012; Amelkin & Singh, 2017), shortest paths (Phillips, 1993; Israeli & Wood, 2002), page rank scores (Csáji et al., 2014), and other metrics (Chan et al., 2014). In the context of graph clustering, Chen et al. (2017b) measure the performance changes when injecting noise to a bi-partite graph of DNS queries, but do not focus on automatically generating attacks. Zhao et al. (2018) study poisoning attacks on multi-task relationship learning, although they exploit relations between tasks, they still deal with the classic scenario of i.i.d. instances within each task.

Robustness and adversarial training. Robustification of machine learning models, including graph based models (Bojchevski et al., 2017; Zügner & Günnemann, 2019), has been studied and is known as adversarial/robust machine learning. These approaches are out of the scope for this paper. Adversarial training, e.g. via GANs (Dai et al., 2018b), is similarly beyond our scope since the goal is to improve the embeddings, while our goal is to asses the vulnerability of existing embedding methods to adversarial perturbations.

3. Attacking Node Embeddings

We study poisoning attacks on the graph structure – the attacker is capable of adding or removing (flipping) edges in the original graph within a given budget. We focus mainly on approaches based on random walks and extend the analysis to spectral approaches (Sec. 6.2 in the appendix).

3.1. Background and Preliminaries

Let $G = (V, E)$ be an undirected unweighted graph where V is the set of nodes, E is the set of edges, and $A \in \{0,1\}^{|V|\times|V|}$ is the adjacency matrix. The goal of network representation learning is to find a low-dimensional embedding $z_v \in \mathbb{R}^K$ for each node with $K \ll |V|$. This dense

low-dimensional representation should preserve information about the network structure – nodes similar in the original network should be close in the embedding space. DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) learn an embedding based on RWs by adapting the skip-gram architecture (Mikolov et al., 2013) for learning word embeddings. They sample finite (biased) RWs and use the co-occurrence of node-context pairs in a given window in each RW as a measure of similarity. To learn z_v they maximize the probability of observing v's neighborhood.

3.2. Attack Model

We denote with $\hat{A}$ the adjacency matrix of the graph obtained after the attacker has modified certain entries in A. We assume the attacker has a given, fixed budget and is only capable of modifying f entries, i.e. $||\hat{A} - A||_0 = 2f$ (times 2 since G is undirected). The goal of the attacker is to damage the quality of the learned embeddings, which in turn harms subsequent learning tasks such as node classification or link prediction that use the embeddings. We consider both a general attack that aims to degrade the embeddings of the network as a whole, and a targeted attack that aims to damage the embeddings regarding a specific target / task.

The quality of the embeddings is measured by the loss $\mathcal{L}(A, Z)$ of the model under attack, with lower loss corresponding to higher quality, where $Z \in \mathbb{R}^{N \times K}$ is the matrix containing the embeddings of all nodes. Thus, the goal of the attacker is to *maximize* the loss. We can formalize this as the following bi-level optimization problem:

$$\hat{A}^* = \arg \max_{\hat{A} \in \{0,1\}^{N \times N}} \mathcal{L}(\hat{A}, Z^*) \qquad Z^* = \min_Z \mathcal{L}(\hat{A}, Z)$$

$$\text{subj. to } ||\hat{A} - A||_0 = 2f \;,\; \hat{A} = \hat{A}^T$$

Here, Z^* is always the *'optimal'* embedding resulting from the (to be optimized) graph $\hat{A}$, i.e. it minimizes the loss, while the attacker tries to maximize the loss. Solving such a problem is challenging given its discrete and combinatorial nature, thus we derive efficient approximations.

3.3. General Attack

The first step in RW-based embedding approaches is to sample a set of random walks that serve as a training corpus further complicating the bi-level optimization problem. We have $Z^* = \min_Z \mathcal{L}(\{r_1, r_2, \dots\}, Z)$ with $r_i \sim RW(\hat{A})$, where RW is an intermediate stochastic procedure that generates RWs given the graph $\hat{A}$ which we are optimizing. By flipping (even a few) edges in the graph, the attacker necessarily changes the set of possible RWs, thus changing the training corpus. Therefore, this sampling procedure precludes any gradient-based methods. To tackle this challenge we leverage recent results that show that (given certain assumptions) RW-based embedding approaches are implicitly factorizing the Pointwise Mutual Information (PMI) matrix (Yang & Liu, 2015; Qiu et al., 2018). We study DeepWalk as an RW-based representative approach since it's one of the most popular methods and has many extensions. Specifically, we use the results from Qiu et al. (2018) to sidestep the stochasticity induced by sampling random walks.

Lemma 1 (Qiu et al. (2018))**.** *DeepWalk is equivalent to factorizing* $\hat{M} = \log(\max(M, 1))$ *with*

$$M = \tfrac{vol(A)}{T \cdot b} S, \quad S = \big(\sum_{r=1}^{T} P^r\big) D^{-1}, \quad P = D^{-1} A \quad (1)$$

where the embedding Z^* *is obtained by the Singular Value Decomposition (SVD) of* $\hat{M} = U \Sigma V^T$ *using the top-*K *largest singular values / vectors, i.e.* $Z^* = U_K \Sigma_K^{1/2}$.

Here, D is the diagonal degree matrix with $D_{ii} = \sum_j A_{ij}$, T is the window size, b is the number of negative samples and $vol(A) = \sum_{i,j} A_{ij}$ is the volume. Since M is sparse and has many zero entries the matrix $\log(M)$ where the $\log$ is elementwise is ill-defined and dense. To cope with this, similar to the Shifted Positive PMI (PPMI), approach the elementwise maximum is introduced to form $\hat{M}$. Using this insight we see that DeepWalk is equivalent to optimizing $\min_{\hat{M}_K} ||\hat{M} - \hat{M}_K||_F^2$ where $\hat{M}_K$ is the best rank-K approximation to $\hat{M}$. This in turn means that the loss for DeepWalk when using the *optimal* embedding Z^* for a given graph A is $\mathcal{L}_{DW_1}(A, Z^*) = \big[\sum_{p=K+1}^{|V|} \sigma_p^2\big]^{1/2}$ where σ_p are the singular values of $\hat{M}(A)$ sorted decreasingly $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_{|V|}$. This result shows that we do not need to construct random walks, nor do we have to (explicitly) learn the embedding Z^* – it is implicitly considered via the singular values of $\hat{M}(A)$. Accordingly, we have transformed the bi-level problem into a single-level optimization problem. However, maximizing $\mathcal{L}_{DW_1}$ is still challenging due to the SVD and the discrete nature of the problem.

Gradient based approach. Maximizing $\mathcal{L}_{DW_1}$ with a gradient-based approach is not straightforward since we cannot easily backpropagate through the SVD. To tackle this challenge we exploit ideas from eigenvalue perturbation theory (Stewart, 1990) to efficiently approximate $\mathcal{L}_{DW_1}(A)$ in closed-form without needing to recompute the SVD.

Theorem 1. *Let* A *be the initial adjacency matrix and* $\hat{M}(A)$ *be the respective co-occurrence matrix. Let* u_p *be the* p*-th eigenvector corresponding to the* p*-th largest eigenvalue of* $\hat{M}$*. Given a perturbed matrix* A'*, with* $A' = A + \Delta A$*, and the respective change* $\Delta \hat{M}$*,* $\mathcal{L}_{DW_1}(A') \approx \big[\sum_{p=K+1}^{N} \big(u_p^T(\hat{M} + \Delta \hat{M}) u_p\big)^2\big]^{1/2} =: \mathcal{L}_{DW_2}(A')$ *is an approximation of the loss and the error is bounded by* $|\mathcal{L}_{DW_1}(A') - \mathcal{L}_{DW_2}(A')| \leq ||\Delta \hat{M}||_F$.

The proof is given in the appendix. For a small ΔA and thus small $\Delta \hat{M}$ we obtain a very good approximation, and

if $\Delta A = \Delta \hat{M} = 0$ then the loss is exact. Intuitively, we can think of using eigenvalue perturbation as analogous to taking the gradient of the loss w.r.t. $\hat{M}(A)$. Now, gradient-based optimization is efficient since $\nabla_A \mathcal{L}_{DW_2}(A)$ avoids recomputing the eigenvalue decomposition. The gradient provides useful information for a small ϵ change, however, here we are considering discrete flips, i.e. $\epsilon = \pm 1$ so its usefulness is limited. Furthermore, using gradient-based optimization requires a dense instantiation of the adjacency matrix which has complexity $O(|V|^2)$ in both runtime and memory, infeasible for large graphs. This motivates the need for our more advanced approach.

Sparse closed-form approach. Our goal is to efficiently compute the change in the loss $\mathcal{L}_{DW_1}(A)$ given a set of flipped edges. To do so we will analyze the change in the spectrum of some of the intermediate matrices and then derive a bound on the change in the spectrum of the co-occurrence matrix, which in turn will give an estimate of the loss. First, we need some results.

Lemma 2. *The matrix S in Eq. 1 is equal to $S = U(\sum_{r=1}^{T} \Lambda^r)U^T$ where the matrices U and Λ contain the eigenvectors and eigenvalues solving the generalized eigenproblem $Au = \lambda Du$.*

The proof is given in the appendix. We see that the spectrum of S (and the spectrum of M by taking the scalars into account) is obtainable from the generalized spectrum of A. In contrast to Lemma 2, Qiu et al. (2018) factorize S using the (non-generalized) spectrum of $A_{norm} := D^{-1/2}AD^{-1/2}$. As we will show, our formulation using the generalized spectrum of A is key for an efficient approximation.

Let $A' = A + \Delta A$ be the adjacency matrix after the attacker performed some edge flips. As above, by computing the generalized spectrum of A', we can estimate the spectrum of the resulting S' and M'. However, recomputing the eigenvalues λ' of A' for every possible set of edge flips is still not efficient for large graphs, preventing an effective application of the method. Thus, we derive our first main result: an efficient approximation bounding the change in the singular values of M' for any edge flip.

Theorem 2. *Let ΔA be a matrix with only 2 non-zero elements, namely $\Delta A_{ij} = \Delta A_{ji} = 1 - 2A_{ij}$ corresponding to a single edge flip (i, j), and ΔD the respective change in the degree matrix, i.e. $A' = A + \Delta A$ and $D' = D + \Delta D$. Let u_y be the y-th generalized eigenvector of A with generalized eigenvalue λ_y. Then the generalized eigenvalue λ'_y of A' solving $A'u'_y = \lambda'_y D'u'_y$ is approximately $\lambda'_y \approx \lambda_y + \Delta\lambda_y := \tilde{\lambda}'_y$ with:*

$$\Delta\lambda_y = \Delta w_{ij}(2u_{yi} \cdot u_{yj} - \lambda_y(u_{yi}^2 + u_{yj}^2)) \qquad (2)$$

where u_{yi} is the i-th entry of the vector u_y, and $\Delta w_{ij} = (1 - 2A_{ij})$ indicates the edge flip, i.e ± 1.

The proof is given in the appendix. By working with the generalized eigenvalue problem in Theorem 2 we were able to express A' and D' after flipping an edge as *additive* changes to A and D, this in turn enabled us to leverage results from eigenvalue perturbation theory to efficiently approximate the change in the spectrum. If we used A_{norm} instead, the change to A'_{norm} would be multiplicative hindering efficient approximation. Using Eq. 2, instead of recomputing λ' we only need to compute $\Delta\lambda$ to obtain the approximation $\tilde{\lambda}'$ significantly reducing the complexity when evaluating different edge flips (i, j). Using this result, we can now efficiently bound the change in the singular values of S'.

Lemma 3. *Let A' be defined as before and S' be the resulting matrix. The singular values of S' are bounded: $\sigma_p(S') \leq \tilde{\sigma}_p := \frac{1}{d'_{min}} \cdot \left| \sum_{r=1}^{T} (\tilde{\lambda}'_{\pi(p)})^r \right|$ where π is a permutation simply ensuring that the final $\tilde{\sigma}_p$ are sorted decreasingly, and d'_{min} is the smallest degree in A'.*

We provide the proof in the appendix. Now we can efficiently compute the loss for a rank-K factorization of M', which we would obtain when performing the edge flip (i, j), i.e. $\mathcal{L}_{DW_3}(A') = \frac{vol(A)+2\Delta w_{ij}}{T \cdot b} \left[\sum_{p=K+1}^{|V|} \tilde{\sigma}_p^2 \right]^{1/2}$, where $\tilde{\sigma}_p$ is obtained by applying Lemma 3 and Theorem 2 and the leading constants follow from Lemma 1.

While the original loss $\mathcal{L}_{DW_1}$ is based on the matrix $\hat{M} = \log(\max(M, 1))$, there are unfortunately currently no tools available to analyze the (change in the) spectrum of $\hat{M}$ given the spectrum of M. Therefore, we use $\mathcal{L}_{DW_3}$ as a surrogate loss for $\mathcal{L}_{DW_1}$ (Yang et al. (2015) similarly exclude the element-wise logarithm). As our experiments show, the surrogate loss is effective and we can successfully attack the node embeddings that factorize the actual co-occurrence matrix $\hat{M}$, as well as the original skip-gram model. Similarly, spectral embedding methods (von Luxburg, 2007), factorize the graph Laplacian and have a strong connection to the RW based approaches. We provide an analysis of their adversarial vulnerability in the appendix (Sec. 6.2).

The overall algorithm. Our goal is to maximize $\mathcal{L}_{DW_3}$ by performing f edge flips. While Eq. 2 enables us to efficiently compute the loss for a single edge, there are still $\mathcal{O}(|V|^2)$ possible flips. To reduce the complexity when *adding* edges we instead form a candidate set by randomly sampling C candidate pairs (non-edges). This introduces a further approximation that nonetheless works well in practice. Since real graphs are usually sparse, for *removing*, all edges are viable candidates with one random edge set aside for each node to ensure we do not have singleton nodes. For every candidate we compute its impact on the loss via $\mathcal{L}_{DW_3}$ and greedily choose the top f flips.[1]

[1] Periodically recomputing the exact eigenvalues when using the greedy approach did not show any benefits. Code and data available at https://www.kdd.in.tum.de/node_embedding_attack.

The runtime complexity of our overall approach is then: $\mathcal{O}(|V|\cdot|E|+C\cdot|V|\log|V|)$. First, we can compute the generalized eigenvectors of A in a sparse fashion in $\mathcal{O}(|V|\cdot|E|)$. Then we sample C candidate edges, and for each we can compute the approximate eigenvalues in constant time (Theorem 2). To obtain the final loss, we sort the values leading to the overall complexity. For the examined datasets the wall-clock time for our approach is negligible: on the order of few seconds when calculating the change in eigenvalues. Furthermore, our approach is trivially parallelizable since every candidate edge flip can be evaluated in parallel.

3.4. Targeted Attack

If the goal of the attacker is to attack a specific target node $t \in V$, or a specific downstream task, it is suboptimal to maximize the overall loss via $\mathcal{L}_{DW_*}$. Rather, we should define some other *target specific* loss that depends on t's embedding – replacing the loss function of the *outer* optimization by another one operating on t's embedding. Thus, for any edge flip (i, j) we now need the change in t's embedding – meaning changes in the eigen*vectors* – which is inherently more difficult to compute compared to changes in eigen/singular-*values*. We study two cases: misclassifying a target node (i.e. node classification tasks) and manipulating the similarity of node pairs (i.e. link prediction task).

Surrogate embeddings. We define surrogate embeddings such that we can efficiently estimate the change for a given edge flip. Specifically, instead of performing an SVD of M (or equivalently S scaled) we define $\bar{Z}^* = U(\sum_{r=1}^{T}\Lambda^r)$, as in Lemma 2. Experimentally, using $\bar{Z}^*$ instead of Z^* as the embedding showed no significant change in the performance on downstream tasks. While we use these surrogate embeddings to select the adversarial edges, during evaluation we use the standard embeddings produced by DeepWalk. Now, by approximating the generalized eigenvectors of A', we can also approximate $\bar{Z}^*(A')$ in closed-form:

Theorem 3. *Let $\Delta A, \Delta D$ and Δw_{ij} be defined as before, and $\Delta\lambda_y$ be the change in the y-th generalized eigenvalue λ_y as derived in Theorem 2. Then, the y-th generalized eigenvector u'_y of A' after performing the edge flip (i, j) can be approximated with:*

$$u'_y \approx u_y - \Delta w_{ij}(A - \lambda_y D)^+\big(-\Delta\lambda_y u_y \circ d + E_i(u_{yj} - \lambda_y u_{yi}) + E_j(u_{yi} - \lambda_y u_{yj})\big) \quad (3)$$

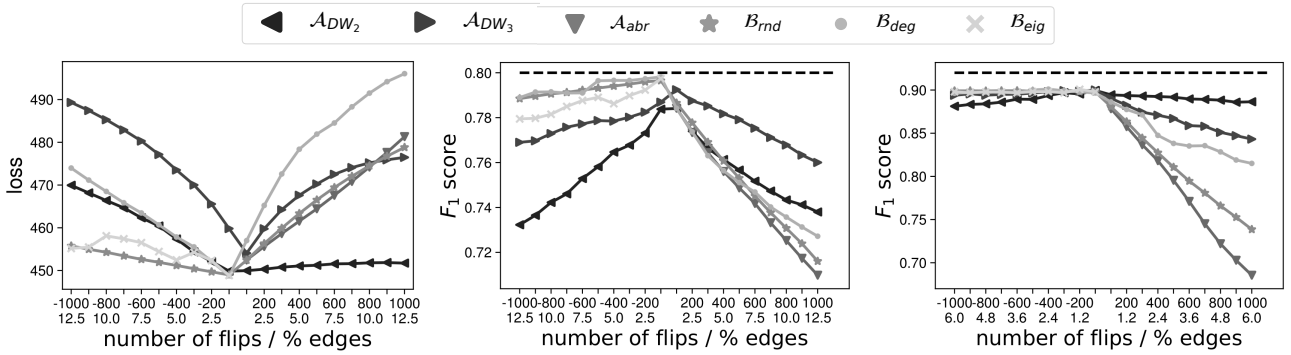
where $E_i(x)$ returns a vector of zeros except at position i where the value is x, d is a vector of the node degrees, $\circ$ is the Hadamard product, and $(\cdot)^+$ is the pseudo-inverse.

We provide the proof in the appendix. Computing Eq. 3 seems expensive at first due to the pseudo-inverse term. However, note that this term does not depend on the particular edge flip we perform. Thus, we can pre-compute it once and furthermore, parallelize the computation for each y. The additional complexity of computing the pseudo-inverse for all y is $\mathcal{O}(K \cdot |V|^{2.373})$. Similarly, we can pre-compute $u_y \circ d$, while the rest of the terms are all computable in $O(1)$. Overall, the wall-clock time for computing the change in the eigenvectors is on the order of few minutes. For any edge flip we can now efficiently compute the optimal embedding $\bar{Z}^*(A')$ using Eqs. 2 and 3. The t-th row of $\bar{Z}^*(A')$ is the desired embedding for a target node t after the attack.

Targeting node classification. Our goal is to missclassify a target node t given a downstream node classification task. To specify the targeted attack we need to define the candidate flips and the target-specific loss responsible for scoring the candidates. We let the candidate set contain all edges (and non-edges) directly incident to the target node, i.e. $\{(v, t)|v \neq t\}$. We restricted our experiments to such candidate flips since initial experiments showed that they can do significantly more damage compared to candidate flips in other parts of the graph. This intuitively makes sense since the further away we are from node t we can exert less influence on it. Zügner et al. (2018) show similar results (e.g. see their indirect attack). Note that for the general (non-targeted) attack all edges/non-edges are viable candidates.

To obtain the loss, we first pre-train a classifier on the clean embedding $\bar{Z}^*$. Then we predict the class probabilities p_t of the target t using the compromised $\bar{Z}^*_{t,\cdot}$ estimated for a given candidate flip and we calculate the classification margin $m(t) = p_{t,c(t)} - \max_{c\neq c(t)} p_{t,c}$, where $c(t)$ is the ground-truth class for t. That is, our loss is the difference between the probability of the ground truth and the next most probable class after the attack. Finally, we select the top f flips with smallest margin m (note when $m(t) < 0$ node t is misclassified). In practice, we average over ten randomly trained logistic regression classifiers. In future work we plan to treat this as a tri-level optimization problem.

Targeting link prediction. The goal of this targeted attack is given a set of target node pairs $\mathcal{T} \subset V \times V$ to decrease the similarity between the nodes that have an edge, and increase the similarity between nodes that do not have an edge by modifying *other* parts of the graph (i.e. we disallow to directly flip pairs in $\mathcal{T}$). For example, in an e-commerce graph representing users and items the goal might be to increase the similarity between a certain item and user by adding/removing connections between other users/items. To achieve this goal, we first train an initial clean embedding on the graph excluding the edges in $\mathcal{T}$. Then, for a candidate flip we estimate the embedding $\bar{Z}^*$ (Eqs. 2 and 3) and use it to calculate the average precision score (AP score) on the target set $\mathcal{T}$, with $\bar{Z}^*_i(\bar{Z}^*_j)^T$ measuring the similarity of nodes i and j (i.e. the likelihood of the link (i, j)). Low AP score then indicates that the edges in $\mathcal{T}$ are less likely (non-edges more likely resp.). Finally, we pick the top f flips with lowest AP scores and use them to poison the network.



(a) Cora: DeepWalk's loss

(b) Cora: Classification

(c) PolBlogs: Classification

Figure 1: Vulnerability of the embeddings under the general attack for increasing number of flips. Positive (resp. negative) numbers on the x-axis indicate adding (resp. removing) edges. The percentage of flips is w.r.t. the total number of edges in the clean graph. The dotted line shows the performance on the clean graph before attacking.

4. Experimental Evaluation

Since this is the first work considering adversarial attacks on node embeddings there are no known baselines. Similar to methods that optimize the graph structure (Chen et al., 2016; Khalil et al., 2014) we compare with several strong baselines. $\mathcal{B}_{rnd}$ randomly flips edges (we report averages over ten seeds), $\mathcal{B}_{eig}$ removes edges based on their eigencentrality in the line graph $L(A)$, and $\mathcal{B}_{deg}$ removes edges based on their degree centrality in $L(A)$ (equivalently sum of degrees in the graph). When adding edges we use the same baselines as above now calculated on the complement graph except for $\mathcal{B}_{eig}$ since it is infeasible to compute even for medium size graphs. $\mathcal{A}_{DW_2}$ denotes our gradient based attack, $\mathcal{A}_{DW_3}$ our closed-form attack, $\mathcal{A}_{link}$ our targeted link prediction attack, and $\mathcal{A}_{class}$ is our targeted node classification attack.

The size of the sampled candidate set for adding edges under the general attack is 20K (we report averages over five trials). To evaluate the targeted link prediction attack we form the target pairs $\mathcal{T}$ by randomly sampling 10% of the edges from the clean graph and three times as many non-edges. This setup reflects the fact that in practice the attackers often care more about increasing the likelihood of a new edge, e.g. increasing the chance of recommending an item to a user.

We aim to answer the following questions: (Q1) how good are our approximations of the loss; (Q2) how much damage is caused to the overall embedding quality by our attacks compared to the baselines; (Q3) can we still perform a successful attack when the attacker is restricted; (Q4) what characterizes the selected (top) adversarial edges; (Q5) how do the targeted attacks affect downstream tasks; and (Q6) are the selected adversarial edges transferable to other models.

We set DeepWalk's hyperparameters to: $T = 5, b = 5, K = 64$ and use logistic regression for classification. We analyze three datasets: Cora ($N = 2810, |E| = 15962$, McCallum et al. (2000); Bojchevski & Günnemann (2018)) and Citeseer ($N = 2110, |E| = 7336$, Giles et al. (1998)) are citation networks commonly used to benchmark embedding approaches, and PolBlogs ($N = 1222, |E| = 33428$, Adamic & Glance (2005)) is a graph of political blogs. Since we are in the poisoning setting, in all experiments after choosing the top f flips we re-train the standard embeddings produced by DeepWalk and report the final performance. Note, for the *general attack* the downstream node classification performance is *only a proxy* for estimating the embedding quality after the attack, it is not our goal to damage this task, but rather to attack the unsupervised embeddings in general.

4.1. Approximation Quality

To estimate the approximation quality we randomly select 20K candidates from the Cora graph and we compute Pearson's R score between the actual loss (including the elementwise logarithm) and our approximations. For example, for dimensionality $K = 32$ we have $R(\mathcal{L}_{DW_2}, \mathcal{L}_{DW_1}) = 0.11$ and $R(\mathcal{L}_{DW_3}, \mathcal{L}_{DW_1}) = 0.90$ showing that our closed-form strategy approximates the loss significantly better than the gradient-based one. This also holds for $K = 64, 128$. Moreover, we randomly select 5K candidates and we compare the true eigenvalues λ' after performing a flip (i.e. doing a full eigen-decomposition) and our approximation $\tilde{\lambda}'$. We found that the difference $|\lambda' - \tilde{\lambda}'|$ is negligible: several orders of magnitude smaller than the eigenvalues themselves. The difference between the terms $|\sum_{r=1}^{T} \lambda_i'^r - \sum_{r=1}^{T} \tilde{\lambda}_i'^r|$ used in Lemma 3 is similarly negligible. Additionally, we compare the true singular values $\sigma_i(S)$ of the matrix S and their respective upper bounds $d_{min}^{-1} |\sum_{r=1}^{T} \lambda_i^r| \geq \sigma_i(S)$ obtained from Lemma 3. The gap is different across graphs and it is relatively small overall. We plot all these quantities for all graphs in the appendix (Sec. 6.3). These results together demonstrate the quality of our approximation.

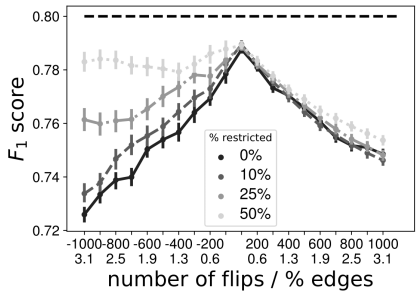
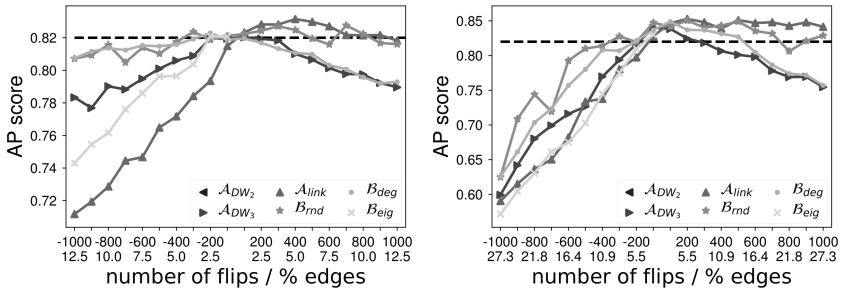


Figure 2: Cora: Classification performance for increasingly restricted attacks.



(a) Cora (b) Citeseer

Figure 3: Targeted attack on the link prediction task.

4.2. General Attack

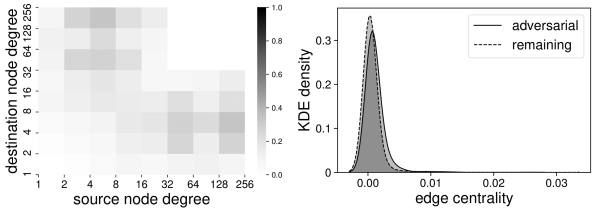
To better understand the attacks we investigate the effect of removing and adding edges separately. We select the top f edges from the respective candidate sets according to our approximation of the loss function. For adding edges, we also implemented an alternative add-by-remove strategy denoted as $\mathcal{A}_{abr}$. Here, we first add cf-many edges randomly sampled from the candidate set to the graph and subsequently remove $(c-1)f$-many of them (equals to only f changes in total). This strategy performed better empirically. Since the graph is undirected, for each (i, j) we also flip (j, i).

Fig. 1 answers question (Q2). Removed/added edges are denoted on the x-axis with negative/positive values respectively. On Fig. 1a we see that our strategies achieve a significantly higher loss compared to the baselines when removing edges. To analyze the change in the embedding quality we consider the node classification task (i.e. using it as a proxy to evaluate quality; this is *not* our targeted attack). Interestingly, $\mathcal{B}_{deg}$ is the strongest baseline w.r.t. to the loss, but this is not true for the downstream task. As shown in Fig. 1b and 1c, our strategies significantly outperform the baselines. As expected, $\mathcal{A}_{DW_3}$ and $\mathcal{A}_{abr}$ perform better than $\mathcal{A}_{DW_2}$. On Cora our attack can cause up to around 5% *more* damage compared to the strongest baseline. On PolBlogs, by adding only 6% edges we can decrease the classification performance by more than 23%, while being more robust to removing edges.

Restricted attacks. In the real world attackers cannot attack any node, but rather only specific nodes under their control, which translates to restricting the candidate set. To evaluate the restricted scenario, we first initialize the candidate sets as before, then we randomly denote a given percentage p_r of nodes as restricted and discard every candidate that includes them. As expected, the results in Fig. 2 show that for increasingly restrictive sets with $p_r = 10\%, 25\%, 50\%$, our attack is able to do less damage. However, we always outperform the baselines (not plotted), and even in the case when half of the nodes are restricted ($p_r = 50\%$) we are still able to damage the embeddings. With this we can answer question (Q3): attacks are successful even when restricted.

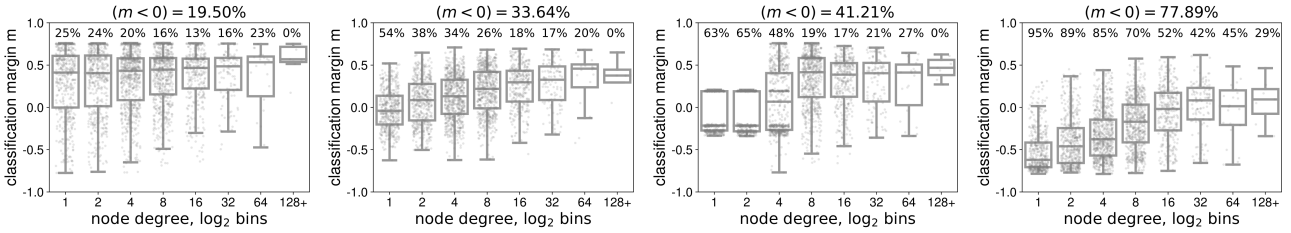
Analysis of the selected adversarial edges. A natural question to ask is what characterizes the adversarial edges that are selected by our attack, and whether its effectiveness can be explained by a simple heuristic such as attacking "important" edges (e.g. edges that have high centrality). To answer this question we analyze the top 1000 edges selected by $\mathcal{A}_{DW_3}$ on the Cora dataset. In Fig. 4a we analyze the adversarial edges in terms of node degrees. Specifically, for each edge we consider the degree of its source node and its destination node and plot it on the x-axis and y-axis respectively. The heatmap shows the number of adversarial edges divided by total number of edges for each degree (binned logarithmically). We see that low, medium and high degree nodes are all represented and therefore we conclude that we cannot distinguish between adversarial and non-adversarial edges based solely on their degrees.

In Fig. 4b we plot the edge centrality distribution for the top 1000 adversarial edges and compare it with the edge centrality distribution of the remaining edges. We can see that there is no clear distinction. Both of these findings highlight the need for a principled method such as ours since using intuitive heuristics such as degree centrality or edge centrality cannot identify adversarial edges.



(a) Degree centrality (b) Edge centrality

Figure 4: Analysis of the adversarial edges selected by $\mathcal{A}_{DW_3}$ on the Cora graph w.r.t. different centrality measures.



(a) Before attack (b) Baseline $\mathcal{B}_{rnd}$ attack (c) Baseline $\mathcal{B}_{deg}$ attack (d) Our $\mathcal{A}_{class}$ attack

Figure 5: Margins for the clean and corrupted graphs for different attacks. Each dot represent one node binned logarithmically according to its degree. The number above each (box-) plot indicates the misclassification rate ($m < 0$). Lower is better.

4.3. Targeted Attack

To obtain a better understanding of the performance we study the margin $m(t)$ (Sec. 3.4) on Cora before and after the attack considering every node t as a potential target. We allow a budget of only (d_t+3) flips per each node (where d_t is the degree of the target node t) ensuring that the degrees do not change noticeably after the attack. Each dot in Fig. 5 represents one node grouped by its degree in the clean graph (logarithmic bins). We see that low-degree nodes are easier to misclassify ($m(t) < 0$), and that high degree nodes are more robust in general – the baselines have 0% success. Our method, however, can successfully attack even high degree nodes. In general, our attack is significantly more effective across all bins – as shown by the numbers on top of each box – with 77.89% nodes successfully misclassified on average compared to e.g. only 33.64% for $\mathcal{B}_{rnd}$.

For the link prediction task (Fig. 3) we are similarly able to cause significant damage – e.g. $\mathcal{A}_{link}$ achieves almost 10% decrease in performance by flipping around 12.5% of edges on Cora, significantly better than all other baselines. Here again, compared to adding edges, removing has a stronger effect. Overall, answering (Q5), both experiments confirm that our attacks hinder the various downstream tasks.

4.4. Transferability

The question of whether attacks learned for one model generalize to other models is important since in practice the attacker might not know the model used by the defender. However, if transferability holds, such knowledge is not required. To obtain the perturbed graph, we remove the top f adversarial edges with $\mathcal{A}_{DW_3}$. The same perturbed graph is used to learn embeddings using several other state-of-the-art approaches: DeepWalk (DW) with both the SVD and the SGNS loss, node2vec (Grover & Leskovec, 2016), Spectral Embedding (von Luxburg, 2007), Label Propagation (Zhu & Ghahramani, 2002), and GCN (Kipf & Welling, 2017).

Table 1 shows the change in node classification performance compared to the embeddings learned on the clean graph for each method respectively. Answering (Q6), the results show that our attack generalizes: the adversarial edges have a noticeable impact on other models as well. We see that we can damage DeepWalk trained with the skip-gram objective with negative sampling (SGNS) showing that our factorization analysis via SVD is successful. We can even damage the performance of semi-supervised approaches such as Graph Convolutional Networks (GCN) and Label Propagation.

Table 1: Transferability: The change in F_1 score (in percent) compared to the clean/original graph. Lower is better.

	Method	DW SVD	DW SGNS	node-2vec	Spect. Embd	Label Prop.	GCN
Our Approach	Cora						
	$f = 250 (03.1\%)$	-3.59	-3.97	-2.04	-2.11	-5.78	-3.34
	$f = 500 (06.3\%)$	-5.22	-4.71	-3.48	-4.57	-8.95	-2.33
	Citeseer						
	$f = 250 (06.8\%)$	-7.59	-5.73	-6.45	-3.58	-4.99	-2.21
	$f = 500 (13.6\%)$	-9.68	-11.47	-10.24	-4.57	-6.27	-8.61
$\mathcal{B}_{eig}$ Baseline	Cora						
	$f = 250 (03.1\%)$	-0.61	-0.65	-0.57	-0.86	-1.23	-6.33
	$f = 500 (06.3\%)$	-0.71	-1.22	-0.64	-0.51	-2.69	-0.64
	Citeseer						
	$f = 250 (06.8\%)$	-0.40	-1.16	-0.26	+0.11	-1.08	-0.70
	$f = 500 (13.6\%)$	-2.15	-2.33	-1.01	+0.38	-3.15	-1.40

Compared to the transferability of the strongest baseline $\mathcal{B}_{eig}$, shown in the lower section of Table 1, we can clearly see that our attack causes significantly more damage.

5. Conclusion

We demonstrated that node embeddings are vulnerable to adversarial attacks which can be efficiently computed and have a significant negative effect on downstream tasks such as node classification and link prediction. Furthermore, successfully poisoning the graph is possible with relatively small perturbations and under restriction. More importantly, our attacks generalize - the adversarial edges are transferable across different models. Developing effective defenses against adversarial attacks as well as more comprehensive modelling of the attacker's knowledge are important directions for improving network representation learning.

Acknowledgments

This research was supported by the German Research Foundation, Emmy Noether grant GU 1409/2-1, and the German Federal Ministry of Education and Research (BMBF), grant no. 01IS18036B. The authors of this work take full responsibilities for its content.

References

Adamic, L. A. and Glance, N. S. The political blogosphere and the 2004 U.S. election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery, LinkKDD*, 2005.

Amelkin, V. and Singh, A. K. Disabling external influence in social networks via edge recommendation. *CoRR*, abs/1709.08139, 2017.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *International Conference on Machine Learning, ICML*, 2012.

Biggio, B., Fumera, G., and Roli, F. Security evaluation of pattern classifiers under attack. *CoRR*, abs/1709.00609, 2017.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations, ICLR*, 2018.

Bojchevski, A., Matkovic, Y., and Günnemann, S. Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings. In *Conference on Knowledge Discovery and Data Mining, SIGKDD*, 2017.

Cai, H., Zheng, V. W., and Chang, K. C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9), 2018.

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

Chan, H., Akoglu, L., and Tong, H. Make it or break it: Manipulating robustness in large networks. In *International Conference on Data Mining, SIAM*, 2014.

Chaoji, V., Ranu, S., Rastogi, R., and Bhatt, R. Recommendations to boost content spread in social networks. In *World Wide Web Conference, WWW*, 2012.

Chen, C., Tong, H., Prakash, B. A., Eliassi-Rad, T., Faloutsos, M., and Faloutsos, C. Eigen-optimization on large graphs by edge manipulation. *TKDD*, 10(4), 2016.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017a.

Chen, Y., Nadji, Y., Kountouras, A., Monrose, F., Perdisci, R., Antonakakis, M., and Vasiloglou, N. Practical attacks against graph-based clustering. In *Proceedings of ACM SIGSAC CCS*, 2017b.

Cissé, M., Adi, Y., Neverova, N., and Keshet, J. Houdini: Fooling deep structured prediction models. *CoRR*, abs/1707.05373, 2017.

Csáji, B. C., Jungers, R. M., and Blondel, V. D. Pagerank optimization by edge selection. *Discrete Applied Mathematics*, 169:73–87, 2014.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. Adversarial attack on graph structured data. In *International Conference on Machine Learning, ICML*, 2018a.

Dai, Q., Li, Q., Tang, J., and Wang, D. Adversarial network embedding. In *Conference on Artificial Intelligence, AAAI*, 2018b.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems, NIPS*, 2016.

Giles, C. L., Bollacker, K. D., and Lawrence, S. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*. ACM, 1998.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations, ICLR*, 2015.

Grosse, K., Papernot, N., Manoharan, P., Backes, M., and McDaniel, P. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pp. 62–79. Springer, 2017.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Conference on Knowledge Discovery and Data Mining, SIGKDD*, 2016.

Gyöngyi, Z. and Garcia-Molina, H. Link spam alliances. In *International Conference on Very Large Data Bases, VLDB*, 2005.

Israeli, E. and Wood, R. K. Shortest-path network interdiction. *Networks*, 40(2), 2002.

Khalil, E. B., Dilkina, B. N., and Song, L. Scalable diffusion-aware optimization of network topology. In *Conference on Knowledge Discovery and Data Mining, SIGKDD*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, ICLR*, 2017.

Klicpera, J., Bojchevski, A., and Gnnemann, S. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations, ICLR*, 2019.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning, ICML*, 2017.

Li, B., Wang, Y., Singh, A., and Vorobeychik, Y. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in Neural Information Processing Systems, NIPS*, 2016.

Liang, B., Li, H., Su, M., Bian, P., Li, X., and Shi, W. Deep text classification can be fooled. In *International Joint Conference on Artificial Intelligence IJCAI*, 2018.

Lin, Y., Hong, Z., Liao, Y., Shih, M., Liu, M., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. In *International Joint Conference on Artificial Intelligence IJCAI*, 2017.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2), 2000.

Mei, S. and Zhu, X. Using machine teaching to identify optimal training-set attacks on machine learners. In *Conference on Artificial Intelligence, AAAI*, 2015.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *ICLR 2013, Workshop Track Proceedings*, 2013.

Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. In *AISec@CCS*, 2017.

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P*, 2016.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: online learning of social representations. In *Conference on Knowledge Discovery and Data Mining, SIGKDD*, 2014.

Phillips, C. A. The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, 1993.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *International Conference on Web Search and Data Mining WSDM*, 2018.

Stewart, G. W. Matrix perturbation theory. 1990.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations, ICLR*, 2014.

von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Yang, C. and Liu, Z. Comprehend deepwalk as matrix factorization. *CoRR*, abs/1501.00358, 2015.

Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. Network representation learning with rich text information. In *International Joint Conference on Artificial Intelligence IJCAI*, 2015.

Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM*, 2006.

Zhao, M., An, B., Yu, Y., Liu, S., and Pan, S. J. Data poisoning attacks on multi-task relationship learning. In *Conference on Artificial Intelligence, AAAI*, 2018.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. 2002.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations, ICLR*, 2019.

Zügner, D. and Günnemann, S. Certifiable robustness and robust training for graph convolutional networks. In *Conference on Knowledge Discovery and Data Mining, SIGKDD*, 2019.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *Conference on Knowledge Discovery and Data Mining, SIGKDD*, 2018.