Applied Network Science

RESEARCH Open Access

Complex network effects on the robustness of graph convolutional networks

Benjamin A. Miller[1*], Kevin Chan[2] and Tina Eliassi-Rad[1]

*Correspondence:
miller.be@northeastern.edu

[1] Northeastern University, Boston, MA, USA
[2] US Army Research Laboratory, Adelphi, MD, USA

Abstract

Vertex classification using graph convolutional networks is susceptible to targeted poisoning attacks, in which both graph structure and node attributes can be changed in an attempt to misclassify a target node. This vulnerability decreases users' confidence in the learning method and can prevent adoption in high-stakes contexts. Defenses have been proposed, focused on filtering edges before creating the model or aggregating information from neighbors more robustly. This paper considers an alternative: we investigate the ability to exploit network phenomena in the training data selection process to improve classifier robustness. We propose two alternative methods of selecting training data: (1) to select the highest-degree nodes and (2) to select nodes with many connections to the test data. In four real datasets, we show that changing the training set often results in far more perturbations required for a successful attack on the graph structure; often a factor of 2 over the random training baseline. We also run a simulation study in which we demonstrate conditions under which the proposed methods outperform random selection, finding that they improve performance most when homophily is higher, clustering coefficient is higher, node degrees are more homogeneous, and attributes are less informative. In addition, we show that the methods are effective when applied to adaptive attacks, alleviating concerns about generalizability.

Keywords: Graph convolutional networks, Poisoning attacks, Robust machine learning

Introduction

Classification of vertices in graphs is an important problem in a variety of applications, from e-commerce (classifying users for targeted advertising) to security (classifying computer nodes as malicious or not) to bioinformatics (classifying roles in a protein interaction network). In the past several years, numerous methods have been developed for this task (see, e.g., Hamilton et al. (2017) and Moore and Neville (2017)). More recently, research has focused on attacks by adversaries (Zügner et al. 2018; Dai et al. 2018) and robustness to such attacks (Wu et al. 2019b). If an adversary were able to insert misleading data into the training set (e.g., generate benign traffic during a data collection period that could conceal its behavior during testing/inference time), the chance of successfully

evading detection would increase, leaving data analysts unable to respond to potential threats.

To classify vertices in the presence of adversarial activity, we must implement learning systems that are robust to such potential manipulation. If such malicious behavior has low cost to the attacker and imposes high cost on the data analyst, machine learning systems will not be trusted and adopted for use in practice, especially in high-stakes scenarios such as network security and traffic safety. Understanding how to achieve robustness is key to realizing the full potential of machine learning.

Adversaries, of course, will attempt to conceal their manipulation. The first published poisoning attack against vertex classification was an adversarial technique called Nettack (Zügner et al. 2018), which can create perturbations that are subtle while still being extremely effective in decreasing performance on the target vertices. The authors use their poisoning attack against a graph convolutional network (GCN).

From a defender's perspective, we aim to make it more difficult for the attacker to cause node misclassification. In addition to changing the properties of the classifier itself, there may be portions of a complex network that provide more information for learning than others. Complex networks are highly heterogeneous and random sampling may not be the best way to obtain labels. If there is flexibility in the means of obtaining training data, the defender should leverage what is known about the graph topology.

This paper investigates the hypothesis that leveraging network properties can improve robustness of GCNs in the presence of adversaries. In particular, we demonstrate that increasing the number of labeled nodes in the unlabeled nodes' neighborhoods does, in some circumstances, increase the number of required perturbations for an attack to be successful. The increase in robustness depends on aspects of the graph topology, such as homophily, clustering coefficient, and degree distribution, as well as information provided by features. This phenomenon can be exploited when selecting training data when conditions are favorable. We focus on two alternative techniques for training data selection. Both methods aim to train with a subset of nodes that are well connected to the held out set. Here we see a benefit, often raising the number of perturbations required for a given level of attack success by over a factor of 2. When it is possible to pick a specific subset on which to train, this can provide a significant advantage.

Scope and contributions

In this paper, we are specifically interested in *targeted poisoning* attacks against vertex classifiers, where the data are modified at training time to cause a specific target node to be misclassified. We consider attacks against the structure of the graph, rather than against node attributes. We focus on classification methods where there is an implicit assumption of homophily, and where the topology does not vary over time. Working within this context, the contributions of this work are as follows:

- We propose two methods—StratDegree and GreedyCover—for selecting training data that are well-connected to the unlabeled vertices.
- We demonstrate that StratDegree and GreedyCover often result in a greater burden on attackers that cannot be reliably obtained by simply increasing the amount of randomly selected training data.

- We show that the most robust defenses—typically those based on singular value decomposition of the adjacency matrix (Entezari et al. 2020) and Jaccard coefficient of attributes (Wu et al. 2019b)—are often improved by working in conjunction with GreedyCover.
- We show that there is no consistent tradeoff between the robustness gained from StratDegree and GreedyCover and classification performance.
- In simulation, we study the effects of various generative models and report the impact of class homophily, topological features, and node attribute similarity across classes on classification performance and robustness to attack.
- We apply StratDegree and GreedyCover to an adaptive attack, demonstrating that the proposed training methods improve robustness in a more general setting.

Paper organization

The remainder of this paper is organized as follows. In "Related work", we briefly contextualize our work within the current literature. In "Problem definition and proposed methods" we describe the vertex classification problem, GCNs, and the Nettack method, and outlines the techniques we investigate to select training data. The "Experiments and results" section details the experimental setup—including datasets, attacks, and classification methods—and results. Experimental results are included for real data, illustrating the effectiveness of the proposed methods in several cases, and the results of a simulation study in which we vary graph topology, node attributes, and homophily, and evaluate robustness of the methods across the landscape. In "Conclusions" we conclude with a summary and outline open problems and future work.

Related work

Adversarial examples in deep neural networks have received considerable attention since they were documented several years ago (Szegedy et al. 2014). Since that time, numerous attack methods have been proposed, largely focused on the image classification domain (though there has been interest in natural language processing as well, e.g., from Jia and Liang (2017)). In addition to documenting adversarial examples, Szegedy et al. (2014) demonstrated that such examples can be generated using the limited-memory BFGS (L-BFGS) algorithm, which identifies an adversarial example in an incorrect class with minimal L_2 norm distance to the true data. Later, Goodfellow et al. (2015) proposed the fast gradient sign method (FGSM), where the attacker starts with a clean image and takes small, equal-sized steps in each dimension (i.e., alters each pixel by the same amount) in the direction maximizing the loss. Another proposed attack—the Jacobian-based Saliency Map Attack (JSMA)—iteratively modifies the pixel with the largest impact on the loss (Papernot et al. 2016a). DeepFool, like L-BFGS, minimizes the L_2 distance from the true instance while crossing a boundary into an incorrect class, but does so quickly by approximating the classifier as linear, stepping to maximize the loss, then correcting for the true classification surface (Moosavi-Dezfooli et al. 2016). Like Nettack, these methods all try to maintain closeness to the original data (L_2 norm for L-BFGS and DeepFool, L_0 norm for JSMA, and L_∞ norm for FGSM).

Some of these methods have been adapted for use with graph neural networks (GNNs). Wu et al. (2019b) modify FGSM and JSMA to use integrated gradients and show it to be effective against vertex classification. In addition, new attacks against vertex classification have been introduced, including a method that uses reinforcement learning to identify modifications to graph structure for an evasion attack (Dai et al. 2018). To increase the scale of attacks, Li et al. (2021) propose an attack that only considers a k-hop neighborhood of the target. This method attacks a simplified GCN, introduced by Wu et al. (2019a), which applies a logistic regression classifier after k rounds of feature propagation.

Defenses to attacks such as FGSM and JSMA have been proposed, although several prove to be insufficient against stronger attacks. A simple improvement is to include adversarial examples in the training data (Goodfellow et al. 2015). Defensive distillation is one such defense, in which a classifier is trained with high "temperature" in the softmax, which is reduced for classification (Papernot et al. 2016b). While this was effective against the methods from Szegedy et al. (2014), Goodfellow et al. (2015), Papernot et al. (2016a), and Moosavi-Dezfooli et al. (2016), it was shown by Carlini and Wagner (2017) that modifying the attack by changing the constraint function (which ensures the adversarial example is in a given class) renders this defense ineffective. More defenses have been proposed, such as pixel deflection (Prakash et al. 2018) and randomization techniques (Xie et al. 2018), but many such methods are still found to be vulnerable to attacks (Athalye and Carlini 2018; Athalye et al. 2018). Other work has focused on provably robust defenses (Wong and Kolter 2018), with empirical performance often close to certifiable claims (Croce et al. 2019). Stochastic networks have also shown improved robustness to various attacks (Dapello et al. 2021). In the wake of growing interest in adversarial robustness, Carlini et al. (2019) have aggregated best practices for evaluation of systems.

More recent work has focused on robustness of GCNs, including work on robustness to attacks on attributes (Zügner and Günnemann 2019b) and more robust GCN variants (Zhu et al. 2019). Multiple authors have considered aggregation techniques that are less sensitive to outliers (Geisler et al. 2020; Chen et al. 2021). One approach to a more robust classifier incorporates an attention mechanism that learns the importance of other nodes' features to a node's class (Veličković et al. 2018). Others have considered using a GCN with modified graph structure to improve robustness, such as using a low-rank approximation for the graph (Entezari et al. 2020) and filtering edges based on attribute values (Wu et al. 2019b). Another method also considers attribute values, in this case creating a similarity graph from the attributes that augments the given graph structure to preserve node similarity in feature space (Jin et al. 2021). The low-rank structure and node similarity concepts are combined by Jin et al. (2020b) to create a neural network that aims to simultaneously learn the true graph structure from poisoned data and learn a classifier of unlabeled nodes. Dai et al. (2022) explore a similar idea in the context of noisy data and few labels, using link prediction to augment the observed graph. Relations between attribute similarity and node class—including possible heterophily—are also exploited in GNNGuard (Zhang and Zitnik 2020). More recent work has shown that attacks that are adaptive to defenses easily undermine the robustness increase observed when using non-adaptive attacks (Mujkanovic et al. 2022). Other

recent GCN developments include modifications to deal with heterophily, via classifier design choices (Zhu et al. 2020) and by learning the level of homophily or heterophily in the graph as part of the training procedure (Zhu et al. 2021). Several attacks (Dai et al. 2018; Zügner et al. 2018; Chen et al. 2018; Zügner and Günnemann 2019a; Wu et al. b; Xu et al. 2019) and defenses (Goodfellow et al. 2015; Wu et al. 2019b; Zhu et al. 2019; Entezari et al. 2020; Jin et al. 2020b) have been incorporated into a software package called DeepRobust, enabling convenient experimentation across a variety of conditions (Li et al. 2020; Jin et al. 2020a). As with neural networks more generally, there has been work on certifiable robustness for GCNs (Bojchevski and Günnemann 2019; Zügner and Günnemann 2020).

While this paper is focused on targeted attacks, several attacks, such as those proposed by Zügner and Günnemann (2019a) and Xu et al. (2019), attack the whole graph in order to degrade overall performance. Some attacks in this area have allowed adding new nodes (Sun et al. 2019), flipping labels (Liu et al. 2019), and rewiring edges (Ma et al. 2019). In addition, there are many machine learning tasks on graphs other than vertex classification, and work has been done on, for example, edge classification in an adversarial context (Yu et al. 2018). Finally, altering a network by removing edges or nodes has been studied well beyond the context of machine learning, focusing on altering shortest paths (Miller et al. 2023) or disconnecting the graph (Albert et al. 2000), where there has been recent work showing the impact of selecting nodes with particular features for removal (Ficara et al. 2023).

Problem definition and proposed methods

Problem definition

We consider the same setting from Zügner et al. (2018). We are given an undirected graph $G = (V, E)$ of size $N = |V|$ and an $N \times d$ matrix of vertex attributes X. Each node has an arbitrary numeric index from 1 to N. We consider only binary attributes. In addition to its d attributes, each node has a label denoting its class. We enumerate classes as integers from 1 to C. Given a subset of labeled instances, the goal is to correctly classify the unlabeled nodes.

We focus on GCNs, which make use of the adjacency matrix for the graph $A = \{a_{ij}\}$, where a_{ij} is 1 if there is an edge between node i and node j and is 0 otherwise. The GCN applies a symmetrized one-hop graph convolution (Kipf and Welling 2017) to the input layer. That is, if we let D be the diagonal matrix of vertex degrees—i.e., the ith diagonal entry is the number of edges connected to vertex i, $d_{ii} = \sum_{j=1}^{N} a_{ij}$—then the output of the first layer of the network is expressed as

$$H = \sigma\left(D^{-1/2}AD^{-1/2}XW_1\right), \tag{1}$$

where W_1 is a weight matrix, X is a feature matrix whose ith row is x_i^T (the attribute vector for row vertex i), and σ is the rectifier function. We use GCNs with a single hidden layer. From the hidden layer to the output layer, a similar graph convolution is performed, followed by a softmax output:

$$Y = \text{softmax}\left(\mathrm{D}^{-1/2}\mathrm{AD}^{-1/2}\mathrm{HW}_2\right),$$

where W_2 is another matrix of learned weights. Each vertex is then classified according to the largest entry in the corresponding row of Y.

Nettack—the vertex attack proposed by Zügner et al. (2018)—operates on a surrogate model where the rectifier function is replaced by a linear function, thus approximating the overall network as

$$\begin{aligned} Y &\approx \text{softmax}\left(\left(\mathrm{D}^{-1/2}\mathrm{AD}^{-1/2}\right)^2 \mathrm{XW}_1\mathrm{W}_2\right) \\ &= \text{softmax}\left(\left(\mathrm{D}^{-1/2}\mathrm{AD}^{-1/2}\right)^2 \mathrm{XW}\right). \end{aligned} \tag{2}$$

Nettack uses a greedy algorithm to determine how to perturb both A and X to make the GCN misclassify a target node. The changes are intended to be "unnoticeable," i.e., the degree distribution of G and the co-occurrence of features are changed negligibly. Using the approximation in (2), Nettack perturbs by either adding or removing edges or turning off binary features so that the classification margin is reduced the most at each step. Note that while it can change the topology and the features, Nettack does *not* change the labels of any vertices. In this paper, we only consider structural perturbations. Nettack allows either *direct* attacks, in which the target node itself has its edges and features changed, or indirect *influence* attacks, where neighbors of the target have their data altered.

The classifier is evaluated in a context where only some of the labels are known, and the labeled data are split into training and validation sets. To train the GCN, 10% of the data are selected at random (or by one of the alternative methods outlined in "Proposed training data selection methods"), and another 10% is selected for validation. The remaining 80% is the test data. After training, nodes are selected for attack among those that are correctly classified. The goal of the defender is to make the a successful attack as expensive as possible.

As we discuss in "Experimental setup", we also consider attacks other than Nettack, and classifiers other than standard GCNs. While the details differ (e.g., using different criteria to identify perturbations), the overall problem definition remains the same.

Proposed training data selection methods

As we investigated classification performance using Nettack, we noted that nodes in the test set with many neighbors in the training set were more likely to be correctly classified. This dependence on labeled neighbors is consistent with previous observations (Neville et al. 2009). We observed this effect using the standard method of training data selection used in the original Nettack paper: randomly select 10% for training, 10% for validation, and 80% for testing. This observation suggested that a training set where the held-out nodes are well represented among neighborhoods of the training data—providing a kind of "scaffolding" for the unlabeled data—could make the classification more robust.

We considered two methods to test this hypothesis. The first simply chooses the highest-degree nodes (stratified by class) to be in the training set. We refer to the stratified degree-based thresholding method as StratDegree. The other method uses a greedy approach in an attempt to ensure every node has at least a minimal number of neighbors in the training set. Starting with an empty training set and a threshold $k = 0$, we iteratively add a node of a particular class with the largest number of neighbors that are connected to at most k nodes in the training set. The class is randomly selected based on how many nodes of each class are currently in the training set and the number required to achieve class stratification (see the pseudo-code for details). When there are no such neighbors, we increment k. This procedure continues until we have the desired proportion of the overall dataset for training. Algorithm 1 provides the pseudo-code.

Algorithm 1 GreedyCover

Algorithm 1 GreedyCover

Input: Graph $G = (V, E)$, training proportion $t \in (0, 1)$, classes C, class map $\text{class} : V \to C$
Output: Training set $T \subset V$
$k \leftarrow 0$
for all $u \in V$ **do**
    $m_u \leftarrow 0$ $\quad \langle\langle$number of neighbors in the training set: initialize all nodes to 0$\rangle\rangle$
end for
for all $c \in C$ **do**
    $\langle\langle$initialize vertex and training subsets partitioned by class$\rangle\rangle$
    $V_c \leftarrow \{v \in V | \text{class}(v) = c\}$
    $T_c \leftarrow \emptyset$
end for
while $|T_c| < t|V_c|$ $\forall c \in C$ **do**
    $c \leftarrow$ class selected with probability proportional to $1 - \frac{|T_c|}{t|V_c|}$
    $v \leftarrow \arg\max_{u \in V_c \setminus T_c} \sum_{u' \in \mathcal{N}(u)} \mathbb{I}[m_{u'} = k]$
    if $\sum_{u' \in \mathcal{N}(v)} \mathbb{I}[m_{u'} = k] = 0$ **then**
        $k \leftarrow k + 1$ $\quad \langle\langle$incr. min. num. trained neighbors$\rangle\rangle$
    else
        $T_c \leftarrow T_c \cup \{v\}$
        $T \leftarrow \bigcup_{c \in C} T_c$
        $m_v \leftarrow -1$
        for all $u' \in \mathcal{N}(v) \setminus T$ **do**
            $m_{u'} \leftarrow m_{u'} + 1$
        end for
    end if
end while
return T

Using StratDegree and GreedyCover has computational costs beyond random sampling. StratDegree requires finding the highest-degree nodes, which, for a constant fraction of the dataset size, will require $O(|E| + |V| \log |V|)$ time (for computing degrees and sorting), compared to $O(|V|)$ time for random sampling. Each step in GreedyCover requires finding the vertex with the most neighbors minimally connected to the training set. As written in Algorithm 1, each iteration requires $O(|E|)$ time

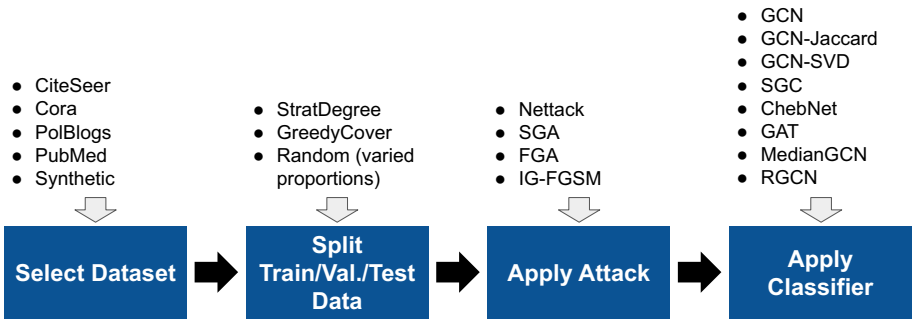


Fig. 1 Processing chain for experiments. Each experiment takes a dataset, applies a method to split training, validation, and test data, applies an attack to a set of target nodes, then applies a classifier to the attacked dataset. We evaluate the robustness of vertex classification—-in terms of required attacker budget at a given attack success rate—across all possible combinations of dataset, selection methods, attacks, and classifier

to count the number of such neighbors each node has, which would result in an overall running time of $O(|V||E|)$. This could be improved using a priority queue—such as a Fibonacci heap—to achieve $O(|E| + |V| \log |V|)$ time ($O(|V|)$ logarithmic-time extractions of the minimum and $O(|E|)$ constant-time key updates). Thus, the two proposed method require moderate overhead compared to the running time for the GCN.

Experiments and results

Experimental setup

Each experiment in our study involves (1) a graph dataset, (2) a method for selecting training data, (3) a structure-based attack against vertex classification, and (4) a classification algorithm. We consider several options for each step in this process, as shown in Fig. 1. This section details the methods and datasets we use across the experiments in this paper. We use the DeepRobust library (Li et al. 2020) for datasets, attacks, and classifiers.

Datasets

We use the three datasets used in the Nettack paper in our experiments, plus one larger citation dataset:

- *CiteSeer* The CiteSeer dataset has 3312 scientific publications put into 6 classes. The network has 4732 links representing citations between the publications. The features of the nodes contain ones and zeros indicating the presence of the word in the paper. There are 3703 unique words considered for the dictionary.
- *Cora* The Cora dataset consists of 2708 machine learning papers classified into one of seven categories. The citation network consists of 5429 citations. For each paper (vertex) in the network there is a feature vector of zeros and ones for whether it contains one of 1433 unique words.
- *PolBlogs* The political blogs dataset consists of 1490 blogs labeled as either liberal or conservative. A total of 19,025 links between blogs form the directed edges of the graph. No attributes are used.

- *PubMed* The PubMed dataset consists of 19,717 papers pertaining to diabetes classified into one of three classes. The citation network consists of 44,338 citations. For each paper in the network there is a binary feature vector representing the presence of 500 words.

We further evaluate performance using synthetic data. Synthetic network generation to evaluate network effects on the performance of GNNs has recently received attention in the research community (Palowitch et al. 2022). In this work, we consider synthetic datasets that vary four key network features: degree distribution, level of clustering, homophily with respect to labels, and information gained via node attributes. We use four random graph models that exhibit different properties in terms of clustering and degree distribution. In each case, we use 1200 nodes and an average degree of approximately 10.

- *Erdős–Rényi (ER) Graphs*: Each pair of nodes shares an edge with probability 1/120. This model yields homogeneous degree distributions and very little clustering.
- *Barabási–Albert (BA) Graphs*: Each node enters the graph and connects 5 edges to existing nodes with probability proportional to their degrees. The process is initialized with a 6-node star. This model yields graphs with heterogeneous degree distributions and very little clustering.
- *Watts–Strogatz (WS) Graphs*: A ring lattice—where each node is connected to 5 nodes on either side—has 10% of its edges randomly rewired. This model yields graphs with substantial clustering and homogeneous degree distributions.
- *Lancichinetti–Fortunato–Radicchi (LFR) Graphs*: Generates a degree sequence with degree distribution $p(d) \propto d^{-3}$, with average and minimum degree set to $d_{\text{avg}} = 10$ and $d_{\max} = 135$. Nodes are randomly assigned to communities, whose sizes are distributed according to $p(|C|) \propto |C|^{-2}$, with the minimum community size being 10. Nodes create 80% of their connections within the community and 20% outside the community.

If the generated graph has multiple connected components, we use the largest connected component for the experiment.

We also vary the homophily of the vertex labels, from no homophily to highly homophilous, and assign vertex attributes with varying levels of predictive power, from completely uninformative about the vertex's label to highly informative. Details on the methods used to generate synthetic labels and attributes are provided in Appendix A.

Training data selection

To select training data, we use StratDegree and GreedyCover as described in "Proposed training data selection methods", as well as random selection. For StratDegree and GreedyCover, we use the proposed algorithms to select 10% of the data, stratified by class. The remaining 90% of the data is randomly split (stratified by class) into validation (10%) and training data (80%). For random selection, we also want to determine whether adding more random training data improves classification robustness. Thus, in addition to using stratified random sampling to select 10% of the data for training, we

consider larger training sets, increasing to 30% in 5% increments. In all cases, 10% of the data are used for validation and the remainder comprise the test set. We measure the average number of neighbors connected to a node outside of the training set, i.e., for the training set $T \subset V$, we record

$$\frac{1}{|V \setminus T|} \sum_{i \in T} \sum_{j \in V \setminus T} a_{ij}. \tag{3}$$

This allows us to evaluate what impact the overall number of connections to the training data has on performance, and whether performance with the proposed training data selection methods match any trend observed with random training.

Attacks

We use the following attacks, which are implemented in DeepRobust:

- *Nettack* The method from Zügner et al. (2018), briefly described in "Problem definition".
- *Fast Gradient Attack (FGA)* Computes the gradient of the loss function at the target node with respect to the adjacency matrix, then perturb the entry with the largest gradient that points in the correct direction (Chen et al. 2018).
- *Integrated Gradient Attack (IG-Attack)* A similar method that integrates the gradient as an entry in the adjacency matrix varies from 1 to 0 (for edge removal) or 0 to 1 (for edge addition) (Wu et al. 2019b).
- *Simplified Gradient Attack (SGA)* In this case, gradients are computed that only consider a k-hop subgraph around the target (Li et al. 2021).

For direct attacks, we use up to 20 edge additions and removals for a target. For influence attacks, we allow up to 50 perturbations.

Classifiers

We consider the following eight classifier models, some of which were developed with the explicit intent of improving robustness to adversarial attack:

- *GCN* The original GCN architecture as used in Zügner et al. (2018).
- *Jaccard* Before training the GCN, removes edges between nodes that have dissimilar feature vectors before (Wu et al. 2019b).
- *SVD* Uses a GCN in which the adjacency matrix is replaced with a low-rank approximation via truncated singular value decomposition (Entezari et al. 2020).
- *ChebNet* Uses the spectral graph convolutions (Defferrard et al. 2016) of which the convolution operator (1) is a first-order approximation.
- *Simple Graph Convolution (SGC)* Applies a model similar to the surrogate (2), where the matrix W is learned via logistic regression on the features defined by $(D^{-1/2}AD^{-1/2})^k X$ (Wu et al. 2019a).
- *Graph Attention Network (GAT)* Includes an attention mechanism based on the importance of each node's neighbors' features (Veličković et al. 2018).

- *Robust Graph Convolutional Network (RGCN)* Uses Gaussian convolutions, in which the output is drawn from a Gaussian distribution whose parameters the output of a neural network (Zhu et al. 2019).
- *MedianGCN* Aggregates neighbors' features based on their median values rather than weighted averages (Chen et al. 2021).

Training

We tuned classifier hyperparameters for each (classifier, attack, training selection method) triple, first performing a coarse grid search over all hyperparameters, then performing some refinements: altering each single parameter 10% and choosing the configuration with the best performance. The performance metric is a linear combination of the F_1 score (macro averaged) before an attack takes place with the was the average margin of 10 randomly selected targets after 5 perturbations with a direct attack. The resulting hyperparameters were used in all cases with the corresponding classifier, attack, and training selection method.

Evaluation

We evaluate performance based on 25 target nodes. The targets are randomly selected from the set of nodes that are correctly classified when no attack takes place. This procedure is repeated five times with the train/validation/test splits recomputed each time. Our robustness metric is the adversary's required budget to achieve a given attack success rate. We compute this based on the number of perturbations required to give a target a negative classification margin in its correct class. If the target is never successfully misclassified, we set the required budget to the maximum number of perturbations. The result is averaged across the five trials.

Computing platform

All experiments are run on a Linux cluster where each machine has 32 cores and 192 GB of memory. Each process is allocated 2 cores and 20 GB. If an attack or defense experiment—considering all targets with a particular attack or defense and a particular training data selection method—does not complete in 24 h per trial, the result is not recorded.

Results

Real data

We first consider influence attacks, where the target node's neighbors are modified rather than the target itself. We apply both Nettack and FGA, replacing Nettack with SGA if the SGC-based classifier is used. We only obtained results using IG-FGSM on the PolBlogs dataset, which can be seen in Appendix B.1. (IG-FGSM did not substantially outperform the other methods for the best-performing classifiers.) In all other cases, IG-FGSM did not finish in the allotted time (24 h per trial). Results are illustrated in Fig. 2. In addition to the results for standard GCNs, we plot the upper envelope for each method: at a given attack success probability, the largest required

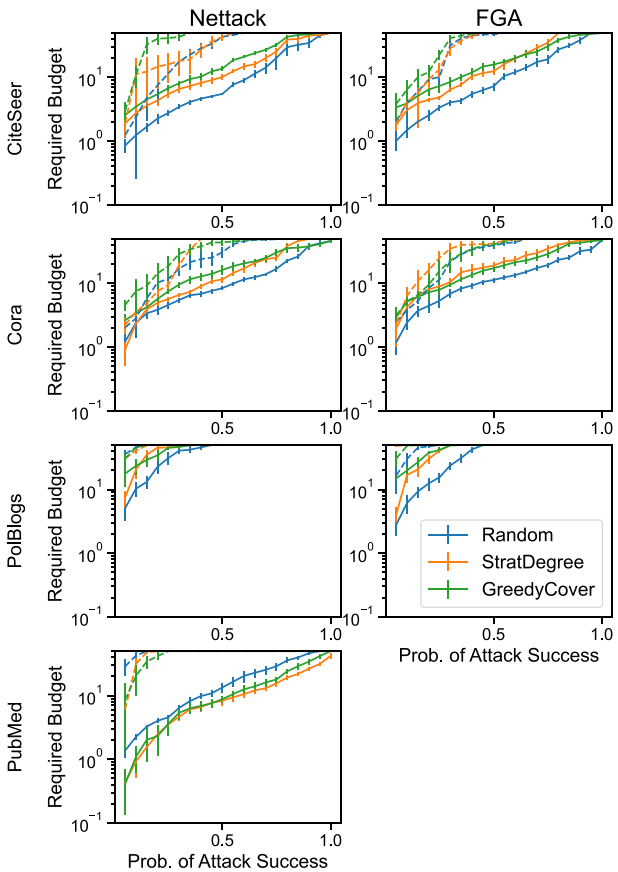


Fig. 2 Robustness to influence attacks using GCNs (solid line) or with the best defense at a given attack success probability (dash line). Results are shown for the CiteSeer, Cora, PolBlogs, and PubMed datasets, each plotted in a subsequent row, and using both the Nettack/SGA (left column) and FGA (right column) attacks. Results were not returned in the allotted time (24 h per trial) for IG-FGSM on all datasets, and FGA for PubMed. Each curve represents the average required budget over 25 randomly selected targets, and error bars are standard errors. Higher is better for the defender. With the exception of the PubMed dataset, GreedyCover performs at least as well as random training selection, and often performs much better

budget across all classifiers. See Appendix B.1 for details about the performance of each individual classifier with each training scheme. CiteSeer has a particularly large increase in the attacker's required budget when using GreedyCover: more than doubling it over several rates of attack success. In fact, at low attack success probabilities, GreedyCover with a GCN provides similar robustness to any of the classifiers listed in the "Classifiers" section with random selection. In addition, GreedyCover provides greater robustness when used in conjunction with the most robust defenses, as shown by the upper envelope. There is a somewhat milder effect on the Cora dataset. In this case, GreedyCover still performs best when using Nettack, but the best performance when attacked with FGA comes from StratDegree (though GreedyCover is within one standard error). With PolBlogs, we also see a benefit from both methods, though we start from a much higher baseline in terms of required perturbations. We see an exception with PubMed, where random training performs best.

Looking deeper into the data, we see that the target nodes for random data tend to have higher margins on the best-performing classifiers. In all other cases, GreedyCover performs as well or better than the other training set selection methods.

Observation 4.1 Training with GreedyCover frequently outperforms other training methods, both with GCNs and in conjunction with published defenses.

For direct attacks—where edges adjacent to the target can be added or removed—we find that there is less improvement in robustness using the alternative methods than indirect attacks. Ensuring that nodes have many neighbors in the training set appears more effective when the neighbors are the nodes whose edges are perturbed. This may be because direct attacks are able to remove connections from the target vertex to the labeled neighbors. Detailed results on direct attacks are available in Appendix B.2.

Observation 4.2 Direct attacks typically benefit less than influence attacks from the alternative training methods.

One additional possibility we considered is that robustness from the alternative training methods comes entirely from the average number of trained neighbors for nodes in the test set. To test this possibility, we performed the same experiments with more randomly selected training data, as described in "Training data selection". Results of these experiments—which show no consistent improvement in robustness by increasing the total number of labeled nodes—are provided in Appendix B.3 and lead to the following observation.

Observation 4.3 Using more training data with random selection does not consistently lead to higher robustness.

Another important consideration is whether increased robustness comes at the expense of classification performance. In Appendix B.4, we provide classification results on all 4 datasets using all defenses and training data selection methods. While StratDegree often results in lower classification performance than random selection, this is not the case for GreedyCover. This yields another datapoint in favor of GreedyCover: it tends to yield the greatest robustness across datasets, and does not seem to greatly hinder overall classification performance.

Observation 4.4 Using GreedyCover yields no consistent reduction in classification performance compared to random training set selection.

The results on real data show that GreedyCover often provides greater robustness to attack, but they are by no means conclusive. In the next section, we further explore the methods with simulated data to observe performance differences while controlling network properties.

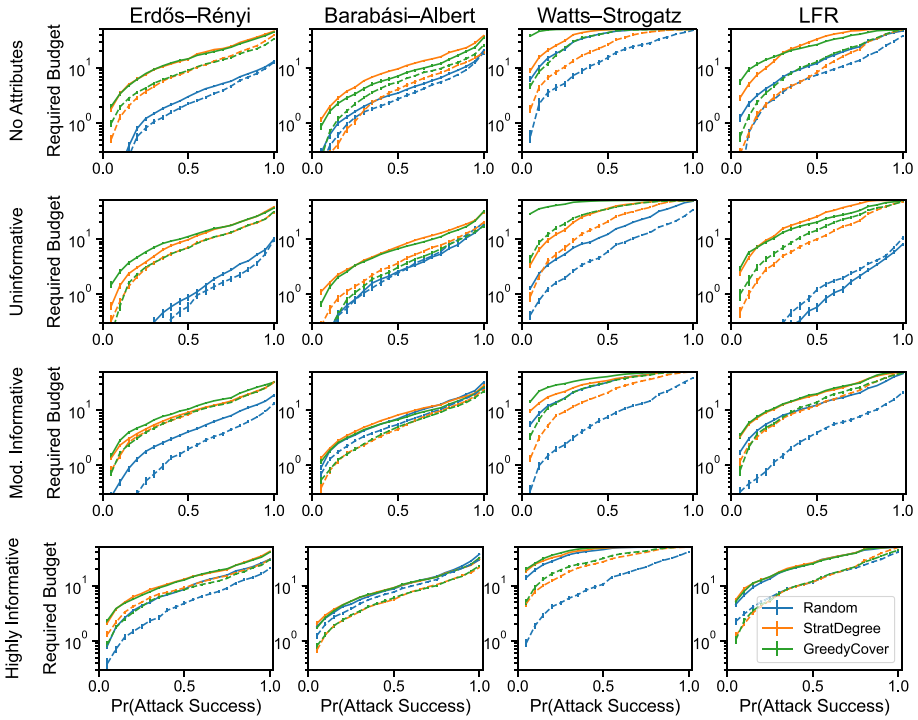


Fig. 3 Robustness to influence attacks against GCNs on simulated data. Results are shown for ER (first column), BA (second column), WS (third column), and LFR (fourth column) graphs, in cases with no attributes (first row), uninformative attributes (second row), moderately informative attributes (third row), and highly informative attributes (fourth row). Each curve represents the average required budget over 25 randomly selected targets, and error bars are standard errors. Higher is better for the defender. Results are shown for high homophily (solid line) and low homophily (dash line) cases. As attributes become more helpful in classification, the advantage gained by the alternative training methods is substantially reduced

Synthetic Data

For each synthetic topology, we ran experiments using Nettack to perform influence attacks against a GCN. We included classifiers trained with data selected via random sampling, StratDegree, and GreedyCover. Robustness results for ER, BA, WS, and LFR graphs are shown in Fig. 3. When no attributes are used and homophily is high, we see a much larger performance difference using GreedyCover than StratDegree in the WS graphs, but the two methods yield more similar performance with the other models. For all models, performance improvement gets more modest as homophily decreases.

When attributes with the same distribution are added to both classes (i.e., the case of "uninformative" attributes), robustness suffers in most cases. The LFR graphs in particular see a large decrease in robustness using random selection, with a much smaller decrease using the alternative methods. As feature distributions become more distinct between the classes, the difference between the methods becomes smaller, suggesting that the robustness improvements we observe are likely due to structural considerations. With highly informative attributes, we note that the models with homogeneous degree distributions still gain a benefit from StratDegree and GreedyCover when homophily is low, while the models with heterogeneous degree distributions are somewhat hindered by these methods. Like in the real data, this is because the targets have higher margins in the case of random training selection. This

may happen due to low-degree nodes, which tend to connect to high-degree nodes: When homophily is low, nodes may become more difficult to predict based on their proximity to hubs, and are *less* likely to be selected to be labeled. We summarize our observations here as follows:

Observation 4.5 With no attributes and high homophily, all models gain robustness from the alternative methods.

Observation 4.6 With no attributes and low homophily, GreedyCover provides robustness for all models, while for BA and LFR, StratDegree improves robustness only at higher attack success rates.

Observation 4.7 The increase in robustness for the alternative methods decreases as homophily decreases and as attributes become better class predictors.

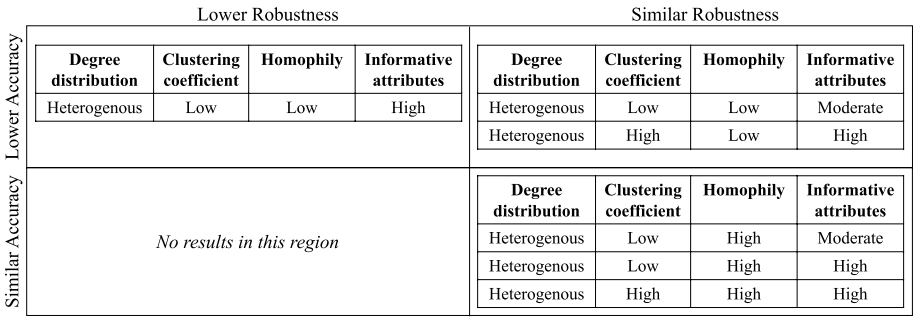
Observation 4.8 With highly informative attributes and low homophily, GreedyCover and StratDegree maintain some increased robustness for homogeneous degree distributions, while they somewhat hinder performance for heterogeneous ones.

As with real data, we consider the possibility that classification performance may be hindered by the alternative training data selection methods. We evaluated classification performance in all cases, with results discussed in detail in Appendix B.5. There are some cases where the alternative methods degrade performance: when homophily is low and degree distributions are heterogeneous. In the high-homophily case, we also see some performance differences, with the alternative methods sometimes yielding higher classification performance than random selection. Any such differences, however, dissipate as the attributes become more informative.

Observation 4.9 Graphs with skewed degree distributions and low homophily achieve lower accuracy with GreedyCover and StratDegree than random selection, but performance is similar in other cases.

Observation 4.10 For higher homophily graphs, performance differences between methods decrease as attributes become more informative.

When there is no homophily in the graph (a node's neighbor is not more likely to have the same label as it is to have another), classification accuracy is very low without informative attributes. Considering cases where there is at least some homophily and at least moderately informative attributes, the simulation results where robustness does not improve with StratDegree or GreedyCover are summarized in Fig. 4. As shown in the table, the cases where there is no improvement all have heterogeneous degree distributions, while the homogeneous degree distributions always have some improvement in robustness when attributes are at least moderately informative. In addition, the low homophily cases result in lower accuracy with the alternative methods. Note also that more informative attributes and lower clustering coefficient hinder the performance benefit.



Lower Robustness, Lower Accuracy			
Degree distribution	**Clustering coefficient**	**Homophily**	**Informative attributes**
Heterogenous	Low	Low	High

Similar Robustness, Lower Accuracy			
Degree distribution	**Clustering coefficient**	**Homophily**	**Informative attributes**
Heterogenous	Low	Low	Moderate
Heterogenous	High	Low	High

Similar Robustness, Similar Accuracy			
Degree distribution	**Clustering coefficient**	**Homophily**	**Informative attributes**
Heterogenous	Low	High	Moderate
Heterogenous	Low	High	High
Heterogenous	High	High	High

Fig. 4 Summary of cases where StratDegree and GreedyCover do not improve robustness when using informative attributes on synthetic graphs. The alternative methods are considered less robust than random training selection if the adversary's budget decreases by at least 1 standard deviation for at least 10 out of 20 points on the associated curve in Fig. 3 (attack success probability in multiples of 0.05). They are considered to be similarly robust if the budget is within 1 standard deviation of for over 10 such points. For accuracy, the alternative methods result in lower accuracy if the average accuracy (see Fig. 9 in Appendix B.5) decreases by at least 3% and similar accuracy if it is within 3%. All cases in the table have heterogeneous degree distributions. All cases with lower accuracy have low homophily. The improvement from the alternatives is also degraded as attributes become more informative (from 70% to 90% accuracy based on attributes alone) and clustering coefficient decreases

Relating the synthetic data to the real datasets, recall that StratDegree and GreedyCover both failed to provide consistent improvement for PubMed, and struggled with direct attacks against PolBlogs. Looking into the features of these datasets, there are two interesting observations. First, PolBlogs has an especially heavy-tailed degree distribution: there are many nodes with hundreds of edges, which is rare in the other datasets. In addition, the PubMed dataset has node attributes that are very useful in identifying the class of the nodes: using a support vector machine with a radial basis function kernel trained on the attribute vectors alone (50% of the nodes used for training), the F_1 score (macro averaged) for the Cora dataset is approximately 0.71, for CiteSeer is approximately 0.75, and for PubMed is about 0.87. As with the synthetic data, the cases with the most informative node attributes are hindered by the alternative training methods.

Adaptive attacks

In the image classification literature, numerous published defenses were found to primarily rely on model obfuscation and remain vulnerable to adaptive attacks that take the new model into account (Athalye and Carlini 2018). Recent work has raised similar concerns regarding the robustness of published defenses against GNN attacks (Mujkanovic et al. 2022). If training set selection makes a classifier more robust, one advantage is that it makes no changes to the model class, and thus should not be vulnerable to such oversights.

We applied our training set selection methods to a demonstration provided by Mujkanovic and Geisler et al,[1] which includes an adaptive attack based on projected gradient descent (PGD) (Xu et al. 2019). The code applies the attack with the objective of reducing the overall classifier accuracy. We applied the demo to the same datasets used by the authors—CiteSeer and Cora—and achieved the results shown in Fig. 5. While the SVD-based method and GNNGuard are both effectively attacked by the PGD-based method,

[1] Available at https://github.com/LoadingByte/are-gnn-defenses-robust.

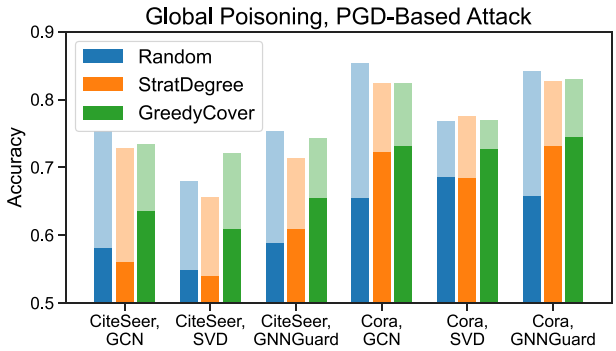


Fig. 5 Performance using an adaptive attack for global poisoning with all three training schemes. Results are shown in terms of overall classifier accuracy using a GCN, an SVD-based GCN, and GNNGuard on the CiteSeer and Cora datasets. Bars showing accuracy before poisoning are desaturated, while accuracy after poisoning is solid. Higher accuracy is better for the defender. In all cases, selecting training data using GreedyCover results in better post-attack accuracy

using GreedyCover to select the training data (again using 10% for training, 10% for validation, and 80% for testing) results in higher post-attack accuracy for with both classifiers. As defenses to new adaptive methods are published, it will be interesting to consider their use in conjunction with alternative training set selection.

Conclusions

This paper explores the impact of complex network characteristics on the robustness of vertex classification using GCNs. In particular, we investigate the hypothesis that the structural relationship between the training data and the remainder of the network can be exploited to improve classifier robustness. We propose two methods to select training data as alternatives to stratified random selection: using the highest degree nodes (StratDegree) and using nodes that result in more connections to training nodes from the test set (GreedyCover). We see the greatest improvement using GreedyCover against influence attacks, though there are improvements in other cases as well. We show that the robustness achieved against Nettack with the alternative training methods is not achieved by increasing the amount of randomly selected training data, and that there is no significant tradeoff between classifier performance and robustness using GreedyCover. In addition, we test StratDegree and GreedyCover against an adaptive global poisoning attack and show that GreedyCover yields better post-attack accuracy than random training.

In simulation, we see other interesting phenomena in the context of influence attacks: GreedyCover increases robustness against Nettack for a diverse set of topologies when label homophily is high and there are no node attributes. We find that GreedyCover and StratDegree cease to be helpful when homophily is very low and degree distributions are heterogeneous, perhaps because there are fewer labels on low-degree nodes that attach to hubs. In all cases, variation between training selection methods becomes less pronounced as node attributes become more helpful in discriminating between classes.

The work documented here points to several open problems and avenues of potential investigation. First, it is interesting that the integrated gradient method is frequently the strongest attack against real data, regardless of how training data are selected. Determining whether some network phenomenon can be exploited to improve robustness against this attack would be an interesting topic for future work. Considering additional models for topologies and attributes could yield additional insight into where the various methods perform best, with Google's GraphWorld being an important enabling technology (Palowitch et al. 2022). Determining the impact of these methods on dynamic GCNs (Pareja et al. 2020), and attacks against dynamic graphs (Sharma et al. 2023), is another important area for future investigation. Another interesting question is whether there are certain topology–attribute combinations where there is a true tradeoff between robustness and classification performance. Identifying such cases—analogous to the work of Tsipras et al. (2019), focused on graph data—would be important to understand what could make classification inherently vulnerable to attack. Another potential area to consider is detectability. Attackers try to hide their manipulation of the data; what would be necessary to determine that an attack has been performed on a graph? For example, we observed that direct attacks from Nettack increase triangle count (Miller et al. 2019). There may be other network statistics that tend to change when an attack is carried out. These are all interesting questions to consider as the research community continues to expand its knowledge of vulnerability and robustness in graph machine learning.

Appendix A Synthetic dataset generation

A.1 Label assignment

We assign labels with varying levels of homophily. For the "high homophily" scenario, we partition the nodes based on the normalized graph Laplacian

$$L = I - D^{-1/2}AD^{-1/2},$$

where A and D are the adjacency matrix and diagonal degree matrix as in the "Problem definition" section (Chung 1997). We select the eigenvector u_2 associated with the second-smallest eigenvalue of L. The nodes associated with the $N/2$ entries in u_2 with the smallest values (i.e., values closest to $-\infty$) are labeled 0, and the other nodes are labeled 1. Let V_0 and V_1 be the respective subsets of vertices.

For lower homophily graphs, we first compute the difference between the number of within-label edges and the number of cross-label edges, i.e., letting E_{ij} be the set of edges between nodes in V_i and nodes in V_j,

$$\Delta = |E_{00}| + |E_{11}| - |E_{01}|. \tag{4}$$

Depending on how homophilous we want the graph to be, we swap labels on pairs of nodes until Δ reaches a given value, based on its value from the initial Laplacian-based partition (e.g., half as homophilous as the original). The node swapping mechanism is detailed in Algorithm 2.

Algorithm 2 SWAP2REDUCE

Algorithm 2 SWAP2REDUCE

Input: Graph $G = (V, E)$, vertex subsets V_0, V_1
Output: Updated vertex subsets $\hat{V}_0, \hat{V}_1$
$x_i \leftarrow \begin{cases} -1 & \text{if } v_i \in V_0 \\ 1 & \text{if } v_i \in V_1 \end{cases}$
$d \leftarrow Ax$
$p_i \leftarrow d_i \cdot \mathbb{I}[d_i >= 0] \cdot \mathbb{I}[x_i > 0]$
$u \leftarrow$ node randomly selected with probability $p_i / \sum_{j=1}^{N} p_j$
$p_i \leftarrow d_i \mathbb{I}[d_i < 0] \mathbb{I}[x_i < 0]$
$v \leftarrow$ node randomly selected with probability $p_i / \sum_{j=1}^{N} p_j$
$\hat{V}_0 \leftarrow (V_0 \cup \{u\}) \setminus v$
$\hat{V}_1 \leftarrow (V_1 \cup \{v\}) \setminus u$
return $\hat{V}_0, \hat{V}_1$

A.2 Synthetic attributes

As with the real graphs, we consider binary attribute vectors on the nodes of the synthetic graphs. In each case, we consider nodes with 20 attributes, and we give each attribute a probability of being true depending on its label. Probabilities are determined by an exponentially decreasing function. We consider three scenarios. In the most difficult case, the probabilities are the same for both classes. As we make the problem easier, we shift the function that determines the attribute probabilities so that high-probability attributes in class 0 still have *relatively* high probabilities class 0, but there is not a perfect match. The shifts were chosen to create cases where a generalized likelihood ratio test (with each attribute having an independent probability parameter, estimated based on 60 cases for each class) achieves accuracy of approximately 0.5, 0.7, and 0.9. We refer to these cases as having *uninformative*, *moderately informative*, and *highly informative* attributes, respectively. In addition, each node has a one-hot encoded attribute indicating its index in the node set.

Appendix B Detailed experimental results

B.1 Extended robustness results

We present results for each classifier at various attacker budgets, and highlight the best-performing pairing of a classifier with a training data selection method. Results on influence attacks for CiteSeer, Cora, Polblogs, and PubMed are in Tables 1, 2, 3, and 4, respectively. Results for direct attacks are likewise in Tables 5, 6, 7, and 8. As shown in the "Real data" section, GREEDYCOVER and STRATDEGREE perform best for CiteSeer and Cora, and for PolBlogs in the case of influence attacks.

Table 1 Results of influence attacks against each classifier with the CiteSeer dataset, with attacker budgets of 10, 30, and 50 edge perturbations

		Budget 10			Budget 30			Budget 50		
defense	**train.**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	*0.248*	0.224	N/A	*0.432*	0.336	N/A	*0.504*	0.408	N/A
Jaccard	SD	0.28	0.24	N/A	0.4	0.36	N/A	0.448	0.368	N/A
Jaccard	GC	0.152	**0.168**	N/A	0.176	**0.288**	N/A	0.192	**0.36**	N/A
RGCN	Rand.	0.752	0.696	N/A	0.944	0.84	N/A	0.976	0.904	N/A
RGCN	SD	0.504	0.328	N/A	0.768	0.592	N/A	0.816	0.744	N/A
RGCN	GC	0.448	0.384	N/A	0.864	0.76	N/A	0.952	0.824	N/A
Cheb	Rand.	0.304	0.384	N/A	0.496	0.464	N/A	0.552	0.544	N/A
Cheb	SD	0.32	0.352	N/A	0.464	0.432	N/A	0.52	0.496	N/A
Cheb	GC	0.352	0.272	N/A	0.52	0.424	N/A	0.552	0.512	N/A
SVD	Rand.	0.536	0.424	N/A	0.816	0.696	N/A	0.904	0.88	N/A
SVD	SD	0.624	0.6	N/A	0.912	0.776	N/A	0.968	0.928	N/A
SVD	GC	0.344	0.376	N/A	0.624	0.6	N/A	0.808	0.688	N/A
median	Rand.	0.584	0.44	N/A	0.816	0.84	N/A	0.864	0.88	N/A
median	SD	0.424	0.376	N/A	0.8	0.784	N/A	0.88	0.912	N/A
median	GC	0.392	0.304	N/A	0.768	0.768	N/A	0.88	0.864	N/A
GAT	Rand.	0.624	0.568	N/A	0.928	0.848	N/A	0.976	0.936	N/A
GAT	SD	0.552	0.392	N/A	0.816	0.728	N/A	0.912	0.896	N/A
GAT	GC	0.424	0.328	N/A	0.864	0.752	N/A	0.936	0.888	N/A
GCN	Rand.	0.68	0.568	N/A	0.848	0.856	N/A	0.872	0.904	N/A
GCN	SD	0.472	0.464	N/A	0.792	0.728	N/A	0.84	0.768	N/A
GCN	GC	0.408	0.368	N/A	0.728	0.768	N/A	0.832	0.872	N/A
SGC	Rand.	0.616	N/A	N/A	0.824	N/A	N/A	0.872	N/A	N/A
SGC	SD	0.696	N/A	N/A	0.8	N/A	N/A	0.816	N/A	N/A
SGC	GC	0.6	N/A	N/A	0.824	N/A	N/A	0.912	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in bold. Entries representing the most robust case for random training are in italic. Entries listed as N/A did not finish in the allotted time (24 h per trial). The Jaccard-based classifier performs best, both overall (with GreedyCover) and using random training

B.2 Direct attacks against real datasets

When considering direct attacks, we use all four attacks, again with SGA replacing Nettack in the appropriate case. Results of these experiments are shown in Fig. 6. It is much more difficult to defend against direct attacks; note that the attacker often only needs one or two perturbations to be successful. With the CiteSeer dataset, we once again see higher robustness with GreedyCover and StratDegree, in particular at low attack probabilities. With Nettack and FGA, GreedyCover improves performance when combined with other defenses. With

Table 2 Results of influence attacks against each classifier with the Cora dataset, with attacker budgets of 10, 30, and 50 edge perturbations

		Budget 10			Budget 30			Budget 50		
Defense	**Training**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	*0.296*	0.24	N/A	*0.472*	0.432	N/A	*0.592*	0.504	N/A
Jaccard	SD	0.264	0.192	N/A	**0.352**	0.312	N/A	0.4	**0.424**	N/A
Jaccard	GC	0.224	**0.256**	N/A	0.36	0.392	N/A	0.448	0.456	N/A
RGCN	Rand.	0.68	0.6	N/A	0.912	0.944	N/A	0.952	0.976	N/A
RGCN	SD	0.448	0.408	N/A	0.848	0.784	N/A	0.904	0.888	N/A
RGCN	GC	0.44	0.296	N/A	0.832	0.808	N/A	0.92	0.888	N/A
Cheb	Rand.	0.448	0.448	N/A	0.784	0.808	N/A	0.872	0.896	N/A
Cheb	SD	0.656	0.48	N/A	0.976	0.816	N/A	0.976	0.912	N/A
Cheb	GC	0.488	0.448	N/A	0.864	0.88	N/A	0.928	0.912	N/A
SVD	Rand.	0.224	0.32	N/A	0.536	0.72	N/A	0.76	0.872	N/A
SVD	SD	0.296	0.216	N/A	0.568	0.536	N/A	0.76	0.728	N/A
SVD	GC	0.264	0.264	N/A	0.584	0.528	N/A	0.84	0.768	N/A
median	Rand.	0.544	0.424	N/A	0.872	0.808	N/A	0.944	0.912	N/A
median	SD	0.4	0.424	N/A	0.8	0.784	N/A	0.912	0.888	N/A
median	GC	0.24	0.312	N/A	0.784	0.84	N/A	0.896	0.912	N/A
GAT	Rand.	0.544	0.552	N/A	0.904	0.88	N/A	0.96	0.952	N/A
GAT	SD	0.608	0.48	N/A	0.88	0.816	N/A	0.968	0.872	N/A
GAT	GC	0.48	0.352	N/A	0.872	0.768	N/A	0.952	0.952	N/A
GCN	Rand.	0.568	0.448	N/A	0.896	0.896	N/A	0.936	0.952	N/A
GCN	SD	0.456	0.32	N/A	0.752	0.696	N/A	0.856	0.872	N/A
GCN	GC	0.384	0.32	N/A	0.816	0.776	N/A	0.896	0.896	N/A
SGC	Rand.	0.568	N/A	N/A	0.824	N/A	N/A	0.888	N/A	N/A
SGC	SD	0.632	N/A	N/A	0.84	N/A	N/A	0.856	N/A	N/A
SGC	GC	0.584	N/A	N/A	0.824	N/A	N/A	0.88	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in **bold**. Entries representing the most robust case for random training are in italic. Entries listed as N/A did not finish in the allotted time (24 h per trial). The Jaccard-based classifier performs best, both overall and using random training. If we focus on classifiers that achieve the best performance in Fig. 8, (i.e., omitting Jaccard and SVD), the best performance is achieved by GCNs with the alternative training methods

IG-FGSM, on the other hand, the alternative training methods provide little benefit. Across datasets, this attack also has the lowest robustness across defenses, which would suggest it is a preferred attack for adversaries. (Note that for CiteSeer, we only obtained data for a GCN classifier when attacked with IG-FGSM when using GreedyCover.) We once again see a detriment in performance with PubMed, though in this case in the area where perturbing a single edge results in an successful attack.

Table 3 Results of influence attacks against each classifier with the PolBlogs dataset, with attacker budgets of 10, 30, and 50 edge perturbations

		Budget 10			Budget 30			Budget 50		
Defense	**Training**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	0.92	0.872	0.928	0.992	1.0	1.0	1.0	1.0	1.0
Jaccard	SD	0.928	0.984	0.936	1.0	1.0	1.0	1.0	1.0	1.0
Jaccard	GC	0.936	0.952	0.944	0.992	1.0	1.0	1.0	1.0	1.0
RGCN	Rand.	0.08	0.128	0.032	0.224	0.248	0.056	0.32	0.304	0.064
RGCN	SD	**0.048**	0.024	0.016	**0.088**	0.072	0.056	**0.112**	0.096	0.064
RGCN	GC	0.088	0.096	0.04	0.192	0.176	0.048	0.232	0.272	0.056
Cheb	Rand.	0.048	0.184	0.112	0.184	0.376	0.176	0.296	0.456	0.216
Cheb	SD	0.128	0.128	0.168	0.256	0.232	0.288	0.368	0.32	0.336
Cheb	GC	0.344	0.08	0.168	0.456	0.208	0.328	0.52	0.288	0.392
SVD	Rand.	0.016	*0.08*	0.048	0.04	*0.12*	0.096	0.112	*0.136*	0.128
SVD	SD	0.064	0.024	0.064	**0.088**	0.024	0.072	**0.112**	0.032	0.096
SVD	GC	0.04	0.04	**0.048**	**0.088**	0.064	0.08	0.104	0.112	0.128
GAT	Rand.	0.224	0.224	0.184	0.384	0.408	0.248	0.448	0.488	0.304
GAT	SD	0.12	0.056	0.08	0.184	0.16	0.152	0.256	0.208	0.208
GAT	GC	0.112	0.12	0.064	0.2	0.256	0.152	0.288	0.296	0.168
GCN	Rand.	0.16	0.208	0.128	0.288	0.368	0.2	0.344	0.424	0.232
GCN	SD	0.104	0.096	0.176	0.168	0.208	0.248	0.208	0.288	0.312
GCN	GC	0.072	0.032	0.056	0.152	0.184	0.104	0.28	0.296	0.128
SGC	Rand.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SGC	SD	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SGC	GC	0.056	N/A	N/A	0.12	N/A	N/A	0.168	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in bold. Entries representing the most robust case for random training are in italic. Entries listed as N/A did not finish in the allotted time (24 h per trial). Best results overall and with random training are achieved with SVD, while RGCN performs equally well when using StratDegree

Table 4 Results of influence attacks against each classifier with the PubMed dataset, with attacker budgets of 10, 30, and 50 edge perturbations

		Budget 10			Budget 30			Budget 50		
Defense	**Training**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	0.224	N/A	N/A	0.576	N/A	N/A	0.744	N/A	N/A
Jaccard	SD	0.12	N/A	N/A	0.136	N/A	N/A	0.16	N/A	N/A
Jaccard	GC	0.128	N/A	N/A	0.192	N/A	N/A	0.2	N/A	N/A
GCN	Rand.	0.456	N/A	N/A	0.76	N/A	N/A	0.888	N/A	N/A
GCN	SD	0.6	N/A	N/A	0.936	N/A	N/A	0.976	N/A	N/A
GCN	GC	0.544	N/A	N/A	0.88	N/A	N/A	0.952	N/A	N/A
Cheb	Rand.	***0.056***	N/A	N/A	***0.072***	N/A	N/A	***0.072***	N/A	N/A
Cheb	SD	0.072	N/A	N/A	0.128	N/A	N/A	0.136	N/A	N/A
Cheb	GC	0.136	N/A	N/A	0.16	N/A	N/A	0.192	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in bold. Entries listed as N/A did not finish in the allotted time (24 h per trial). Only results using Jaccard, GCN, and ChebNet were obtained in time. While StratDegree and GreedyCover improve performance with the Jaccard-based classifier, the best performance is achieved by a ChebNet classifier with random training. In our experiments, this classifier with the PubMed data typically has a much higher margin before the attack takes place

Table 5 Results of direct attacks against each classifier with the CiteSeer dataset, with attacker budgets of 5, 10, and 20 edge perturbations

		Budget 5			Budget 10			Budget 20		
Defense	**Training**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	0.384	0.256	0.296	0.592	0.392	0.368	0.752	0.464	0.472
Jaccard	SD	0.432	0.32	0.256	0.672	0.416	0.32	0.808	0.52	0.432
Jaccard	GC	0.2	0.184	**0.224**	0.264	**0.336**	**0.336**	0.408	0.52	0.456
RGCN	Rand.	0.976	0.848	0.8	0.992	0.936	0.936	1.0	0.968	0.96
RGCN	SD	0.88	0.568	0.912	0.992	0.856	0.976	1.0	0.944	0.976
RGCN	GC	0.896	0.712	0.808	1.0	0.88	0.936	1.0	0.952	0.976
Cheb	Rand.	0.224	*0.28*	0.264	0.344	0.352	*0.384*	0.424	0.44	*0.464*
Cheb	SD	0.264	0.24	0.304	0.32	0.336	0.392	0.4	0.376	**0.448**
Cheb	GC	0.288	0.24	0.256	0.376	0.312	0.328	0.472	0.4	0.4
SVD	Rand.	0.552	0.392	0.768	0.76	0.576	0.936	0.944	0.856	0.952
SVD	SD	0.84	0.544	0.936	0.984	0.696	0.976	1.0	0.856	0.976
SVD	GC	0.408	0.312	0.864	0.688	0.488	0.952	0.968	0.792	0.992
median	Rand.	0.808	0.792	0.792	0.96	0.936	0.96	0.984	0.952	0.984
median	SD	0.904	0.84	0.872	0.992	0.952	0.952	1.0	0.96	0.952
median	GC	0.856	0.832	0.848	0.96	0.952	0.96	0.992	0.968	0.976
GAT	Rand.	0.92	0.864	0.84	0.984	0.952	0.936	1.0	0.96	0.952
GAT	SD	0.944	0.808	0.952	1.0	0.952	0.992	1.0	0.984	1.0
GAT	GC	0.936	0.808	0.832	1.0	0.952	0.92	1.0	0.984	0.96
GCN	Rand.	0.944	0.872	N/A	0.992	0.952	N/A	1.0	0.976	N/A
GCN	SD	0.984	0.832	N/A	1.0	0.968	N/A	1.0	0.992	N/A
GCN	GC	0.912	0.904	0.936	1.0	0.976	0.992	1.0	0.976	0.992
SGC	Rand.	0.832	N/A	N/A	0.944	N/A	N/A	1.0	N/A	N/A
SGC	SD	0.936	N/A	N/A	1.0	N/A	N/A	1.0	N/A	N/A
SGC	GC	0.88	N/A	N/A	0.96	N/A	N/A	1.0	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in bold. Entries representing the most robust case for random training are in italic. Entries listed as N/A did not finish in the allotted time (24 h per trial). As with influence attacks, the Jaccard-based classifier performs best, though ChebNet also performs well for all training methods

B.3 Impact of additional training data

Results of experiments with greater proportions of labeled data are shown in Fig. 7, using Nettack as an influence attack against a GCN. While using more randomly selected training data does sometimes increase robustness, it is not consistent, and in some cases more randomly selected training data results in a slightly less robust classifier. The one case where additional training data consistently outperforms GreedyCover in terms of robustness is Cora, where training using 30% of the dataset, randomly selected, outperforms the alternatives. In this case, the average number of neighbors per target for GreedyCover and StratDegree are 1.084 and

Table 6 Results of direct attacks against each classifier with the Cora dataset, with attacker budgets of 5, 10, and 20 edge perturbations

		Budget 5			Budget 10			Budget 20		
Defense	**Training**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	0.504	0.328	N/A	*0.712*	0.48	N/A	*0.952*	0.68	N/A
Jaccard	SD	0.448	0.216	N/A	**0.656**	0.36	N/A	**0.776**	0.552	N/A
Jaccard	GC	0.528	0.296	N/A	0.76	0.424	N/A	0.912	0.616	N/A
RGCN	Rand.	0.936	0.896	N/A	0.984	0.992	N/A	0.992	0.992	N/A
RGCN	SD	0.944	0.832	N/A	1.0	0.952	N/A	1.0	0.96	N/A
RGCN	GC	0.976	0.84	0.832	1.0	0.96	0.976	1.0	0.96	0.976
Cheb	Rand.	0.88	0.728	N/A	0.976	0.928	N/A	0.984	0.96	N/A
Cheb	SD	0.96	0.752	N/A	1.0	0.928	N/A	1.0	0.936	N/A
Cheb	GC	0.944	0.816	N/A	0.992	0.952	N/A	1.0	0.96	N/A
SVD	Rand.	***0.36***	0.24	N/A	0.776	0.592	N/A	0.992	0.928	N/A
SVD	SD	0.696	0.288	N/A	0.936	0.632	N/A	1.0	0.92	N/A
SVD	GC	0.432	0.184	N/A	0.792	0.448	N/A	1.0	0.84	N/A
median	Rand.	0.936	0.768	0.824	0.992	0.96	0.968	1.0	0.976	0.992
median	SD	0.968	0.864	0.864	1.0	0.952	0.984	1.0	0.952	0.984
median	GC	0.912	0.824	0.824	1.0	0.968	0.976	1.0	0.968	0.976
GAT	Rand.	0.944	0.856	N/A	1.0	0.96	N/A	1.0	0.968	N/A
GAT	SD	0.928	0.736	N/A	1.0	0.92	N/A	1.0	0.968	N/A
GAT	GC	0.92	0.824	N/A	1.0	0.952	N/A	1.0	0.984	N/A
GCN	Rand.	0.928	0.888	N/A	0.992	0.976	N/A	1.0	0.976	N/A
GCN	SD	0.928	0.624	0.904	0.992	0.944	0.992	1.0	0.968	0.992
GCN	GC	0.904	0.832	0.808	0.992	0.976	0.984	1.0	0.984	0.992
SGC	Rand.	0.896	N/A	N/A	1.0	N/A	N/A	1.0	N/A	N/A
SGC	SD	0.944	N/A	N/A	1.0	N/A	N/A	1.0	N/A	N/A
SGC	GC	0.904	N/A	N/A	1.0	N/A	N/A	1.0	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in bold. Entries representing the most robust case for random training are in italic. Entries listed as N/A did not finish in the allotted time (24 h per trial). While random training with the SVD classifier works best at a low attack budget, Jaccard with StratDegree performs better against better-resourced attackers

1.135, both between the values for 25% random training (1.029) and 30% (1.237). Thus, increasing the number of neighbors in the training set by adding more randomly selected training data does not necessarily increase classifier robustness to the same extent.

B.4 Robustness vs. Classification performance

In Fig. 8, we show the macro-averaged F_1 score for each method using all classifiers. Performance does occasionally vary. In particular, StratDegree results in somewhat lower performance than random training among most classifiers for all datasets.

Table 7 Results of direct attacks against each classifier with the PolBlogs dataset, with attacker budgets of 5, 10, and 20 edge perturbations

		Budget 5			Budget 10			Budget 20		
Defense	**Training**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	0.672	0.616	0.688	0.896	0.848	0.856	0.992	0.968	0.976
Jaccard	SD	0.752	0.8	0.808	0.936	0.928	0.952	1.0	0.984	1.0
Jaccard	GC	0.768	0.76	0.872	0.904	0.92	0.976	0.992	1.0	1.0
RGCN	Rand.	0.48	0.464	0.312	0.632	0.664	0.504	0.784	0.872	0.704
RGCN	SD	0.304	0.28	0.344	0.4	0.44	0.52	0.648	0.568	0.736
RGCN	GC	0.56	0.488	0.336	0.72	0.648	0.456	0.832	0.832	0.64
Cheb	Rand.	0.376	0.352	0.448	0.536	0.552	0.544	0.72	0.68	0.744
Cheb	SD	0.408	0.376	0.432	0.624	0.544	0.584	0.784	0.688	0.768
Cheb	GC	0.576	0.424	0.472	0.68	0.568	0.592	0.8	0.728	0.768
SVD	Rand.	0.184	0.056	*0.336*	0.368	0.104	0.552	0.496	0.192	0.704
SVD	SD	0.288	0.016	0.368	0.416	0.024	0.536	0.496	0.064	0.808
SVD	GC	0.12	0.04	**0.32**	0.36	0.04	0.512	0.464	0.048	0.768
GAT	Rand.	0.456	0.52	0.376	0.624	0.792	0.52	0.824	0.912	0.672
GAT	SD	0.32	0.24	0.36	0.464	0.384	0.504	0.664	0.648	0.752
GAT	GC	0.552	0.528	0.336	0.744	0.84	0.576	0.856	0.936	0.84
GCN	Rand.	0.472	0.624	0.408	0.68	0.784	0.52	0.816	0.92	0.76
GCN	SD	0.424	0.344	0.408	0.568	0.528	0.544	0.76	0.76	0.672
GCN	GC	0.496	0.552	0.312	0.72	0.768	0.512	0.88	0.912	0.728
SGC	Rand.	0.36	N/A	N/A	***0.464***	N/A	N/A	***0.6***	N/A	N/A
SGC	SD	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SGC	GC	0.528	N/A	N/A	0.736	N/A	N/A	0.856	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in bold. Entries representing the most robust case for random training are in italic. Entries listed as N/A did not finish in the allotted time (24 h per trial). The best-performing cases for attacks with 10 or 20 perturbations use random sampling with an SGC classifier, though in these cases FGA and IG-FGSM were unavailable to the attacker. (If we *only* consider Nettack, SVD with GreedyCover consistently performs best.)

Table 8 Results of direct attacks against each classifier with the PubMed dataset, with attacker budgets of 5, 10, and 20 edge perturbations

		Budget 5			Budget 10			Budget 20		
Defense	**Training**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**	**Net**	**FGA**	**IG**
Jaccard	Rand.	0.784	N/A	N/A	0.952	N/A	N/A	0.992	N/A	N/A
Jaccard	SD	0.208	N/A	N/A	0.248	N/A	N/A	0.328	N/A	N/A
Jaccard	GC	0.208	N/A	N/A	0.328	N/A	N/A	0.456	N/A	N/A
GCN	Rand.	0.92	N/A	N/A	1.0	N/A	N/A	1.0	N/A	N/A
GCN	SD	0.952	N/A	N/A	1.0	N/A	N/A	1.0	N/A	N/A
GCN	GC	0.936	N/A	N/A	0.984	N/A	N/A	1.0	N/A	N/A
Cheb	Rand.	***0.056***	N/A	N/A	***0.088***	N/A	N/A	***0.088***	N/A	N/A
Cheb	SD	0.072	N/A	N/A	**0.088**	N/A	N/A	0.104	N/A	N/A
Cheb	GC	0.088	N/A	N/A	0.112	N/A	N/A	0.152	N/A	N/A
GAT	Rand.	0.792	N/A	N/A	0.896	N/A	N/A	0.992	N/A	N/A
GAT	SD	0.936	N/A	N/A	0.992	N/A	N/A	1.0	N/A	N/A
GAT	GC	0.92	N/A	N/A	0.984	N/A	N/A	0.992	N/A	N/A

Results are included for Nettack (Net), FGA, and IG-FGSM (IG). For each classifier, we train with random (Rand.), StratDegree (SD), and GreedyCover (GC). Each entry is a probability of attack success, thus higher is better for the attacker and lower is better for the defender. To yield the most robust classifier, the defender picks the classifier/training method combination that minimizes the worst-case attack probability. These entries are listed in bold. Entries representing the most robust case for random training are in italic. Entries listed as N/A did not finish in the allotted time (24 h per trial). While StratDegree and GreedyCover work well in conjunction with the Jaccard classifier, a disparity in the classification margin hinders their performance in the best case, using a ChebNet classifier

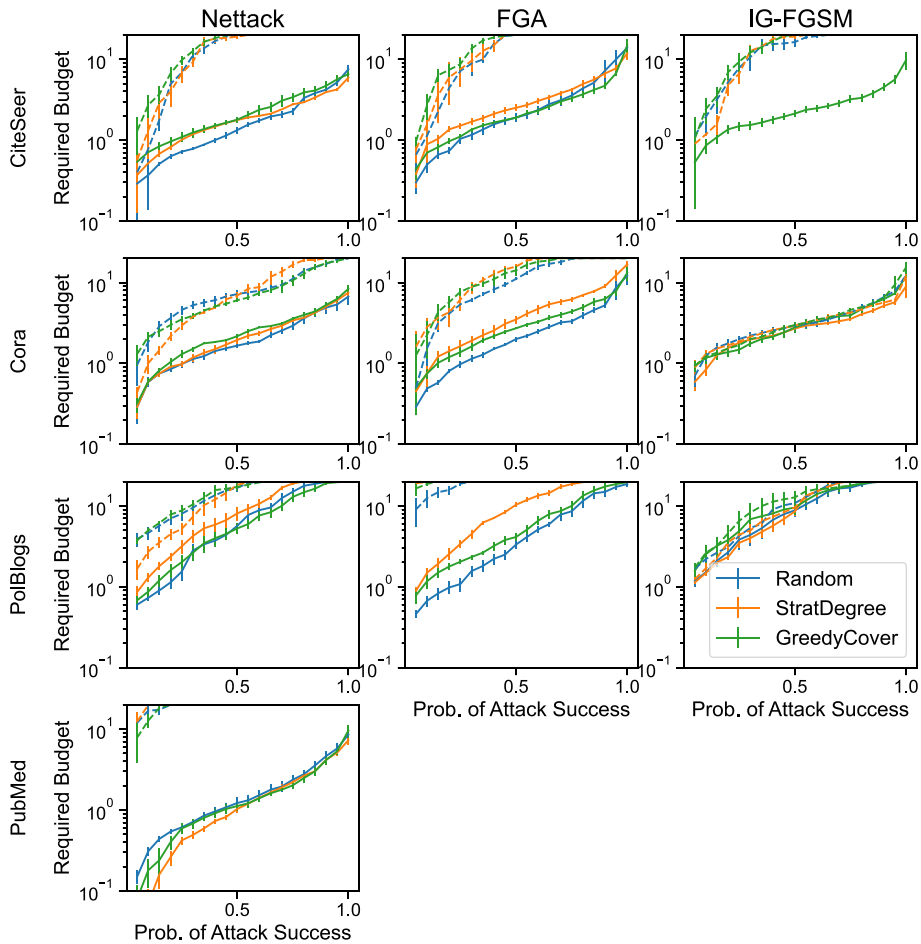


Fig. 6 Robustness to direct attacks using GCNs (solid line) or with the best defense at a given attack success probability (dash line). Results are shown for the CiteSeer, Cora, PolBlogs, and PubMed datasets, attacked with Nettack/SGA (left column), FGA (center column), and IG-FGSM (right column). Results were not returned in the allotted time (24 h per trial) for IG-FGSM and FGA on the PubMed dataset, or for IG-FGSM on the CiteSeer dataset when using a GCN with random training or StratDegree. (CiteSeer/IG-FGSM experiments with StratDegree and random selection completed for other classifiers; see Table 5 for details.) Each curve represents the average required budget over 25 randomly selected targets, and error bars are standard errors. Higher is better for the defender. While GreedyCover performs better when paired with defenses on CiteSeer when attacked with Nettack or FGA, the alternative methods generally increase robustness less than with indirect attacks

GreedyCover, on the other hand, typically yields similar performance to random selection and occasionally outperforms it, e.g., using SGC on CiteSeer and ChebNet on Cora.

B.5 Robustness vs. Classification performance—synthetic data

We consider the potential impact of the alternative training data on classifier performance as well. Results are shown in Fig. 9. Since we use two balanced classes in all cases,

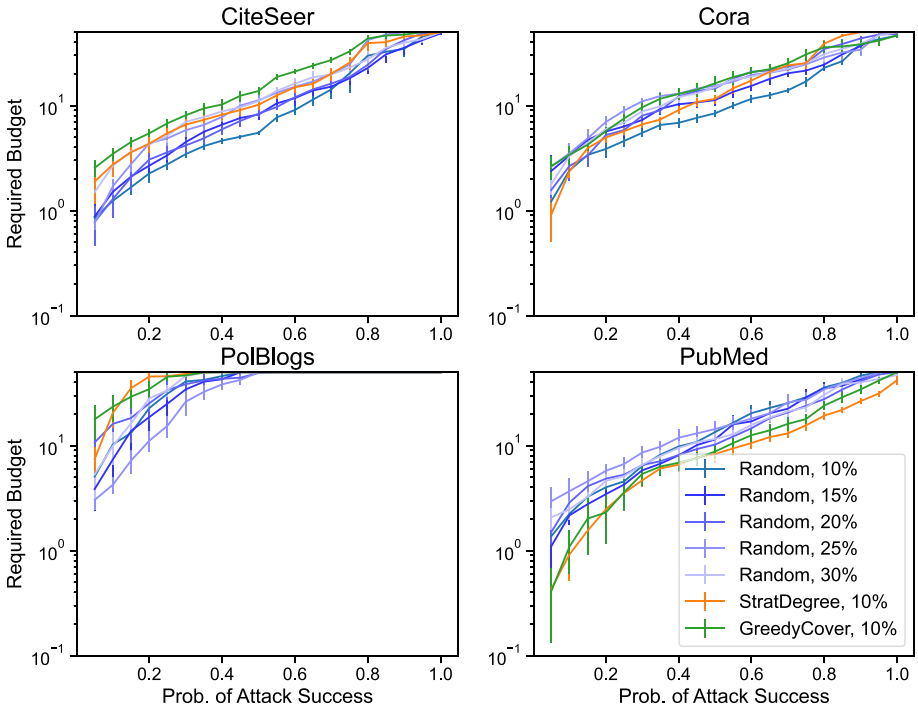


Fig. 7 Robustness to influence attacks using GCNs when training data are selected using GreedyCover, StratDegree, or varying amounts of random selection. Results are shown for the CiteSeer (upper left), Cora (upper right), PolBlogs (lower left), and PubMed (lower right) datasets. Each curve represents the average required budget over 25 randomly selected targets, and error bars are standard errors. Higher is better for the defender. Of the datasets where robustness improves using GreedyCover (i.e., CiteSeer, Cora, and PolBlogs), the only case that consistently performs better than GreedyCover is 30% random selection on the Cora dataset

we use accuracy as the classification metric. For each case, we plot accuracy as a function of *heterophilicity* Park and Barabási (2007), computed as

$$H = \frac{\#\{\text{edges between } V_0 \text{and} V_1\}}{|V_0||V_1|M/\binom{N}{2}}. \tag{5}$$

The denominator in (5) is the expected number of edges between V_0 and V_1 after random rewiring. A high-homophily graph will have relatively low heterophilicity. Note that both ER and BA graphs span the same range of heterophilicity, while LFR graph can achieve lower heterophilicity and WS can be almost perfectly homophilous. When no attributes are used, performance is similar across methods in the high-homophily (low-heterophilicity) cases, while the alternative methods perform worse in the low-homophily cases.

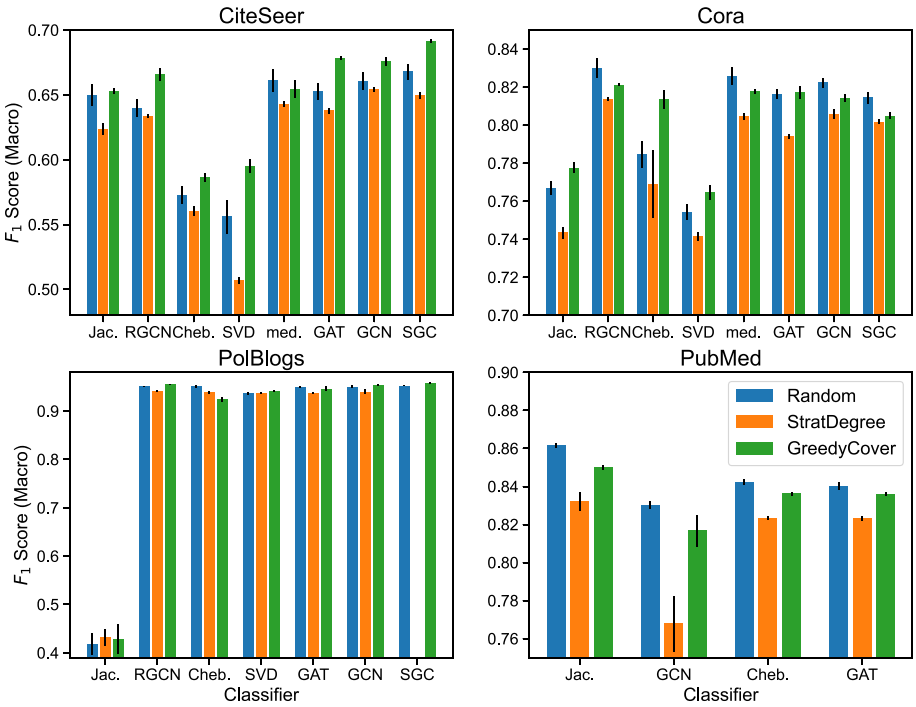


Fig. 8 Classifier performance across datasets when training data are selected using GreedyCover, StratDegree, or varying amounts of random selection. Results are shown for the CiteSeer (upper left), Cora (upper right), PolBlogs (lower left), and PubMed (lower right) datasets. Each bar height represents the average F_1 score (macro averaged) across 5 separate train/validation/test sets, and error bars are standard errors. Performance is shown for each classifier where experiments completed within the allotted time (24 h per trial). Higher is better for the defender. While StratDegree often underperforms random selection, GreedyCover typically shows similar performance

This yields a significant gap in the in the cases with skewed degree distributions. In particular, LFR graphs maintain 75% accuracy with random training even in the case where there is no homophily (heterophilicity is 1). As in the analogous results in Fig. 3, this may be due to low-degree nodes that are unlikely to be chosen as training data, but are more difficult to classify in a less homogeneous setting.

As attributes are added to the graphs, we see a decrease in performance when the uninformative attributes are added, though the difference is very small using GreedyCover for the clustered models. As we expect, accuracy increases as the attributes become more informative. As we observed in the robustness results, we see differences between methods diminish as attributes help discriminate the classes.

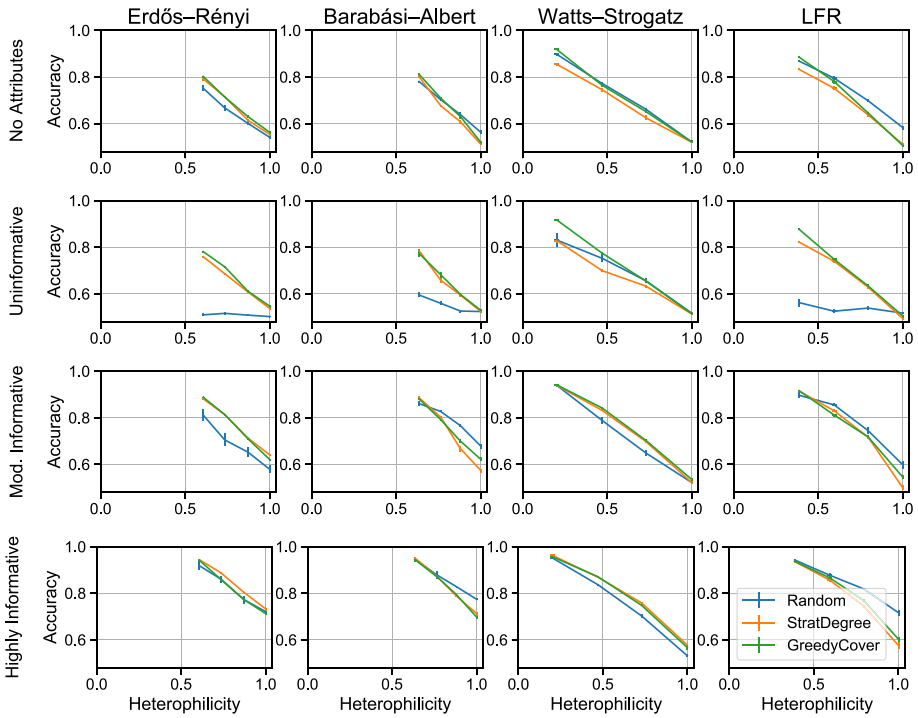


Fig. 9 Classification accuracy as a function of heterophilicity using GCNs on simulated data. Results are shown for ER (first column), BA (second column), WS (third column), and LFR (fourth column) graphs, in cases with no attributes (first row), uninformative attributes (second row), moderately informative attributes (third row), and highly informative attributes (fourth row). Each curve represents the average required budget over 5 train/validation/test splits, and error bars are standard errors. Higher is better for the defender. The principal performance differences occur with skewed degree distributions when homophily is low

Acknowledgements
The authors wish to thank Mustafa Çamurcu and Alexander J. Gomez for their assistance with the initial experiments on StratDegree and GreedyCover

Authors' contributions
This manuscript would not exist without BAM; he was the lead contributor in all aspects of the manuscript. TER supervised the work and contributed to methodology and experimental design. KC was involved in the development and analysis of the methods. BAM was the lead developer of the code and conducted all the experiments. BAM was the lead contributor in writing the manuscript. All authors read and approved the final manuscript.

Funding
Open access funding provided by Northeastern University Library. This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001 and the Combat Capabilities Development Command Army Research Laboratory (under Cooperative Agreement Number W911NF-13-2-0045). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force or Army Research Laboratory.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 18 November 2023 Accepted: 30 January 2024
Published online: 21 February 2024

References
Albert R, Jeong H, Barabási AL (2000) Error and attack tolerance of complex networks. Nature 406(6794):378–382

Athalye A, Carlini N (2018) On the robustness of the CVPR 2018 white-box adversarial example defenses. CoRR abs/1804.03286
Athalye A, Carlini N, Wagner D (2018) Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: ICML, pp 274–283
Bojchevski A, Günnemann S (2019) Certifiable robustness to graph perturbations. In: NeurIPS, pp 8317–8328
Carlini N et al (2019) On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705
Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: SP, pp 39–57
Chen L, Li J, Peng Q et al (2021) Understanding structural vulnerability in graph convolutional networks. In: IJCAI
Chen J, Wu Y, Xu X et al (2018) Fast gradient attack on network embedding. arXiv preprint arXiv:1809.02797
Chung FR (1997) Spectral Graph Theory. American Mathematical Soc
Croce F, Andriushchenko M, Hein M (2019) Provable robustness of relu networks via maximization of linear regions. In: AISTATS, pp 2057–2066
Dai E, Jin W, Liu H et al (2022) Towards robust graph neural networks for noisy graphs with sparse labels. In: WSDM, pp 181–191
Dai H, Li H, Tian T et al (2018) Adversarial attack on graph structured data. In: ICML, pp 1115–1124
Dapello J, Feather J, Le H et al (2021) Neural population geometry reveals the role of stochasticity in robust perception. In: Ranzato M, Beygelzimer A, Dauphin Y, et al (eds) NeurIPS, vol 34. Curran Associates, Inc., pp 15595–15607, https://proceedings.neurips.cc/paper/2021/file/8383f931b0cefcc631f070480ef340e1-Paper.pdf
Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Lee D, Sugiyama M, Luxburg U et al (eds) NeurIPS, https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf
Entezari N, Al-Sayouri SA, Darvishzadeh A et al (2020) All you need is low (rank): defending against adversarial attacks on graphs. In: WSDM, pp 169–177
Ficara A, Curreri F, Fiumara G et al (2023) Human and social capital strategies for mafia network disruption. IEEE Trans Inf Forensics Secur 18:1926–1936
Geisler S, Zügner D, Günnemann S (2020) Reliable graph neural networks via robust aggregation. In: NeurIPS, pp 13272–13284
Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: ICLR
Hamilton WL, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: NeurIPS, pp 1025–1035
Jia R, Liang P (2017) Adversarial examples for evaluating reading comprehension systems. In: EMNLP, pp 2021–2031
Jin W, Derr T, Wang Y et al (2021) Node similarity preserving graph convolutional networks. In: WSDM, pp 148–156
Jin W, Li Y, Xu H et al (2020a) Adversarial attacks and defenses on graphs: a review, a tool and empirical studies. arXiv preprint arXiv:2003.00653
Jin W, Ma Y, Liu X et al (2020b) Graph structure learning for robust graph neural networks. In: KDD, pp 66–74
Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: ICLR, https://openreview.net/forum?id=SJU4ayYgl
Li J, Xie T, Liang C et al (2021) Adversarial attack on large scale graph. IEEE Trans Knowl Data Eng 35(1):82–95
Li Y, Jin W, Xu H et al (2020) DeepRobust: A PyTorch library for adversarial attacks and defenses. arXiv preprint arXiv:2005.06149
Liu X, Si S, Zhu J et al (2019) A unified framework for data poisoning attack to graph-based semi-supervised learning. NeurIPS
Ma Y, Wang S, Derr T et al (2019) Attacking graph convolutional networks via rewiring. arXiv preprint arXiv:1906.03750
Miller BA, Çamurcu M, Gomez AJ et al (2019) Improving robustness to attacks against vertex classification. In: MLG Workshop
Miller BA, Shafi Z, Ruml W et al (2023) Attacking shortest paths by cutting edges. ACM Trans Knowl Discov Data 18(2):1–42
Moore J, Neville J (2017) Deep collective inference. In: AAAI, pp 2364–2372
Moosavi-Dezfooli SM, Fawzi A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks. In: CVPR, pp 2574–2582
Mujkanovic F, Geisler S, Günnemann S et al (2022) Are defenses for graph neural networks robust? In: NeurIPS, https://openreview.net/forum?id=yCJVkELVT9d
Neville J, Gallagher B, Eliassi-Rad T (2009) Evaluating statistical tests for within-network classifiers of relational data. In: ICDM, pp 397–406
Palowitch J, Tsitsulin A, Mayer B et al (2022) GraphWorld: fake graphs bring real insights for GNNs. In: KDD, pp 3691–3701
Papernot N, McDaniel P, Jha S et al (2016a) The limitations of deep learning in adversarial settings. In: EuroSP, pp 372–387
Papernot N, McDaniel P, Wu X et al (2016b) Distillation as a defense to adversarial perturbations against deep neural networks. In: SP, pp 582–597
Pareja A, Domeniconi G, Chen J et al (2020) EvolveGCN: evolving graph convolutional networks for dynamic graphs. In: AAAI, pp 5363–5370
Park J, Barabási AL (2007) Distribution of node characteristics in complex networks. Proc Nat Acad Sci 104(46):17916–17920. https://doi.org/10.1073/pnas.0705081104
Prakash A, Moran N, Garber S et al (2018) Deflecting adversarial attacks with pixel deflection. In: CVPR, pp 8571–8580
Sharma K, Trivedi R, Sridhar R et al (2023) Temporal dynamics-aware adversarial attacks on discrete-time dynamic graph models. In: KDD, p 2023–2035
Sun Y, Wang S, Tang X et al (2019) Node injection attacks on graphs via reinforcement learning. arXiv preprint arXiv:1909.06543
Szegedy C, Zaremba W, Sutskever I et al (2014) Intriguing properties of neural networks. In: ICLR
Tsipras D, Santurkar S, Engstrom L et al (2019) Robustness may be at odds with accuracy. In: ICLR
Veličković P, Cucurull G, Casanova A et al (2018) Graph attention networks. In: ICLR
Wong E, Kolter Z (2018) Provable defenses against adversarial examples via the convex outer adversarial polytope. In: ICML, pp 5286–5295

Wu F, Souza A, Zhang T et al (2019a) Simplifying graph convolutional networks. In: ICML, pp 6861–6871
Wu H, Wang C, Tyshetskiy Y et al (2019b) Adversarial examples for graph data: deep insights into attack and defense. In: IJCAI, pp 4816–4823
Xie C, Wang J, Zhang Z et al (2018) Mitigating adversarial effects through randomization. In: ICLR
Xu K, Chen H, Liu S et al (2019) Topology attack and defense for graph neural networks: an optimization perspective. In: IJCAI, pp 3961–3967
Yu S, Vorobeychik Y, Alfeld S (2018) Adversarial classification on social networks. In: AAMAS, pp 211–219
Zhang X, Zitnik M (2020) GNNGuard: Defending graph neural networks against adversarial attacks. In: NeurIPS, pp 9263–9275
Zhu J, Rossi RA, Rao AB et al (2021) Graph neural networks with heterophily. In: AAAI, pp 11168–11176
Zhu J, Yan Y, Zhao L et al (2020) Beyond homophily in graph neural networks: Current limitations and effective designs. In: NeurIPS, pp 7793–7804
Zhu D, Zhang Z, Cui P et al (2019) Robust graph convolutional networks against adversarial attacks. In: KDD, pp 1399–1407
Zügner D, Akbarnejad A, Günnemann S (2018) Adversarial attacks on neural networks for graph data. In: KDD, pp 2847–2856
Zügner D, Günnemann S (2019a) Adversarial attacks on graph neural networks via meta learning. In: ICLR
Zügner D, Günnemann S (2019b) Certifiable robustness and robust training for graph convolutional networks. In: KDD, pp 246–256
Zügner D, Günnemann S (2020) Certifiable robustness of graph convolutional networks under structure perturbations. In: KDD, pp 1656–1665

Publisher's Note