Personalized privacy protection in social networks through adversarial modeling

Sachin Biradar and Elena Zheleva
Department of Computer Science, University of Illinois at Chicago
Chicago, IL, USA
sachinbiradar9@gmail.com, ezheleva@uic.edu

Abstract

In order to personalize user experiences and improve the reach of advertisers, online companies increasingly rely on powerful predictive models to infer user attributes and preferences. Meanwhile, some of these attributes may be of sensitive nature and users have little say in keeping such predicted attribute values as private. Here, we propose a personalized adversarial approach which can empower users to change parts of their online profile in a way that makes it harder for companies to predict their private information. Our work is tailored to state-of-the-art graph convolutional networks which are increasingly used for attribute prediction in graphs. We mitigate their predictive power by helping users decide which relationships and non-sensitive personal attributes they need to change, in order to protect their privacy. Unlike existing adversarial approaches, we consider two constraints that correspond to realistic real-world scenarios: 1) a user has control over changing their own attributes only and no collusion between users can occur, 2) a user is willing to change only certain attributes and relationships but not others. Our approach achieves a much better user privacy protection when compared to state-of-the-art adversarial algorithms for graphs.

Introduction

User-generated content on social media has led to increased interest in mining actionable patterns from user data (Zafarani, Abbasi, and Liu 2014). This content varies from declaring personal attributes, such as name, gender, and education, to sharing ideas, photos, events, and interests. It can be broadly classified into explicit and implicit (Novak and Li 2012). Explicit content is information provided by the user on the social media platform; it can be made publicly available or available to a limited set of users on that platform. In contrast, implicit content is one that the users have chosen to withhold from the platform but the platform can infer based on explicit user content and the content in their social networks. Advancements in data mining techniques have led to the development of powerful models that can predict implicit user information with good accuracy. Predicting implicit user information has many applications, from advertising and recommender systems (Ruining He 2018) to understanding health-related behavior (Sun et al. 2019a).

While leading to a better and more personalized user experience and improved reach for advertisers, implicit attribute prediction has also caused concerns regarding the privacy of users who participate in social networks (Lee Rainie 2018). Multiple studies have shown that it is possible to reliably infer sensitive information from publicly available data (Zheleva and Getoor 2009; Lindamood et al. 2009; Kosinski, Stillwell, and Graepel 2013; Garcia 2017), leading to sensitive attribute disclosure (Zheleva, Terzi, and Getoor 2012). In the absence of rules and guidelines on how to ethically perform such predictions, or how much of it is even acceptable, individuals, companies and government entities can apply machine learning models on public data for classification and prediction of user behaviour without the explicit permission of the users themselves. Solutions such as allowing users to hide their own information have only been successful to a limited extent, due to the fact that these models also take into account the data generated by other users in the network. A natural question one can ask is whether users have any power over protecting their own private information from being predicted accurately.

We propose a user-centric solution that can help users have more control over the prediction of their sensitive attributes. We take an adversarial learning approach and devise an algorithm, Outwit, which can suggest to users what explicit content to change on their social network profiles, in order to mislead predictive algorithms. Unlike a typical adversarial scenario in which the adversary is the person trying to mislead the predictive algorithm, we assume that the predictive algorithm itself is the adversary because it is trying to infer a sensitive attribute. We focus on graph data and assume that the predictive algorithm uses graph convolutional networks (GCN) for sensitive attribute prediction.

Outwit's goal is to mislead the predictive algorithm and help a user protect their privacy. As such, Outwit falls under the umbrella of evasion attacks in which contaminated test data is introduced, in Outwit's case the perturbed data of an individual profile. Unlike existing evasion attacks for GCN (Dai et al. 2018b; Wang and Gong 2019) which require the content of multiple profiles to be perturbed, we consider realistic constraints which assume that 1) a user has control over changing their own content only and not the content of other users and 2) that they are willing to change some attributes and relationships but not others.

Algorithm	Type	Model knowledge	Perturbations	User-centric	Target Model
(Zügner et al. 2018)	Poisoning	Black Box	Feature & Edge	No	GCN
(Bojchevski et al. 2019)	Poisoning	White Box	Edge	No	DeepWalk
(Sun et al. 2019b)	Poisoning	Black Box	Edge	No	GCN
(Chang et al. 2019b)	Poisoning	Black Box	Edge	No	DeepWalk, LINE & GCN
(Dai et al. 2018a)	Evasion	Black-box/White-box	Edge	No	GCN
(Wang and Gong 2019)	Evasion	White Box	Edge	No	LinLBP
Outwit	Evasion	**Grey-box**	Feature & Edge	**Yes**	GCN

Table 1: Comparison between Outwit and other adversarial algorithms for node classification.

Our main contributions are:

Framing a new privacy problem of user-centric adversarial perturbations with utility constraints.

Proposing a realistic "grey-box" scenario in which the model type of the predictive algorithm is known and the algorithm parameters can be estimated using publicly available data.

Developing a new gradient-based algorithm which finds the minimum number of node attribute and edge changes necessary to get the sensitive attribute misclassified while satisfying the utility constraints.

Showing empirically that our solution leads to 8 25% absolute accuracy decrease over a state-of-the-art adversarial algorithm with as few as 6 edge deletions.

Related work

Friendship links and group affiliations can leak a surprisingly large amount of sensitive data about individuals (Zheleva and Getoor 2009; Lindamood et al. 2009; Kosinski, Stillwell, and Graepel 2013) There has been an increasing interest in personalized privacy assistants that can help users manage their privacy online (e.g., (Fang and LeFevre 2010; Ghazinour, Matwin, and Sokolova 2013; Wisniewski, Knijnenburg, and Lipford 2017)). Meanwhile, the use of graph neural networks has increased in popularity recently (Gori, Monfardini, and Scarselli 2005; Scarselli et al. 2005, 2009a,b; Li et al. 2016), posing threats to personal privacy. However, none of the personalized privacy assistant studies consider user-centric adversarial approaches to combat the predictive power of neural network algorithms, as we do in this work.

The goal of adversarial attacks is to mislead machine learning models by either introducing misleading examples during the testing phase (evasion attack)(Dai et al. 2018a; Wang and Gong 2019), contaminating the training data with malicious examples (poisoning attack)(Zügner, Akbarnejad, and Günnemann 2018; Sun et al. 2019b, 2018; Chang et al. 2019b) or by learning about the model it is trying to mislead as much as possible (exploratory attack) (Biggio, Fumera, and Roli 2014). Adversarial learning in graphs is fairly new (Zügner, Akbarnejad, and Günnemann 2018; Dai et al. 2018a; Sun et al. 2018). Dai et al. (Dai et al. 2018a) demonstrate a black-box evasion attack assuming GCN (Kipf and Welling 2017) as the node classification model. The model is allowed to add or delete edges from the graph but node features remain unchanged. Dai et al. also propose a gradient-based white-box attack algorithm which is closest to our work and we use as a baseline. In contrast to their greedy combinatorial approach, we use the magnitude of the gradients along with the degree of the nodes to find optimal perturbations. Zügner et al. (Zügner, Akbarnejad, and Günnemann 2018) focus on poisoning attacks, also using GCN (Kipf and Welling 2017) as the target model but without preserving its non-linearity. The attacker is allowed to manipulate the graph structure as well as the node features while preserving important graph characteristics (degree distribution and feature co-occurrences). Sun et al. (Sun et al. 2018) present a poisoning attack against unsupervised node embedding methods. The goal is to either change the similarity score of a target node pair, or to affect the link prediction accuracy of a test set by adding or deleting edges. Wang et al. (Wang and Gong 2019) optimizes attack targeting Linearized Loopy Belief classifier in order to evade detection via manipulating the graph structure resulting in increase of False Negative Rate. Table 1 summarizes the differences between these approaches and ours. To the best of our knowledge, we are the first ones to consider feature perturbations for evasion attacks and to use graph derivative mapping between node features and class labels for determining feature perturbations. While other white box evasion models (Dai et al. 2018b) greedily add and delete edges based on the gradient information, we use the gradient information to first find the dominant nodes in the graph and then make changes to the target node's edges.

Preliminaries

Data model

We represent the social network as an undirected graph. The users in the network are represented by nodes and their profile information is represented by the attributes or features of the nodes. Relationships, such as friendships or following relationships, are represented by edges between the users. Formally, let $G = (A; X)$ be an attributed undirected graph where $A \ 2 \ f0; 1g^{N \ N}$ is the symmetric binary adjacency matrix representing the relationships between the users, $X \ 2 \ f0; 1g^{N \ D}$ represents the binary nodes' features, N represents the number of users and D represents the number of node features. We denote the D-dimensional feature vector of node i as $X_i \ 2 \ f0; 1g^D$. We assume the node ID to be $\mathcal{N} = f1; 2; :::; Ng$ and feature-ids to be $f1; 2; :::; Dg$.

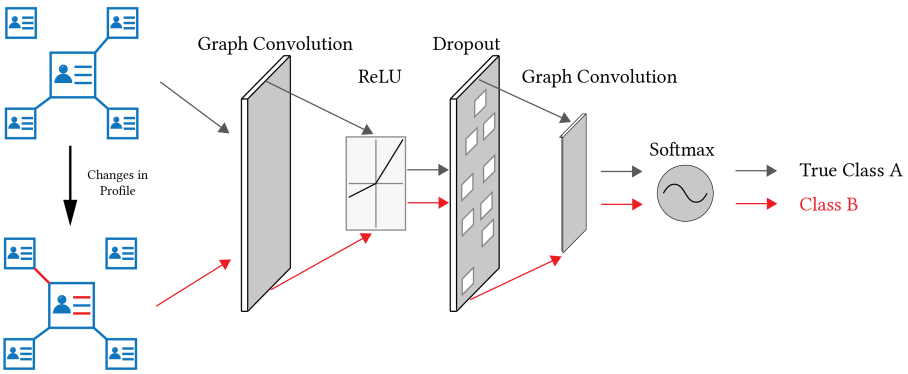

Figure 1: A user in social network from class A changes some parts of his profile to keep his class label private, successfully gets misclassified to class B

Target classifier

We assume that there is a target classifier that is used to predict missing (and potentially sensitive) attributes on user profiles, in order to target users based on hidden characteristics. This classifier can be built by the company behind the social network or third parties with access to the social network data. The target classifier performs node classification and it considers a partially labeled graph $G_L = (A; X; Y_L)$, where Y_L are nodes with explicit labels. If there are C classes, we assume the class labels to be $\mathcal{C} = f1; 2; :::; Cg$. Given this partially labelled graph G_L, the task of the target classifier is to find an optimal function $f : \mathcal{N} \; ! \; \mathcal{C}$ which maps each node $i \; 2 \; \mathcal{N}$ to a single class in $\mathcal{C}$ in order to correctly classify the unlabeled nodes. Since we have the complete A and X, the features and edges of unlabelled nodes (test nodes) in the graph G_L are known and are part of the training data hence this corresponds to a transductive learning scenario (Chapelle, Scholkopf, and Zien 2009).

Graph Convolutional Networks (GCNs) have achieved state-of-the-art results on node classification tasks hence we assume that the target classifier is based on GCN by Kipf and Welling (2016) (Kipf and Welling 2017). GCNs use spectral graph convolutions in which filter parameters are shared over all locations in the graph. A single layer in this model is a non linear function of the form:

$$H^{(k+1)} = \; (H^{(k)}; \hat{A}; W) \tag{1}$$

where $H^{(0)} = X$, K is the number of layers and W is trainable weight matrix. The model uses ReLU as the non-linear activation function; the adjacency matrix A is normalized for faster computation. The final model is a 2-layer neural network with the following propagation rule -

$$Z = f(X; A) = softmax(\hat{A} \; ReLU(\hat{A} \; X \; W^{(0)}) \; W^{(1)}) \tag{2}$$

where $W^{(0)} \; 2 \; R^{D \;\; H}$ is an input-to-hidden weight matrix for a hidden layer with H feature maps. $W^{(1)} \; 2 \; R^{H \;\; C}$ is a hidden-to-output weight matrix. The softmax activation function, defined as $softmax(h_i) = \frac{1}{Z} exp(h_i)$ with $Z = \sum_i exp(h_i)$, is applied row-wise. To optimize node classification, cross-entropy error is used as loss over all labeled examples -

$$L = \sum_{Y \; 2 \; Y_L} \sum_{c2\mathcal{C}} Y_c \ln Z_c \tag{3}$$

Information available to the adversarial algorithm

The goal of our adversarial algorithm is to perturb the graph attributes of a focus node in a way that makes the target classifier predict poorly the node's sensitive attribute. There are two types of information that the adversarial algorithm has access to: 1) information about the target classifier, and 2) user attributes and edges that adversarial algorithm has the power to perturb.

Adversarial attacks can be classified into two main groups based on the amount of target classifier information accessible to them:

White-box attack: The algorithm has access to the target classifier predictions and to its model parameters (Dai et al. 2018a; Sun et al. 2018).

Black-box attack: The algorithm has access only to the predictions of the target classifier (Chang et al. 2019a; Zügner, Akbarnejad, and Günnemann 2018; Dai et al. 2018a; Bojchevski and Günnemann 2019).

Here we introduce a third type of attack which we call *Grey-box attack*. In a grey-box attack, the adversarial algorithm does not have access to the actual target model but has partial access to the training nodes of the target classifier and their true labels. Therefore, it can estimate the parameters of the target model based on this data and make node label predictions using the estimated model. Grey-box attacks are more practical than white box attacks because the adversarial algorithm is unlikely to have access to the target classifier parameters but can estimate them based on publicly available data. At the same time, black box attacks are rather limited in their capabilities and are known to perform much worse than white-box attacks (Dai et al. 2018a).

Our work assumes a grey-box attack and that the adversarial algorithm has access to true class labels for *some* nodes in the graph to train the estimated classifier. We also make the practical assumption that the only information that the adversarial algorithm can change is the focus node's own

attributes and friendship links. This user-centric approach is one of the main distinguishing characteristics of our setup as compared to current adversarial algorithms on graphs which assume that they can perturb multiple nodes' profiles at the same time (Zügner, Akbarnejad, and Günnemann 2018; Dai et al. 2018a; Bojchevski and Günnemann 2019; Chang et al. 2019a; Sun et al. 2018; Zügner and Günnemann 2019; Wang and Gong 2019; Sun et al. 2019b; Chang et al. 2019b).

Privacy protection problem

Adversarial Perturbations

The task of the adversarial algorithm is to modify the profile information and relationships of a target user t in the social network such that the target classifier f cannot predict the true class label of t. Since f's parameters are unknown, we estimate a classifier $\hat{f}$ trained on a partially labelled social network graph G_L. Let $\tilde{G}$ be the graph after the adversarial algorithm is applied, $X_{td} \in \{0,1\}$ and $\tilde{X}_{td} \in \{0,1\}$ be the values of feature d for target node t before and after the perturbation, $A_{ti} \in \{0,1\}$ be a binary value representing whether an edge exists between node t and node i, and let the constants $\Delta, \Gamma \in \mathbb{Z}_{\geq 0}$ represent the maximum number of changes that are allowed to be made to the feature vector and edges respectively for node t. Given the classifier $\hat{f}$ and (G_t, Y_t) where G_t is the graph visible to the target node t and Y_t is the true class of t, the task of the adversarial algorithm $g : G \rightarrow \tilde{G}$ is to modify the graph $G = (A, X)$ to $\tilde{G} = (\tilde{A}, \tilde{X})$ such that:

$$\begin{aligned} \max_{\tilde{G}} \; & P(\hat{f}(\tilde{G}, t) = Y_t) \\ s.t. \; & \tilde{G} = g(\hat{f}, (G_t, Y_t)) \\ & \sum_{d=1}^{D} |X_{td} - \tilde{X}_{td}| \leq \Delta \\ & \sum_{i=1}^{i=N} |A_{ti} - \tilde{A}_{ti}| \leq \Gamma \end{aligned} \tag{4}$$

The constraints ensure that the adversarial algorithm does not drastically modify the target node and that the changes it makes are unnoticeable. These perturbations will allow a social network user to change their profile and safeguard their sensitive attributes from machine learning predictions.

The computational complexity of training the estimated model and finding the features and nodes that are most helpful in prediction is $O(EHDC)$ (Kipf and Welling 2017) where $E = \sum_{i=1;j=1}^{N} A_{i;j} > 0$ is the number of graph edges. The computational complexity to find edge perturbations is $O(E_t + D)$ where $E_t$is the degree of target node and the complexity to find the optimal feature perturbation is $O(D)$. Hence the complexity to find both feature and edge perturbations for a target user is $O(E_t + D)$.

Utility-constrained Perturbations

There is a utility cost associated with hiding or changing attributes on social media. For example, a user may not be willing to declare a different gender from their real gender online or unfriend their spouse, in order to impact the accuracy of predicting another, sensitive attribute. A user may be more willing to agree to change favorite foods and movies or to "unfriend" an elementary school classmate. Therefore, there is a cost to changing some attribute and our algorithm needs to take this cost into consideration. We define user-specific feature utilities as $U \in \mathbb{R}^{N \times D}$ such that a high utility $U_{i;d}$ corresponds to user i being less willing to change feature X_{id}.

In order to account for feature utilities in deciding which features to change, we add a utility constraint to our problem. The utility-constrained adversarial algorithm is allowed to change only those attributes whose user utilities are below a certain threshold. The following is the final optimization problem with utility constraints:

$$\begin{aligned} \max_{\tilde{G}} \; & P(\hat{f}(\tilde{G}, t) = Y_t) \\ s.t. \; & \tilde{G} = g(\hat{f}, (G_t, Y_t)) \\ & \forall t, d : U_{td} \geq \mu \Rightarrow \tilde{X}_{td} = X_{td} \\ & \forall i, j : V_{ij} \geq \nu \Rightarrow \tilde{A}_{ij} = A_{ij} \\ & \sum_{d=1}^{D} |X_{td} - \tilde{X}_{td}| \leq \Delta \\ & \sum_{i=1}^{i=N} |A_{ti} - \tilde{A}_{ti}| \leq \Gamma \end{aligned} \tag{5}$$

where U_{td} is the utility of the feature d for node t, V_{ij} is the utility of the edge between node i and j, μ and ν are the thresholds below which the features and edge are allowed to be modified respectively.

Adversarial, user-centric privacy protection

Here, we present Outwit, our algorithm for solving the privacy protection problem through adversarial perturbations, targeting Graph Convolutional Networks. Given a graph G and a target classifier $\hat{f}$, our goal is to protect the privacy of a target user t by performing a small number of perturbations on their profile leading to the graph $\tilde{G}$, such that t gets misclassified by $\hat{f}$. Outwit has two types of perturbations: **feature perturbations** and **edge perturbations** that we describe next. To the best of our knowledge, we are the first ones to consider feature perturbations for evasion attacks and to use graph derivative mapping between node features and class labels for determining feature perturbations. As far as novelty in the edge perturbations, other white box evasion models (Dai et al. 2018b) greedily add and delete edges based on the gradient information but we use the gradient information to first find the dominant nodes in the graph and then make changes to the target node's edges.

Feature perturbations

Given the estimated node classifier $\hat{f}$, Outwit finds the the node's attributes most likely to cause the change to the prediction in the target classifier. In order to do so, it looks at

$\frac{dL}{dW^{(0)}}$, the gradient of the loss in the *final* layer with respect to the weights in the *first* layer:

$$\frac{dL}{dW^{(0)}} = \frac{dL}{dH^{(0)}} \quad \frac{dH^{(0)}}{dW^{(0)}} = \mathrm{R}^{D \quad H} \tag{6}$$

Outwit feeds feature matrix X as the input in the first layer hence the output of the first layer $H^{(0)}$ is $\hat{A} \; X \; W^{(0)}$. Because weight vector $W^{(0)}$ is being multiplied with X this gradient indicates the importance given to different features in the nodes. There are H spectral filters in the first hidden layer of our GCN i.e. $W^{(0)} = \mathrm{R}^{N \quad D}$, hence there are H gradient vectors, each associated with one of the filter parameters. Each of the spectral filters focuses on specific features to pass to the next GCN layer. Outwit makes a list of all such features in decreasing order of the magnitude of their gradients and select the top features. Then it finds the association of these features with one of the class labels.

For each node in the labeled set Outwit finds the distribution of class labels with respect to the feature. For all such features in our final list, it finds one class or at most a combination of two classes to have a much higher frequency than the remaining classes.

$$d \; =) \quad c \; j \; c = argmax(\sum_{i=1}^{N} Y_i; where X_{i;d} = X) \tag{7}$$

where X is a particular feature. Then Outwit creates a mapping for the classes and the features associated with them.

Given a target node, the top features associated with the true class of this node which are visible to the classifier $\hat{f}$ are set to 0. Then Outwit uses the list of mappings between features and the classes to select specific features from these mappings that are associated with a class other than the target user's true class. These features are changed to 1 if they were 0 before, until reaching perturbations, the upper limit of allowed feature perturbations.

Edge perturbations

Edges between nodes in a graph play a prominent role in node classification. The target GCN model can predict the target node's label with considerable accuracy even when none of the features from the test set are revealed to it (refer to Table 4 in Section 29) by using labels from the neighbouring labeled nodes. It is thus important to find these dominant neighboring users in the target user's neighbourhood and remove the connections to them, in order to decrease the confidence of the classifier towards the target node's true class label. To this end, Outwit finds

$$\frac{dL}{d(edges)} = \frac{dL}{dA} = \frac{dL}{dH^{(0)}} \quad \frac{dH^{(0)}}{dA} = \mathrm{R}^{N \quad N} \tag{8}$$

the gradient of the loss in the final layer with respect to edges represented by A, the adjacency matrix. Some of the edges have a larger gradient magnitude than others which suggests that these edges provide significant information to their neighbouring node in the aggregation process of graph convolution. Outwit orders all node pairs (edges) in decreasing order of their magnitude of gradient. The users represented by nodes with a significantly large frequency of occurrence are the best classification contributors in the graph, they pass their class label information to their neighbouring nodes. Outwit maintains a mapping of all classes to these dominant nodes. Outwit scans all the edges of the target user in the order of decreasing gradient and connections to the top nodes that have the same class as the target node's true class are removed.

It is also possible to connect the target node to dominant users in G_L, in order to increase the confidence of the node classifier towards the class label of these dominant nodes. Outwit finds the dominant users from the class other than the target user's true class and adds edges between the target node and these dominant nodes until it reaches the constraint of maximum edge perturbations . Dependent on the platform, edge addition perturbations can be harder to achieve than edge removals. For example, it is a lot easier to add a link to someone when reciprocity of relationship is not required (e.g., follower relationship on Twitter) versus when it is required (e.g., friendship relationship on Facebook).

Utility-constrained Outwit Algorithm

To protect privacy, Outwit perturbs information on user's profile as described in the previous subsections. The input to the Outwit algorithm is a partially labeled graph G_L, a target node t and its true class label Y_t. The output is G such that the target user profile information is perturbed. Before it begins, it trains the GCN classifier $\hat{f}$ according to Equation 2 to obtain trained weights $W^{(0)}$ and $W^{(1)}$. Algorithm 1 shows the pseudo-code for Outwit which first looks for features to perturb (Lines 8 18). It finds the gradient of loss L of $\hat{f}$ with respect to $W^{(0)}$. It looks for the most important feature in every spectral filter $h \; 2 \; H$ and stores them in the $features$ variable (lines 10 12). These features represent the most important attributes of the user profile as these help the classifier to identify the node's class. But what is the relation between these features and different classes? Which feature combinations help the classifier to predict a certain class? Lines 13 15 try to find exactly this. For each of the important features in the variable $features$, Outwit maps them to a class in $feature_{map}$ variable. Now that Outwit has the most important node features and their relation to the class labels, it changes the feature to decreases the confidence of node classifier prediction towards y and increase the confidence towards other classes for our target user node (lines 17 18). Then it finds edge perturbation for the target user t (lines 20 27), starting by finding the gradient of loss L with respect to A, adjacency matrix. Outwit finds the users which are most dominant in passing their class labels to their neighbours in the GCN aggregation phase (lines 20 25). Now that it has the dominant nodes in the graph, Outwit removes the edges to dominant nodes whose class label is same as target node's class y and add edges to dominant nodes from other class in (lines 26 27). This, combined with the feature perturbation, will outwit the target classifier

into misclassifying the target user.

Algorithm 1: Outwit Algorithm

Input:
Graph $G_L = (A, X, Y_L)$
Target node t and it's true class Y_t
Utility U
Output:
Perturbed graph $G = (\tilde{A}, \tilde{X})$

Train $\hat{f}$ on G_L to obtain $W^{(0)}$ and $W^{(1)}$

/* Feature Perturbations */
$features = []$
$gradient_W = \frac{dL}{dW^{(0)}}$
foreach $h \in H$ **do**
  if $U[t][d]$ **then**
    $features$ $sort\_descending(d; key = gradient_W[h][d])$
$feature\_map = []$
foreach d **do**
  $distribution$ = class_distribution(d)
  $feature\_map$ (d, index(max($distribution$)))
Set top features of t to 0 from $feature\_map$ where class $= y$
Set top features of t to 1 from $feature\_map$ where class $\neq y$

/* Edge Perturbations */
$gradient_A = \frac{dL}{dA}$
$edge\_grads$ $sort\_descending(e; key = gradient_A)$
$node\_frequency$ []
foreach $(n_1, n_2) \in edge\_grads$ **do**
  $node\_frequency$ $frequency(n_1, n_2)$
$sort(node\_frequency)$
Remove top edges from G such that $t \in (,)$ and the node class $= y$ // Equation 4
Add top edges to G such that $t \in (,)$ and the node class $\neq y$

Return G

Experiments

Datasets

We consider three citation network datasets, Citeseer (Lu and Getoor 2003), Cora (Lu and Getoor 2003), and Pubmed (Sen et al. 2008), together with one Twitter retweet network (Ribeiro et al. 2018). Dataset statistics are summarized in Table 2. We treat each scientific paper as a node and the citation links as (undirected) edges between them. The Cora and CiteSeer datasets consists of sparse 0/1-valued bag-of-words feature vectors for each document indicating the absence/presence of the corresponding word in the document and a list of citation links between them. Each publication in the the Pubmed diabetes dataset is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words. The Twitter node features are spaCy's off-the-shelf 300-dimensional GloVe's vector along with few profile attributes and manual annotation whether a user is hateful or not. The features are averaged across all words in a given tweet, and subsequently, across all tweets a user has. Because these features are not binary, we apply only edge perturbations to this dataset and not feature ones.

Dataset	Nodes	Edges	Features	Classes
Twitter	4971	15,142	320	2
Citeseer	3,327	4,732	3,703	6
Cora	2,708	5,429	1,433	7
Pubmed	19,717	44,338	500	3

Table 2: Dataset statistics.

Experimental setup

Our target model for the node classification task is a Graph Convolutional Network from (Kipf and Welling 2017). The model is trained with a 2-layer GCN. The model hyperparameters such as number of epochs, layers and hidden units, learning rate and dropout rate are kept the same as in (Kipf and Welling 2017).

We show results of the Outwit algorithm with equal utility and various settings of and ranging from 6 feature perturbations, 2 edge perturbations to 20 feature perturbation, 8 edge perturbations. To fairly compare with the baseline algorithms, we also run experiments with edge perturbations only with varying amount of from 2 to 8. We test our algorithm with varying utility distribution with range values for and . As we know that the user has no control over the amount of information that is publicly available to train the model, we run experiments with different ratio of training dataset varying the set of labelled nodes in training set from 10% to 100%.

Baselines

We use Dai et al.'s (Dai et al. 2018a) and Wang et al.'s (Wang and Gong 2019) white-box models as baseline models in our experiments and adapt them to the user-centric, grey-box assumptions of our privacy protection scenario. Because (Dai et al. 2018a) and (Wang and Gong 2019) don't perturb features, a fair comparison requires to show results after edge perturbations only. We also consider simple feature perturbations as baselines, such as setting all the target node features to zeros, to ones and random. For simple edge perturbations we use random edge rewiring to change the links between the nodes. Table 4 shows classification accuracy for the simple perturbations. The row "unperturbed" shows the result for $\hat{f}$. As we can see the average accuracy drops very little for both simple feature and edge perturbations which makes Outwit's task very challenging.

Results

Target node perturbation Figure 2 shows the classification performance of the target model when perturbing one

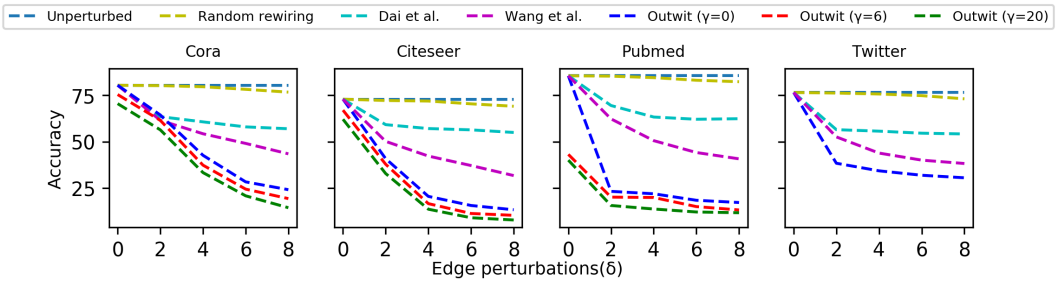

Figure 2: Average node classification accuracy after perturbing test nodes using Outwit algorithm. and represent maximum number of edge and feature perturbations with =2; =2 additions and =2; =2 deletions.

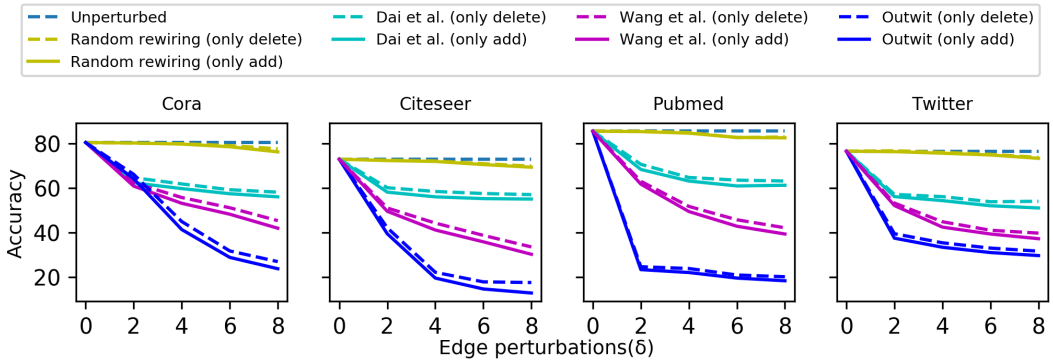

Figure 3: Average node classification accuracy after perturbing test nodes using Outwit algorithm, showing significant decrease (23 47%) in accuracy compared to the baseline when 6 or more edges are deleted.

node at a time. The test nodes are perturbed using the Outwit algorithm with different settings of and . There is a significant decrease in classification accuracy with only 4 edge perturbations and 6 feature perturbations. As expected, the average accuracy drops as we allow more perturbations. With a maximum of just 8 edge perturbations and 10 feature perturbations (0.5% of the total information present in the target node) the average accuracy drastically falls more than 50%. Figure 3 shows the comparison of Outwit with the baseline model with only edge perturbations.

Different ratios of training data Since social networks evolve and change over time, the ratio of publicly available and hidden data can change as well. In order to train the target model, we assume that few of the nodes in the graph are labelled. The ratio of this trained data may vary across different social networks and may also vary over time. So we test the perturbations generated by Outwit algorithm with different ratios of training data. Table 3 shows the performance of the perturbations with varying ratios of labelled nodes. "Estimated" column shows the performance of model which is trained on partial training data and "Target" column shows the performance of the company model trained on 100% of training data. It is seen that the perturbations generated by our algorithm work well for all the ratios of training data and as the previous experiments, we are perturbing one test node at a time, and not their neighbors' information.

Increasing number of colluding users One of the assumptions of our paper is that perturbations are user-centric and Outwit is not allowed to change the attributes of multiple nodes. Here we relax this assumption and show empirically how the average accuracy of the target model is further reduced when a randomly selected group of 10 to 50 users simultaneously use adversarial perturbations from the Outwit algorithm. Figure 4 shows the performance of the target model when many random users simultaneously use 6 edge perturbations and 8 feature perturbations with utility constraints with = = 0:5. The x-axis represent number of users simultaneously adding adversarial information and y-axis represent the average node classification accuracy of GCN. As the number of users increases, the average classification accuracy of the target model decreases even further because adversarial information is passed on to neighbouring nodes in the GCN aggregation phase, causing the target model performance to decrease further.

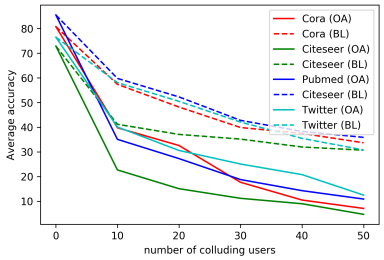

Figure 4: Performance with increasing number of colluding users for Outwit (OA) and the baseline (BL).

Training Ratio	Cora				Citeseer			
	Estimated		Target		Estimated		Target	
	Perturbed	Unperturbed	Perturbed	Unperturbed	Perturbed	Unperturbed	Perturbed	Unperturbed
10%	**19.1%**	70.1%	40.2%	85.5%	**5.8%**	65.1%	38.4%	77.7%
30%	**13.1%**	80.8%	31.7%	85.5%	**6.0%**	72.9%	21.7%	77.7%
50%	**12.8%**	85.3%	22.1%	85.5%	**8.3%**	74.9%	19.4%	77.7%
70%	**15.1%**	86.3%	20.3%	85.5%	**9.6%**	75.8%	15.1%	77.7%
90%	**14.2%**	85.4%	16.8%	85.5%	**10%**	75.8%	12.5%	77.7%
100%	**15.0%**	85.5%	15.0%	85.5%	**9.3%**	77.7%	9.3%	77.7%
Training Ratio	Pubmed				Twitter			
	Estimated		Target		Estimated		Target	
	Perturbed	Unperturbed	Perturbed	Unperturbed	Perturbed	Unperturbed	Perturbed	Unperturbed
10%	**36.3%**	83.2%	53.2%	87.9%	**32.8%**	66.8%	52.7%	76.5%
30%	**38.7%**	85.5%	49.5%	87.9%	**29.6%**	68.9%	44.3%	76.5%
50%	**39.8%**	85.8%	46.9%	87.9%	**27.9%**	70.4%	38.5%	76.5%
70%	**39.7%**	86.8%	43.4%	87.9%	**25.7%**	72.3%	29.9%	76.5%
90%	**40.2%**	86.7%	41.7%	87.9%	**26.8%**	75.7%	27.3%	76.5%
100%	**40.8%**	87.9%	40.8%	87.9%	**25.2%**	76.5%	25.2%	76.5%

Table 3: Average accuracy for varying ratios of labelled training nodes. The "unperturbed" column shows the average accuracy of the learned classifier (estimated or target) on test data and the "perturbed" column shows average accuracy after the Outwit user-centric perturbation.

Strategy		Cora	Citeseer	Pubmed
Unperturbed		80.8%	72.9%	85.5%
Feature	All set to 0	76.6%	67.3%	81.1%
	All set to 1	76.6%	67.2%	81.4%
	Random	76.8%	67.0%	80.3%
Edge	Random rewiring	75.7%	65.1%	78.7%

Table 4: Accuracy for simple feature and edge perturbations

Sensitivity of Outwit to node degree Here we analyze the sensitivity of Outwit to the node degree and whether it performs poorly for low-degree nodes. Figure 5 shows the relation between average accuracy of perturbed nodes to their degree. As expected, Outwit can better perturb the node to preserve privacy with increasing node degree. Nevertheless, Outwit still performs much better than all the baselines as shown earlier.

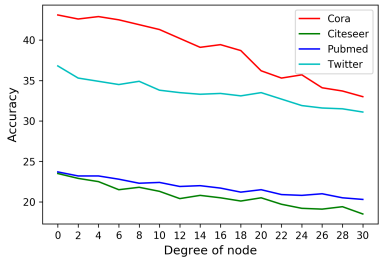

Figure 5: Sensitivity of Outwit to node degree.

Varying Utility In the real world every user has different utility distribution for their profile attributes. In order to replicate the level of mutability of profile attributes we generate U, a utility distribution on top of each dataset. We model the utility values for attributes of a user as random variables from a Bernoulli distribution. Each feature has a different parameter p for their Bernoulli drawn from a Beta distribution prior and . We vary the Beta parameter = 2, = 5(i.e., features tend to have mostly high levels of utility) to = 5, = 2 (i.e., features tend to have mostly low levels of utility p parameters). This allows us to draw a utility value for each user and attribute pair. Figure 6 shows the average classification accuracy under utility constraints. and represent the parameters for the beta distribution used to generate the utility distribution. It is observed that even when utility constraints are applied, the perturbations suggested by algorithm significantly decrease the node classification accuracies when test data is perturbed one node at a time. The accuracies are consistently low for all values of and .

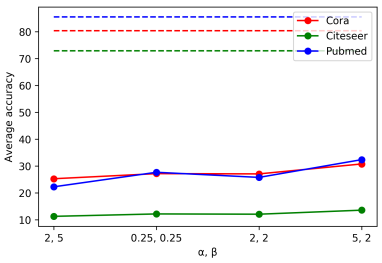

Figure 6: Average accuracy with varying utility. , are the parameters of the beta distribution.

Conclusion

In this work, we presented Outwit, an algorithm which helps social network users find small user-centric adversarial perturbations in their profile which lead to a significant decrease in prediction performance of target classifiers in the form of

graph convolutional networks. Outwit is a viable approach by which users can ensure that their sensitive information cannot be reliably used. Our experiments show that only a few perturbations are required to prevent a target classifier from predicting the true label. Even after applying the utility constraints on the perturbations, the algorithm was able to perform well.

Acknowledgments

This material is based on research sponsored in part by the National Science Foundation under Grant No. 1801644.

References

Biggio, B.; Fumera, G.; and Roli, F. 2014. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering* .

Bojchevski, A.; and Günnemann, S. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 695–704. Long Beach, California, USA: PMLR. URL http://proceedings.mlr.press/v97/bojchevski19a.html.

Chang, H.; Rong, Y.; Xu, T.; Huang, W.; Zhang, H.; Cui, P.; Zhu, W.; and Huang, J. 2019a. The General Black-box Attack Method for Graph Neural Networks.

Chang, H.; Rong, Y.; Xu, T.; Huang, W.; Zhang, H.; Cui, P.; Zhu, W.; and Huang, J. 2019b. A Restricted Black-box Adversarial Framework Towards Attacking Graph Embedding Models.

Chapelle, O.; Scholkopf, B.; and Zien, A. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks* .

Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; and Song, L. 2018a. Adversarial Attack on Graph Structured Data. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR.

Dai, Q.; Li, Q.; Tang, J.; and Wang, D. 2018b. Adversarial Network Embedding. In *AAAI*.

Fang, L.; and LeFevre, K. 2010. Privacy Wizards for Social Networking Sites. In *Proc. WWW*.

Garcia, D. 2017. Leaking privacy and shadow profiles in online social networks. *Science advances* .

Ghazinour, K.; Matwin, S.; and Sokolova, M. 2013. Monitoring and recommending privacy settings in social networks. In *Proc. EDBT*.

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* IEEE.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*.

Kosinski, M.; Stillwell, D.; and Graepel, T. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences* .

Lee Rainie. 2018. Americans' complicated feelings about social media in an era of privacy concerns. URL http://www.pewresearch.org/fact-tank/2018/03/27/americans-complicated-feelings-about-social-media-in-an-era-of-privacy-concerns/.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. S. 2016. Gated Graph Sequence Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016*.

Lindamood, J.; Heatherly, R.; Kantarcioglu, M.; and Thuraisingham, B. 2009. Inferring private information using social network data. In *Proceedings of the 18th international conference on World wide web*. ACM.

Lu, Q.; and Getoor, L. 2003. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*.

Novak, E.; and Li, Q. 2012. A survey of security and privacy in online social networks. *College of William and Mary Computer Science Technical Report* .

Ribeiro, M. H.; Calais, P. H.; Santos, Y. A.; Almeida, V. A.; and Meira Jr, W. 2018. Characterizing and detecting hateful users on twitter. In *Twelfth International AAAI Conference on Web and Social Media*.

Ruining He. 2018. PinSage: A new graph convolutional neural network for web-scale recommender systems. URL https://medium.com/pinterest-engineering/pinsage-a-new-graph-convolutional-neural-network-for-web-scale-recommender-systems-88795a107f48.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009a. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks* .

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009b. The graph neural network model. *IEEE Transactions on Neural Networks* .

Scarselli, F.; Yong, S. L.; Gori, M.; Hagenbuchner, M.; Tsoi, A. C.; and Maggini, M. 2005. Graph neural networks for ranking web pages. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* .

Sun, M.; Tang, J.; Li, H.; Li, B.; Xiao, C.; Chen, Y.; and Song, D. 2018. Data Poisoning Attack against Unsupervised Node Embedding Methods. *arXiv preprint arXiv:1810.12881* .

Sun, M.; Zhao, S.; Gilvary, C.; Elemento, O.; Zhou, J.; and Wang, F. 2019a. Graph convolutional networks for computational drug development and discovery. *Briefings in Bioinformatics* ISSN 1477-4054.

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2019b. Node Injection Attacks on Graphs via Reinforcement Learning.

Wang, B.; and Gong, N. Z. 2019. Attacking Graph-based Classification via Manipulating the Graph Structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, 2023–2040. New York, NY, USA: ACM. ISBN 978-1-4503-6747-9. doi:10.1145/3319535.3354206. URL http://doi.acm.org/10.1145/3319535.3354206.

Wisniewski, P.; Knijnenburg, B. P.; and Lipford, H. R. 2017. Making privacy personal: Profiling social network users to inform privacy education and nudging. *Int. J. Human-Computer Studies* 98.

Zafarani, R.; Abbasi, M. A.; and Liu, H. 2014. *Social Media Mining: An Introduction*. New York, NY, USA: Cambridge University Press. ISBN 1107018854, 9781107018853.

Zheleva, E.; and Getoor, L. 2009. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*. ACM.

Zheleva, E.; Terzi, E.; and Getoor, L. 2012. Privacy in social networks. *Synthesis Lectures on Data Mining and Knowledge Discovery* 3(1): 1–85.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18. New York, NY, USA: ACM. ISBN 978-1-4503-5552-0.

Zügner, D.; and Günnemann, S. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations (ICLR)*.