Adversarial Classification on Social Networks

Sixie Yu
Electrical Engineering and Computer Science, Vanderbilt University
Nashville, TN
sixie.yu@vanderbilt.edu

Yevgeniy Vorobeychik
Electrical Engineering and Computer Science, Vanderbilt University
Nashville, TN
yevgeniy.vorobeychik@vanderbilt.edu

Scott Alfeld
Computer Science, Amherst College
Amherst, MA
salfeld@amherst.edu

ABSTRACT

The spread of unwanted or malicious content through social media has become a major challenge. Traditional examples of this include social network spam, but an important new concern is the propagation of fake news through social media. A common approach for mitigating this problem is by using standard statistical classification to distinguish malicious (e.g., fake news) instances from benign (e.g., actual news stories). However, such an approach ignores the fact that malicious instances propagate through the network, which is consequential both in quantifying consequences (e.g., fake news diffusing through the network), and capturing detection redundancy (bad content can be detected at different nodes). An additional concern is evasion attacks, whereby the generators of malicious instances modify the nature of these to escape detection. We model this problem as a Stackelberg game between the defender who is choosing parameters of the detection model, and an attacker, who is choosing both the node at which to initiate malicious spread, and the nature of malicious entities. We develop a novel bi-level programming approach for this problem, as well as a novel solution approach based on implicit function gradients, and experimentally demonstrate the advantage of our approach over alternatives which ignore network structure.

ACM Reference Format:
Sixie Yu, Yevgeniy Vorobeychik, and Scott Alfeld. 2018. Adversarial Classification on Social Networks. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Consider a large online social network, such as Facebook or Twitter. It enables unprecedented levels of social interaction in the digital space, as well as sharing of valuable information among individuals. It is also a treasure trove of potentially vulnerable individuals to exploit for unscrupulous parties who wish to gain an economic, social, or political advantage. In a perfect world, the social network is an enabler, allowing diffusion of valuable information. We can think of this "benign" information as originating stochastically from some node, and subsequently propagating over the network to its neighbors (e.g., through retweeting a news story), then their neighbors, and so on. But just as the network is a conduit for valueable information, so it is for "malicious" content. However, such undesirable content can be targeted: first, by selecting an influential starting point on the network (akin to influence maximization), and second, by tuning the content for maximal impact. For example, an adversary may craft the headline of a fake news story to capture the most attention. Consider the illustration in Figure 1, where an attacker crafts a fake news story and shares it with Adam. This story is then shared by Adam with his friends, and so on.

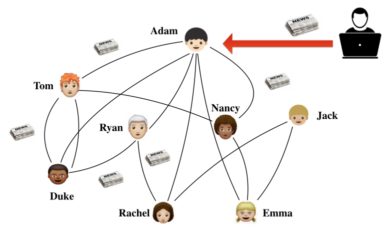

Figure 1: An example of the propagation of malicious contents.

These are not abstract concerns. Recently, widespread malicious content (e.g., fake news, antisocial posts) in online social networks has become a major concern. For example, considering that over 50% adults in the U.S. regard social media as their primary sources for news [17], the negative impact of fake news can be substantial. According to Allcott et al. [1] over 37 million news stories that are later proved fake were shared on Facebook in the last three months of 2016 U.S. presidential election. In addition to fake news, antisocial posts in online communities negatively affect other users and damage community dynamics [9], while social network spam and phish can defraud users and spread malicious software [11].

The managers of online social networks are not powerless against these threats, and can deploy detection methods, such as statistical classifiers, to identify and stop the spread of malicious content. However, such traditional mitigations have not as yet proved adequate. We focus on two of the reasons for this: first, adversaries can tune content to avoid being detected, and second, traditional learning approaches do not account for network structure. The implication of network structure mediating both spread and detection has in turn two consequences: first, we have to account for impact of detection errors in terms of benign or malicious content subsequently propgatating through the network, and second, the fact that we can potentially detect malicious content at multiple nodes on the network creates a degree of redundancy. Consequently, while traditional detection methods use training data to learn a single "global" classifier of malicious and benign content, we show that specializing such learning to network structure, and using *different classifiers at different nodes* can dramatically improve performance.

To address the problem of malicious content detection on social networks, we propose two significant modeling innovations. First, we explicitly model the diffusion process of content over networks

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

as a function of content (or, rather, features thereof). This is a generalization of typical network influence models which abstract away the nature of information being shared. It is also a crucial generalization in our setting, as it allows us to directly model the balancing act by the attacker between increasing social influence and avoiding detection. Second, we consider the problem of designing a collection of *heterogeneous* statistical detectors *which explicitly account for network structure and diffusion* at the level of individual nodes, rather than merely training data of past benign and malicious instances. We formalize the overall problem faced as a Stackelberg game between a defender (manager of the social network) who deploys a collection of heterogeneous detectors, and an attacker who optimally chooses both the starting node for malicious content, and the content itself. This results in a complex bi-level optimization problem, and we introduce a novel technical approach for solving it, first considering a naive model in which the defender knows the node being attacked, which allows us to develop a projected gradient descent approach for solving this restricted problem, and subsequently utilizing this to devise a heuristic algorithm for tackling the original problem. We show that our approach offers a dramatic improvement over both traditional homogeneous statistical detection and a common adversarial classification approach.

Related Work. A number of prior efforts have considered limiting adversarial influence on social networks. Most of these pit two influence maximization players against one another, with both choosing a subset of nodes to maximize the spread of their own influence (blocking the influence of the other). For example, Cerenet et al. [7] consider the problem of blocking a "bad" campaign using a "good" campaign that spreads and thereby neutralizes the "bad" influence. Similarly, Tsai et al. [26] study a zero-sum game between two parties with competing interests in a networked environment, with each party choosing a subset of nodes for initial influence. Vorobeychik et al. [28] considered an influence blocking game in which the defender chooses from a small set of security configurations for each node, while the attacker chooses an initial set of nodes to start a malicious cascade. The main differences between this prior work and ours is that (a) our diffusion process depends on the malicious content in addition to network topology, (b) detection at each node is explicitly accomplished using machine learning techniques, rather than an abstract small set of configurations, and (c) we consider an attacker who, in addition to choosing the starting point of a malicious cascade, chooses the content in part to evade the machine learning-based detectors. The issue of using heterogeneous (personalized) filters was previously studied by Laszka et al. [19], but this work did not consider network structure or adversarial evasion.

Our paper is also related to prior research in single-agent influence maximization and adversarial learning. Kempe et al. [18] initiated the study of influence maximization, where the goal is to select a set of nodes to maximize the total subset of network affected for discrete-time diffusion processes. Rodriguez et al. [16] and Du et al. [13–15] considered the continuous-time diffusion process to model information diffusion; we extend this model. Prior adversarial machine learning work, in turn, focuses on the design of a single detector (classifier) that is robust to evasion attacks [6, 12, 20]. However, this work does not consider malicious content spreading over a social network.

2 MODEL

We are interested in protecting a set of agents on a social network from malicious content originating from an external source, while allowing regular (benign) content to diffuse. The social network is represented by a graph $G = (V, E)$, where V is the set of vertices (agents) and E is the set of edges. An edge between a pair of nodes represents communication or influence between them. For example, an edge from i to j may mean that j can see and repost a video or a news article shared by i. For simplicity, we assume that the network is undirected; generalization is direct.

We suppose that each message (benign or malicious) originates from a node on the network (which may differ for different messages) and then propagates to others. We utilize a finite set of instances diffusing over the network (of both malicious and benign content) as a training dataset D. Each instance, malicious or benign, is represented by a feature vector $x \in \mathbb{R}^n$ where n is the dimension of the feature space. The dataset D is partitioned into D^+ and D^-, where D^+ corresponds to malicious and D^- to benign instances.

To analyze the diffusion of benign and malicious content on social networks in the presence of an adversary, we develop formal models of (a) the diffusion process, (b) the defender who aims to prevent the spread of malicious content while allowing benign content diffuse, (c) the attacker who attempts to maximize the influence of a malicious message, and (d) the game between the attacker and defender. We present these next.

2.1 Continuous-Time Diffusion

Given an undirected network with a known topology, we use a continuous-time diffusion process to model the propagation of content (malicious or benign) through the social network, extending Rodriguez et al. [16]. In our model, diffusion will depend not merely on the network structure, but also on the nature of the item propagating through the network, which we quantify by a feature vector x as above.

Suppose that the diffusion process for a single message originates at a node s. First, x is transmitted from s to its direct neighbors. The time taken by a propagation through an edge e is sampled from a distribution over time, $f_e(t; \mathbf{w}_e, x)$, which is a function of the edge itself and the entity x, and parametrized by $\mathbf{w}_e$. The affected (influenced) neighbors of s then propagate x to their neighbors, and so on. We assume that an affected agent remains affected through the diffusion process.

Given a sample of propagation times over all edges, the time t_i taken to affect an agent i is the length of the shortest path between s and i, where the weights of edges are propagation times associated with these edges. The continuous-time diffusion process is supplied with a time window T, which is used to simulate time-sensitive natures of propagation, for example, people are generally concerned about a news for several months but not for years. An agent is affected if and only if its shortest path to s is less than or equal to T. The diffusion process terminates when the path from s to every unaffected agent is above T. We define the influence $\sigma(s, x)$ of an instance x initially affecting a network node s as the expected number of affected agents over a fixed time window T.

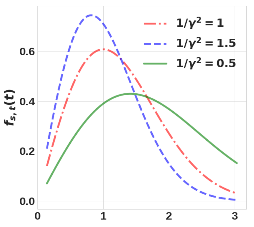

Figure 2: Rayleigh distributions with different $1/\gamma^2$.

We assume that the distributions associated with edges are Rayleigh distributions (illustrated in Figure 2), which have density function $f(t;\gamma) = \frac{t}{\gamma^2} e^{-t^2/(2\gamma^2)}$, where $t \geq 0$ and γ is the scale parameter.[1] The Rayleigh distribution is commonly used in epidemiology and survival analysis [29] and has been recently applied to model information diffusion in social networks [14, 16]. In order to account for heterogeneity among mutual interactions of agents, and to let the influence of a process depend on the content being diffused, we parameterize the Rayleigh distribution of each edge by letting $1/\gamma^2 = \mathbf{w}^T x$, where $\mathbf{w}$ is sampled from the uniform distribution over $[0, 1]$. This parameterization results in the following density function for an arbitrary edge:

$$f_e(t; \mathbf{w}_e, x) = t(\mathbf{w}_e{}^T x) e^{-\frac{1}{2} t^2 (\mathbf{w}_e{}^T x)}. \tag{1}$$

We denote by $\mathcal{W} = \{\mathbf{w}_e | \forall e \in E\}$ the joint parametrization of all edges.

Throughout, we assume that the parameters $\mathcal{W}$ are given, and known to both the defender and attacker. A number of other research studies explore how to learn these parameter vectors from data [14, 15].

2.2 Defender Model

To protect the network against the spread of malicious content, the network manager—henceforth, *the defender*—can deploy statistical detection, which considers a particular instance (e.g., a tweet with an embedded link) and decides whether or not it is safe. The traditional way of deploying such a system is to take the dataset D of labeled malicious and benign examples, train a classifier, and use it to detect new malicious content. However, this approach entirely ignores the fact that a social network mediates the spread of *both* malicious and benign entities. Moreover, both the nature (as captured in the feature vector) and the origin of malicious instances are *deliberate decisions by the adversary* aiming to maximize impact (and harm, from the defender's perspective). Our key innovations are (a) to allow heterogeneous parametrization of classifiers deployed at different nodes, and (b) to explicitly consider both diffusion and adversarial manipulation during learning. In combination, this enables us to significantly boost detection effectiveness in social network settings.

Let $\Theta = \{\theta_1, \theta_2, \cdots, \theta_{|V|}\}$ be a vector of parameters of detection models deployed on the network where each $\theta_i \in \Theta$ represents the model used for content shared by node i.[2] We now extend our definition of expected influence to be a function of detector parameters, denoting it by $\sigma(i, \Theta, x)$, since any content x (malicious or benign) starting at node i which is classified as malicious at a node j (not necessarily the same as i) will be blocked from spreading any further.

We define the defender's utility as

$$U_d = \alpha \sum_{x \in D^-} \sum_{i \in V} \sigma(i, \Theta, x) - (1 - \alpha) \sum_{x \in D^+} \sigma(s, \Theta, z(x)), \tag{2}$$

where s is the starting node targeted by the adversary, which is subsequently modified by the same adversary into $z(x)$ (in an attempt to bypass detection) when the original content used by the adversary is x. The first part of the utility represents the influence of benign content that the defender wishes to maximize, while the second part denotes the influence of malicious content that the defender aims to limit, with α trading off the relative importance of these two considerations. Observe that we assume that benign content originates uniformly over the set of nodes, while malicious origin is selected by the adversary. The defender's action space is the set of all possible parameters Θ of the detectors deployed at all network nodes. Note that, as is typical in machine learning, we are using the finite labeled dataset D as a proxy for expected utility with respect to malicious and benign content generated from the same distribution as the data.

2.3 Attacker Model

The attacker's decision is twofold: (1) find a node $s \in V$ to start diffusion; and (2) transform malicious content from x (its original, or ideal, form) into another feature vector $z(x)$ with the aim of avoiding detection. The first decision is reminiscent of the influence maximization problem[18]. The second decision is commonly known as the *evasion attack* on classifiers [20, 23]. In our case, the adversary attempts to balance three considerations: (a) impact, mediated by the diffusion of malicious content, (b) evasion, or avoiding being detected (a critical consideration for impact as well), and (c) a cost of modifying original "ideal" content into another form, which corresponds to the associated reduced effectiveness of the transformed content, or effort involved in the transformation. We impose this last consideration as a hard constraint that $||z(x) - x||_p \leq \epsilon$ for an exogenously specified ϵ, where $|| \cdot ||_p$ is the l_p norm.

Consider the collection of detectors with parameters Θ deployed on the network. We say that a malicious instance is *d*etected at a node i if $\mathbb{1}[\theta_i(x) = 1] = 1$, where $\mathbb{1}(\cdot)$ is the 0-1 indicator function. The optimization problem of the attacker corresponding to an original malicious instance $x \in D^+$ is then:

$$\begin{aligned} \max_{i,z} \quad & \sigma(i, \Theta, z) \\ s.t \quad & ||z - x||_p \leq \epsilon \\ & \mathbb{1}[\theta_j(z) = 1] = 0, \forall j \in V \end{aligned} \tag{3}$$

where the first constraint is the attacker's budget limit, while the second constraint requires that the attack instance z remains undetected. If Problem (3) does not have a feasible solution, the attacker sends the original malicious instance without any modification. Consequently, the pair $(s, z(x))$ in the defender's utility function above are the solutions to Problem (3).

[1]It is straightforward to allow for alternative distributions, such as Weibull.
[2]Below, we focus on θ_i corresponding to detection thresholds as an illustration; generalization is direct.

2.4 Stackelberg Game Formulation

We formally model the interaction between the defender and the attacker as a Stackelberg game in which the defender is the leader (choosing parameters of node-level detectors) and the attacker the follower (choosing a node to start malicious diffusion, as well as the content thereof). We assume that the attacker knows Θ, as well as all relevant parameters (such as $\mathcal{W}$) before constructing its attack. The equilibrium of this game is the joint choice of $(\Theta, s(\Theta), z(x;\Theta))$, where $s(\Theta)$ and $z(x;\Theta)$ solve Problem (3), thereby maximizing the attacker's utility, and Θ maximizes the defender's utility given s and z. More precisely, we aim to find a Strong Stackelberg Equilibrium (SSE), where the attacker breaks ties in the defender's favor.

We propose finding solutions to this Stackelberg game using the following optimization problem:

$$\begin{aligned} \max_{\Theta} \quad & \alpha \sum_{x \in D^-} \sum_i \sigma(i,\Theta,x) - (1-\alpha) \sum_{x \in D^+} \sigma(s,\Theta,z(x)) \\ s.t.: \quad & \forall x \in D^+: \quad (s, z(x)) \in \arg\max_{j,z} \sigma(j,\Theta,z) \\ & \forall x \in D^+: \quad ||z(x) - x||_p \leq \epsilon \\ & \forall x \in D^+: \quad \mathbb{1}[\theta_k(x) = 1] = 0, \forall k \in V \end{aligned} \tag{4}$$

This is a hierarchical optimization problem, where the upper-level optimization corresponds to maximizing the defender's utility. The constraints of the upper-level optimization are called the lower-level optimization, which is the attacker's optimization problem.

The optimization problem (4) is generally intractable for several reasons. First, Problem (4) is a bilevel optimization problem [10], which is hard even when the upper- and lower-level problems are both linear [10]. The second difficulty lies in maximizing $\sigma(i,\Theta,x)$ (the attacker's problem), as the objective function does not have an explicit closed-form expression. In what follows, we develop a principled approach to address these technical challenges.

3 SOLUTION APPROACH

We start by making a strong assumption that the defender *knows* the node being attacked. This will allow us to make considerable progress in transforming the problem into a significantly more tractable form. Subsequently, we relax this assumption, developing an effective heuristic algorithm for computing the SSE of the original problem.

First, we utilize the tree structure of a continuous-time diffusion process to convert (4) into a tractable bilevel optimization. We then collapse the bilevel optimization into a single-level optimization problem by leveraging Karush-Kuhn-Tucker (KKT) [5] conditions. The assumption that the defender knows the node being attacked allows us to solve the resulting single-level optimization problem using projected gradient descent.

3.1 Collapsing the Bilevel Problem

A continuous-time diffusion process proceeds in a breadth-first-search fashion. It starts from an agent i trying to influence each of its neighbors. Then its neighbors try to influence their neighbors, and so on. Notice that once an agent becomes affected, it is no longer affected by others. The main consequence of this propagation process is that it results in a propagation tree rooted at i, with its structure intimately connected to the starting node i. This is where we leverage the assumption that the defender knows the starting node of the attack: in this case, the tree structure can be pre-computed, and fixed for the optimization problem.

We divide the agents traversed by the tree into several layers in terms of their distances to the source, where each layer is indexed by l. Since the structure of the tree depends on i, l is a function of i, $l(i)$. An example of the influence propagation tree is depicted in Figure 3, where the first layer consists of $\{j, k, \cdots, g\}$. The number next to each edge represents the weight sampled from the associated distribution.

We define a matrix $\mathbf{A}_l \in \mathbb{R}^{N_l \times n}$ where N_l is the number of agents in layer l and n is the feature dimension of x. Each row of $\mathbf{A}_l$ corresponds to the parametrization vector $\mathbf{w}$ of an edge in layer l (an edge is in layer l if one of its endpoint is in layer $l-1$ while the other is in layer l; the source is always in layer zero). For example, in Figure 3, $\mathbf{A}_1 = [\mathbf{w}_{ij}^T; \mathbf{w}_{ik}^T; \cdots; \mathbf{w}_{ig}^T]$. The product of $\mathbf{A}_l x$ is a vector in $\mathbb{R}^{N_l}$, where each element corresponds to the parameter $1/\gamma^2$ of an edge in layer l.

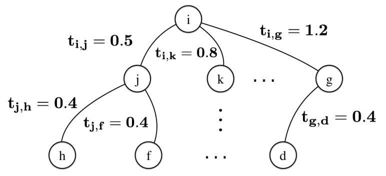

Figure 3: A example continuous-time diffusion process.

Recall that a sample of random variables from Rayleigh distributions associated with edges corresponds to a sample of weights associated with these edges. With a fixed time window T, small edge weights result in wider diffusion of the content over the social network. For example, in Figure 3 if the number next to each edge represents a sample of weights, then with $T = 1$ the propagation starting from i can only reach agents j and k. However, if we assume that in another sample $t_{i,j}, t_{i,k}, t_{i,g}$ all become 0.1, then with the same time window the propagation can reach every agent in the network. Consequently, the attacker's goal is to increase $1/\gamma^2 = \mathbf{w}_e^T x$ for each edge e. This suggests that in order to increase $1/\gamma^2$ the attacker can modify the malicious instance x such that the inner products between x and the parameter vectors $\mathbf{w}_e$ of edges are large. Consequently, we can formulate the attacker's optimization problem with respect to malicious content z for a given original feature vector x as

$$\begin{aligned} \max_{z} \quad & \sum_l k_l \mathbf{1}^T \mathbf{A}_l z \\ s.t. \quad & ||z - x||_p \leq \epsilon \\ & \mathbb{1}[\theta_k(z) = 1] = 0, \forall k \in V. \end{aligned} \tag{5}$$

The attacker aims to make $1/\gamma^2$ for each edge as large as possible, which is captured by the objective function $\mathbf{1}^T \mathbf{A}_l z$, where $\mathbf{1} \in \mathbb{R}^{N_l}$ is a vector with all elements equal to one. Intuitively, this means the attacker is trying to maximize on average the parameter $1/\gamma^2$ of every edge at layer l. Here, $[k_1, k_2, \cdots, k_l]$ is a vector of decreasing coefficients that provides more flexibility to modeling the attacker's behavior: they are used to re-weight the importance of each layer.

For example, setting $k_1 = e^0, k_2 = e^{-1}, \cdots, k_l = e^{-l}$ models the attacker who tries to make malicious instances spread wider at the earlier layers of the diffusion.

We now use similar ideas to convert the upper-level optimization problem of (4) into a more tractable form. Suppose that the node being attacked is s (and known to the defender). Then the defender wants the detection model at j to accurately identify both malicious and benign contents. This is achieved by the two indicator functions inside ① and ② in the reformulated objective function of the defender (6):

$$\max_{\Theta} \underbrace{\alpha \sum_{x \in D^-} \sum_{j} \mathbb{1}[\theta_j(x) = 0] \sum_{l} k_l \mathbf{c}_{l,j}^T \mathbf{A}_l x}_{①} \\ \underbrace{- (1-\alpha) \sum_{x \in D^+} \mathbb{1}[\theta_s(z(x)) = 0] \sum_{l} k_l \mathbf{c}_{l,s}^T \mathbf{A}_l z(x)}_{②} \tag{6}$$

Notice that this expression includes a vector $\mathbf{c}_{l,j} \in \mathbb{R}^{N_l}$ that does not appear in (5). $\mathbf{c}_{l,j}$ is a function of Θ and x, for a given node j which triggers diffusion (which we omit below for clarity):

$$\mathbf{c}_{l,j} = \begin{bmatrix} \mathbb{1}[\theta_{l_1}(x) = 0] \\ \mathbb{1}[\theta_{l_2}(x) = 0] \\ \vdots \\ \mathbb{1}[\theta_{l_{N_l}}(x) = 0]. \end{bmatrix} \tag{7}$$

Slightly abusing notation, we let $l_i, i \in [1, 2, \cdots, N_l]$ denote the ith agent in layer l. The term $k_l \mathbf{c}_{l,j}^T \mathbf{A}_l x$ in ① can be expanded as follows:

$$\begin{aligned} & k_l \mathbf{c}_{l,j}^T \mathbf{A}_l x \\ &= k_l \left[\mathbb{1}[\theta_{l_1}(x) = 0], \ldots, \mathbb{1}[\theta_{l_{N_l}}(x) = 0] \right] \begin{bmatrix} \mathbf{w}_{l_1}^T x \\ \vdots \\ \mathbf{w}_{l_{N_l}}^T x \end{bmatrix} \\ &= k_l \left(\mathbb{1}[\theta_{l_1}(x) = 0] \mathbf{w}_{l_1}^T x + \cdots + \mathbb{1}[\theta_{l_{N_l}}(x) = 0] \mathbf{w}_{l_{N_l}}^T x \right), \end{aligned} \tag{8}$$

noting again that l and N_l depend on j, the starting node of the diffusion process. From the expression (8), the defender tries to find Θ that minimizes the impact of false positives while maximizing the impact of true negatives. This is because if each benign instance $x \in D^-$ is correctly identified (false-positive rates are zero and true-negative rates are one), the summation at the second line of expression (8) will attain its maximum possible value.

In addition to facilitating the propagation of benign contents, the defender wants to limit the propagation of malicious contents, which is embodied in ②. The equations in ② are similar to those in ①, except that the summation is over malicious contents D^+, and ② is accounting for the false negatives. In this case, $\mathbf{c}_{l,s}$ is a function of $z(x)$, the adversarial feature vector which transforms x into another, z.

We now re-formulate the problem (4) as a new bilevel optimization problem (9). The upper-level problem corresponds to the defender's strategy (6), and the lower-level problem to the attacker's optimization problem (5). Here, s is again the node chosen by the attacker.

$$\begin{aligned} \min_{\Theta} \; & (1-\alpha) \sum_{x \in D^+} \mathbb{1}[\theta_s(x) = 0] \sum_{l} k_l \mathbf{c}_{l,s}^T \mathbf{A}_l z(x) \\ & - \alpha \sum_{x \in D^-} \sum_{j} \mathbb{1}[\theta_j(x) = 0] \sum_{l} k_l \mathbf{c}_{l,j}^T \mathbf{A}_l x \\ & s.t: \; \forall x \in D^+ : z(x) \leftarrow \arg\max_{z} \sum_{l} k_l \mathbf{1}^T \mathbf{A}_l z \\ & \quad s.t. \; \forall x \in D^+ : ||z(x) - x||_p \leq \epsilon \\ & \quad \quad \forall x \in D^+ : \mathbb{1}[\theta_k(z(x)) = 1] = 0, \forall k \in V \\ & \quad \quad \forall x \in D^+ : z(x) \geq 0, \end{aligned} \tag{9}$$

where the last constraint ensures that $\mathbf{w}^T z(x) \geq 0$ for all attacks $z(x)$.

The final step, inspired by [24, 25], is to convert (9) into a single-level optimization problem via the KKT [5] conditions of the lower-level problem. With appropriate norm constraints (e.g., l_2 norm) and a convex relaxation of the indicator functions (i.e., convex surrogates of the indicator functions), the lower-level problem of (9) is convex. A convex optimization problem can be equivalently represented by its KKT conditions[8]. The single-level optimization problem then becomes:

$$\begin{aligned} & \min_{\Theta} \quad \hat{F}_d \\ & s.t. \quad \forall x : \\ & \partial_z \left(- \sum_{l} k_l \mathbf{c}_{l,s}^T \mathbf{A}_l z + \lambda g(z, x) + \mu^T h(z, \Theta) - \eta^T z \right) = 0 \\ & \quad \lambda g(z, x) = 0, \lambda \geq 0 \\ & \quad g(z, x) \leq 0 \\ & \quad \eta \odot (-z) = 0, \eta \geq 0 \\ & \quad h(z, \Theta) = 0 \end{aligned} \tag{10}$$

where $\hat{F}_d$ is the objective function of Problem (9), and λ, μ, η are vectors of lagrangian multipliers. $g(z, x) = ||z - x||_p - \epsilon \leq 0$ is the attacker's budget constraint. $h(x, \Theta)$ is the set of equality constraints $\mathbb{1}[\theta_j(z) = 1] = 0, \forall j \in V$. $\eta \odot (-z)$ is the Hadamard (elementwise) product between η and $(-z)$.

3.2 Projected Gradient Descent

In this section we demonstrate how to solve the single-level optimization obtained above by projected gradient descent. The key technical challenge is that we don't have an explicit representation of the gradients with respect to the defender's decision Θ, as these are indirectly related via the optimal solution to the attacker's optimization problem. We derive these gradients based on the implicit function of the defender's utility with respect to Θ.

We begin by outlining the overall iterative projected gradient descent procedure. In iteration t we update the parameters of detection models by taking a projected gradient step:

$$\Theta^{(t+1)} = \text{Proj}_{\mathcal{A}_d} \left(\Theta^{(t)} - \beta_t \nabla_{\Theta} \hat{F}_d \Big|_{\Theta = \Theta^{(t)}} \right) \tag{11}$$

where $\mathcal{A}_d$ is the feasible domain of Θ and β_t is the learning rate. With $\Theta^{(t+1)}$ we solve for $z^{(t+1)}$, which is the optimal attack for a fixed $\Theta^{(t+1)}$. $\nabla_\Theta \hat{F}_d$ is the gradient of the upper-level problem.

Expanding $\nabla_\Theta \hat{F}_d$ using the chain rule and still using s as the initially attacked node, we obtain

$$\nabla_\Theta \hat{F}_d = (1-\alpha)①-\alpha②$$
$$① = \sum_{x\in D^+}\Bigg[\frac{\partial \mathbb{1}[\theta_s(z(x))=0]}{\partial\Theta}\sum_l k_l \mathbf{c}_{l,s}^T \mathbf{A}_l z(x) + \mathbb{1}[\theta_s(z(x))=0]\underbrace{\frac{\partial[\sum_l k_l \mathbf{c}_{l,s}^T \mathbf{A}_l z(x)]}{\partial\Theta}}_{(a)}\Bigg] \tag{12}$$
$$② = \sum_{x\in D^-}\sum_j\Bigg[\frac{\partial \mathbb{1}[\theta_j(x)=0]}{\partial\Theta}\sum_l k_l \mathbf{c}_{l,j}^T \mathbf{A}_l x + \mathbb{1}[\theta_j(x)=0]\underbrace{\frac{\partial[\sum_l k_l \mathbf{c}_{l,j}^T \mathbf{A}_l x]}{\partial\Theta}}_{(b)}\Bigg]$$

In both ① and ② we note that $\frac{\partial \mathbb{1}[\theta_j(x)=0]}{\partial\Theta}$ is dependent on the specific detection models. We will give a concrete example of their derivation in Section 3.5.

In $\sum_l k_l \mathbf{c}_{l,s}^T \mathbf{A}_l z(x)$ there are two terms that are functions of Θ: $\mathbf{c}_{l,s}$ and $z(x)$. Consequently, (a) can be expanded as:

$$(a) = \sum_l k_l\left[\frac{\partial \mathbf{c}_{l,s}}{\partial\Theta_l}\mathbf{A}_l z(x) + \left[\frac{\partial z(x)}{\partial\Theta_l}\right]^T \mathbf{A}_l^T \mathbf{c}_{l,s}\right]. \tag{13}$$

Note that only the detection models of those agents at layer l have contribution to $\mathbf{c}_{l,s}$. Thus, $\frac{\partial \mathbf{c}_{l,s}}{\partial\Theta_l}$ is a Jacobian matrix with dimension $N_l \times N_l$, where N_l is the number of agents at layer l and Θ_l denotes the detection models of those N_l agents. Since $\mathbf{c}_{l,s}$ is also dependent on the specific detection models of agents, we defer its derivation to Section 3.5.

$\frac{\partial z(x)}{\partial\Theta_l}$ is a $n\times N_l$ Jacobian matrix and is the main difficulty because we do not have an explicit function of the attacker's optimal decision $z(x)$ with respect to Θ_l. Fortunately, the constraints in (10) implicitly define $z(x)$ in terms of Θ:

$$\mathbf{f}(\Theta,z,\lambda,\mu,\eta) = \begin{bmatrix} \partial_z\Big(-\sum_l k_l \mathbf{c}_{l,s}^T \mathbf{A}_l z + \lambda g(z,x) + \mu^T h(z,\Theta) - \eta^T z\Big) \\ \lambda g(z,x) \\ \mu^T h(z,\Theta) \\ \eta \odot (-z) \end{bmatrix} \tag{14}$$

Θ and the attacked malicious instance z satisfy $\mathbf{f}(\Theta,z,\lambda,\mu,\eta)=\mathbf{0}$. The Implicit Function Theorem[31] states that if $\mathbf{f}(\Theta,z,\lambda,\mu,\eta)$ is continuous and differentiable and the Jacobian matrix

$$\left[\frac{\partial\mathbf{f}}{\partial z}\Big|\frac{\partial\mathbf{f}}{\partial\lambda}\Big|\frac{\partial\mathbf{f}}{\partial\mu}\Big|\frac{\partial\mathbf{f}}{\partial\eta}\right]$$

has full rank, there is a unique implicit function $I(\Theta)=(z,\lambda,\mu,\eta)$. Moreover, the derivative of $\frac{\partial I}{\partial\Theta}$ is:

$$\frac{\partial I}{\partial\Theta} = -\left[\frac{\partial\mathbf{f}}{\partial z}\Big|\frac{\partial\mathbf{f}}{\partial\lambda}\Big|\frac{\partial\mathbf{f}}{\partial\mu}\Big|\frac{\partial\mathbf{f}}{\partial\eta}\right]^{-1}\left(\frac{\partial\mathbf{f}}{\partial\Theta}\right). \tag{15}$$

$\frac{\partial\mathbf{f}}{\partial z}$ is the Jacobian matrix of $\mathbf{f}(\Theta,z,\lambda,\mu,\eta)$ with respect to z, and so on. $\frac{\partial z}{\partial\Theta}\in\mathbb{R}^{n\times N}$ is the first n rows of $\frac{\partial I}{\partial\Theta}$, where $\frac{\partial z}{\partial\Theta_l}$ can be column-wise indexed by the nodes at layer l.

(b) can be similarly expanded as we had done for (a), except that the attacker does not modify benign content, so that $x\in D^-$ is no longer a function of Θ:

$$(b) = \sum_l\sum_j k_l\left[\frac{\partial\mathbf{c}_{l,j}}{\partial\Theta_l}\mathbf{A}_l x\right]. \tag{16}$$

The full projected gradient descent approach is given by Algorithm 1.

Algorithm 1 Find Defense Strategy

```
1: Input: agent $j$
2: Initialize: $\Theta^{(0)},\lambda,\mu,\eta,\beta_0$
3: for $t = 1\cdots k$ do
4:     $\Theta^{(t+1)} = \mathrm{Proj}_{\mathcal{A}_d}\Big(\Theta^{(t)} - \beta_t \nabla_\Theta \hat{F}_d\big|_{\Theta=\Theta^{(t)}}\Big)$
5: end for
6: return $\Theta^{(k+1)}$
```

3.3 Optimal Attack

So far, we had assumed that the network node being attacked is fixed. However, the ultimate goal is to allow the attacker to choose both the node s, and the modification of the malicious content z. We begin our generalization by first allowing the attacker to optimize these jointly.

The full attacker algorithm which results is described in Algorithm 2.

Algorithm 2 Optimal Attack Strategy

```
1: Input: $\Theta, x$
2: Initialize: $ret$ = []
3: for $i = 1\cdots|V|$ do
4:     $x(i) \leftarrow$ Solve (5)
5:     $\hat{U}_a(i) \leftarrow$ Optimal objective value of (5)
6:     $\big(i, z(i,x), \hat{U}_a(i)\big)$ appended to $ret$
7: end for
8: $z, s \leftarrow$ OptimalTuple(ret)
9: return $z, s$
```

Recall that the tree structure of a propagation is dependent on the agent being attacked, which makes the objective function of (5) a function of the agent being attacked. Thus, for a given fixed Θ, the attacker iterates through each agent i and solves the problem (5), assuming the propagation starts from i, resulting in associated utility $\hat{U}_a(i)$ and an attacked instance $z(i,x)$. Then i, $z(i,x)$, and $\hat{U}_a(i)$ are appended into a list of a 3-tuples (the sixth step in Algorithm 2). When the iteration completes the attacker picks the optimal 3-tuple in terms of utility (eighth step in Algorithm 2, where the function *OptimalTuple(ret)* finds the optimal 3-tuple from the list *ret*). The node s and the corresponding attack instance z in this optimal 3-tuple become the optimal attack.

3.4 SSE Heuristic Algorithm

Now we take the final step, relaxing the assumption that the attacker chooses a fixed node to attack which is known to the defender prior to choosing Θ. Our main observation is that fixing s in the defender's algorithm above allows us to find a collection of heterogeneous detector parameters Θ, and we can evaluate the *actual* utility of the associated defense (i.e., if the attacker optimally chooses both s and z in response) by using Algorithm 2. We use this insight to devise a simple heuristic: iterate over all potential nodes s that can be attacked, compute the associated defense $\Theta(s)$ (using the optimistic definition of defender's utility in which s is assumed fixed), then find the actual optimal attack in response for each $x \in D^+$. Finally, choose the $\Theta(s)$ which has the best *actual* defender utility.

This heuristic algorithm is described in Algorithm 3.

Algorithm 3 Optimal Defense Strategy

```
1: Input: $G = (V, E), \mathcal{W}, D$
2: for $j = 1 \cdots |V|$ do
3:     $\Theta_j \leftarrow$ Apply Algorithm 1
4:     $\forall x \in D^+ : (s, z(x)) \leftarrow$ Apply Algorithm 2
5:     $\hat{U}_d(j) \leftarrow DefenderUtility(\Theta_j, (s, z(x)))$
6: end for
7: $j \leftarrow \arg\max_j \hat{U}_d(j)$
8: return $\Theta_j$
```

The fifth step in the algorithm includes the function *DefenderUtility*, which evaluates the defender's utility $\hat{U}_d(j)$. Note that the input argument s of this function is used to determine the tree structure of the propagation started from s.

Recall that Algorithm 1 solves (10), which depends on the specific detection model to compute the relevant gradients. Therefore, in what follows, we present a concrete example of how to solve (10) where detection models are logistic regressions. Specifically, we illustrate how to derive the two terms, $\frac{\partial \mathbb{1}[\theta_j(z)=0]}{\partial \Theta}$ and $\frac{\partial \mathbf{c}_{l,j}}{\partial \Theta_l}$ that depend on particular details of the detection model.

3.5 Illustration: Logistic Regression

We consider the logistic regression model used for detection at individual nodes to illustrate the ideas developed above. For a node i, its detection model has two components: the logistic regression $\frac{1}{1+e^{-\phi^T x}}$, where ϕ is the weight vector of the logistic regression and x the instance propagated to i, and a detection threshold θ_i (which is the parameter the defender will optimize). An instance is classified as benign if $\frac{1}{1+e^{-\phi^T x}} \le \theta_i$. Thus (slightly abusing notation as before), $\theta_i(x) \ne 0$ (x is classified as malicious) if $\frac{1}{1+e^{-\phi^T x}} \ge \theta_i$.

With the specific forms of the detection models we can derive $\frac{\partial \mathbb{1}[\theta_j(x)=0]}{\partial \Theta}$ and $\frac{\partial \mathbf{c}_l}{\partial \Theta_l}$ (omitting the node index s or j for clarity). A technical challenge is that the indicator function $\mathbb{1}(\cdot)$ is not continuous or differentiable, which means that it's difficult to characterize its derivative with respect to Θ. However, observe that for logistic regression $\theta_j(x) = 0 \left(\frac{1}{1+e^{-\phi^T x}} \le \theta_j\right)$ is equivalent to $\log\left(\frac{\theta_j}{1-\theta_j}\right) \ge \phi^T x$. Therefore we use $\log\left(\frac{\theta_j}{1-\theta_j}\right) - \phi^T x$ as a surrogate function for $\mathbb{1}[\cdot]$. Then $\frac{\partial \mathbb{1}[\theta_j(x)=0]}{\partial \Theta}$ is a N-dimension vector with the jth element equal to $\frac{1}{\theta_j - \theta_j^2}$. The $\mathbf{c}_l$ vector then becomes:

$$\mathbf{c}_l = \begin{bmatrix} \log\left(\frac{\theta_{l_1}}{1-\theta_{l_1}}\right) - \phi^T x \\ \log\left(\frac{\theta_{l_2}}{1-\theta_{l_2}}\right) - \phi^T x \\ \vdots \end{bmatrix} \tag{17}$$

and $\frac{\partial \mathbf{c}_l}{\partial \Theta_l}$ becomes a $N_l \times N_l$ diagonal matrix:

$$\frac{\partial \mathbf{c}_l}{\partial \Theta_l} = \begin{bmatrix} \frac{1}{\theta_{l_1} - \theta_{l_1}^2} & & \\ & \ddots & \\ & & \frac{1}{\theta_{N_l} - \theta_{N_l}^2} \end{bmatrix} \tag{18}$$

With equations (12)-(16), $\frac{\partial \mathbf{c}_l}{\partial \Theta_l}$ and $\frac{\partial \mathbb{1}[\theta_j(x)=0]}{\partial \Theta}$, we can now calculate $\nabla_\Theta \hat{F}_d$. Since the thresholds $\theta_i \in [0,1]$, the defender's action space is $[0,1]^N$. When updating Θ by (11) we therefore project it back to $[0,1]^N$ in each iteration.

4 EXPERIMENTS

In this section we experimentally evaluate our proposed approach. We used the Spam dataset [22] from UCI machine learning repository as the training dataset for the logistic regression model. The Spam dataset contains 4601 emails, where each email is represented by a 57-dimension feature vector. We divided the dataset into three disjoint subsets: D' used to learn the logistic regression (tuning the weight vectors with thresholds setting to 0.5) as well as other models to which we compare, D_{train} used in Algorithm 3 to find the optimal defense strategy, and D_{test} to test the performance of the defense strategy. The sizes of D', D_{train}, and D_{test} are 3681, 460, and 460, respectively. They are all randomly sampled from D.

Our experiments were conducted on two synthetic networks with 64 nodes: Barabasi-Albert preferential attachment networks (BA) [2] and Watts-Strogatz networks (Small-World) [30]. BA is characterized by its power-law degree distribution, where connectivity is heavily skewed towards high-degree nodes. The power-law degree distribution, $P(k) \sim k^{-r}$, gives the probability that a randomly selected node has k neighbors. The degree distributions of many real-world social networks have previously been shown to be reasonably approximated by the power-law distribution with $r \in [2.1, 2.4]$ [3]. Our experiments for BA were conducted across two sets of parameters: $r = 2.1$ and $r = 2.3$.

The Small-World topology is well-known for balancing shortest path distance between pairs of nodes and local clustering in a way as to qualitatively resemble real networks [27]. In our experiments we consider two kinds of Small-World networks. The first has average length of shortest path equal to 5.9 and local clustering coefficient equal to 0.144. In this case the local clustering coefficient is close to what had been observed in large-scale Facebook friendship networks [27]. The second one has average shortest path length of 5 and local clustering coefficient of 0.08, where the local clustering coefficient is close to that for the electric power grid of the western United States [30].

Our node-level detectors use logistic regression, with our algorithm producing the threshold for these. The trade-off parameter α was set to 0.5 and the time window T was set to 1. We applied standard pre-processing techniques to transform each feature to lie between zero and one. The attacker's budget is measured by squared l_2 norm and the budget limit ϵ is varied from 0.001 to 0.01. We compare our strategy with three others based on traditional approaches: *Baseline*, *Re-training*, and *Personalized-single-threshold*; we describe these next.

Baseline: This is the typical approach which simply learns a logistic regression on training data, sets all thresholds to 0.5, and deploys this model at all nodes.

Re-training: The idea of re-training, common in adversarial classification, is to iteratively augment the original training data with attacked instances, re-training the logistic regression each time, until convergence [4, 21]. The logistic regressions deployed at the nodes are homogeneous, with all thresholds being 0.5.

Personalized-single-threshold: This strategy is only allowed to tune a single agent's threshold. It has access to D_{train} that includes unattacked emails. The strategy iterates throught each node i and finds its optimal threshold. The optimality is measured by the defender's utility as defined in (2), where the expected influence of an instance is estimated by simulating 1000 propagations started from i. Then the strategy picks the node with largest utility and sets its optimal threshold.

As stated earlier, network topologies and parameter vectors associated with edges are assumed to be known by both the defender and the attacker. The attacker has full knowledge about the defense strategy, including the weight vectors of logistic regressions as well as their thresholds. As in the definition of Stackelberg game, the evaluation procedure lets the defender first choose its strategy Θ^*, and then the attacker computes its best response, which chooses the initial node for the attack s and transformations of malicious content z aimed at evading the classifier. Finally the defender's utility is calculated by (2), where the expected influence is estimated by simulating 1000 propagations originating from s for each malicious instance z.

The experimental results for BA ($r = 2.1$) and Small-World (average length of shortest path=5.9 and local clustering coefficient=0.144) are shown in Figure 4, and the experimental results for BA ($r = 2.3$) and Small-World (average length of shortest path=5 and local clustering coefficient=0.08) are shown in Figure 5.

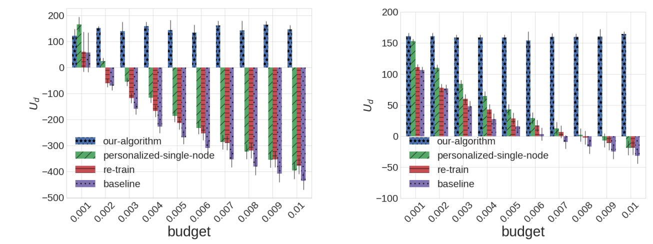

Figure 4: The performance of each defense strategy. Each bar is averaged over 10 random topologies. Left: BA. Right: Small-world)

As we can observe from the experiments, our algorithm outperforms all of the alternatives in nearly every instance; the sole exception is when the attacker budget is 0.001, which effectively eliminates the adversarial component from learning. For larger budgets, our algorithm remarkably robust even as other algorithms perform quite poorly, so that when $\epsilon = 0.01$, there is a rather dramatic

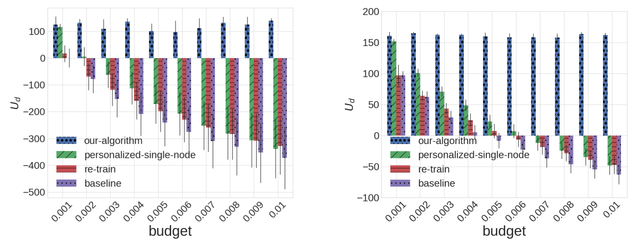

Figure 5: The performance of each defense strategy. Each bar is averaged over 10 random topologies. Left: BA. Right: Small-world)

gap between our approach and all alternatives. Not surprisingly, the most dramatic differences can be observed in the BA topology: with the large variance in the degree distribution of different nodes, our heterogeneous detection is particularly valuable in this setting. In contrast, the degradation of the other methods on Small-World topologies is not quite as dramatic, although the improvement offered by the proposed approach is still quite pronounced. Among the alternatives, it is also revealing that personalizing thresholds results in second-best performance: again, takng account of network topology is crucial; somewhat surprisingly, it often outperforms re-training, which explicitly accounts for adversarial evasion, but not network topology.

5 CONCLUSION

We address the problem of adversarial detection of malicious content spreading through social networks. Traditional approaches use with a homogeneous detector or a personalized filtering approach. Both ignore (and thus fail to exploit knowledge of) the network topology, and most filtering approaches in prior literature ignore the presence of an adversary. We present a combination of modeling and algorithmic advances to systematically address this problem. On the modeling side, we extend diffusion modeling to allow for dependence on the *content* propagating through the network, model the attacker as choosing both the malicious content, and initial target on the social network, and allow the defender to choose heterogeneous detectors over the network to block malicious content while allowing benign diffusion. On the algorithmic side, we solve the resulting Stackelberg game by first representing it as a bilevel program, then collapsing this program into a single-level program by exploiting the problem structure and applying KKT conditions, and finally deriving a projected gradient descent algorithm using explicit and implicit gradient information. Our experiments show that our approach dramatically outperforms, homogeneous classification, adversarial learning, and heterogeneous but non-adversarial alternatives.

ACKNOWLEDGEMENT

This work was supported, in part by the National Science Foundation (CNS-1640624, IIS-1649972, and IIS-1526860), Office of Naval Research (N00014-15-1-2621) and Army Research Office (W911NF-16-1-0069).

REFERENCES

[1] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. Technical report, National Bureau of Economic Research, 2017.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[3] A.-L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications*, 311(3):590–614, 2002.

[4] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006.

[5] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] M. Brückner and T. Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD*, pages 547–555. ACM, 2011.

[7] C. Budak, D. Agrawal, and A. El Abbadi. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web*, pages 665–674. ACM, 2011.

[8] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[9] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec. Antisocial behavior in online discussion communities. In *International Conference on Weblogs and Social Media*, pages 61–70, 2015.

[10] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[11] G. V. Cormack et al. Email spam filtering: A systematic review. *Foundations and Trends® in Information Retrieval*, 1(4):335–455, 2008.

[12] N. Dalvi, P. Domingos, S. Sanghai, D. Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD*, pages 99–108. ACM, 2004.

[13] N. Du, L. Song, M. G. Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems*, pages 3147–3155, 2013.

[14] N. Du, L. Song, H. Woo, and H. Zha. Uncover topic-sensitive information diffusion networks. In *Artificial Intelligence and Statistics*, pages 229–237, 2013.

[15] N. Du, L. Song, M. Yuan, and A. J. Smola. Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems*, pages 2780–2788, 2012.

[16] M. Gomez-Rodriguez and B. Schölkopf. Influence maximization in continuous time diffusion networks. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 579–586. Omnipress, 2012.

[17] J. Holcomb, J. Gottfried, and A. Mitchell. News use across social media platforms. *Pew Research Journalism Project*, 2013.

[18] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD*, pages 137–146. ACM, 2003.

[19] A. Laszka, Y. Vorobeychik, and X. Koutsoukos. Optimal personalized filtering against spear-phishing attacks. In *AAAI Conference on Artificial Intelligence*, 2015.

[20] B. Li and Y. Vorobeychik. Feature cross-substitution in adversarial classification. In *Advances in neural information processing systems*, pages 2087–2095, 2014.

[21] B. Li, Y. Vorobeychik, and X. Chen. A general retraining framework for scalable adversarial classification. *arXiv preprint arXiv:1604.02606*, 2016.

[22] M. Lichman. UCI machine learning repository, 2013.

[23] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD*, pages 641–647. ACM, 2005.

[24] S. Mei and X. Zhu. The security of latent dirichlet allocation. In *Artificial Intelligence and Statistics*, pages 681–689, 2015.

[25] S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI Conference on Artificial Intelligence*, pages 2871–2877, 2015.

[26] J. Tsai, T. H. Nguyen, and M. Tambe. Security games for controlling contagion. In *AAAI Conference on Artificial Intelligence*, 2012.

[27] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[28] Y. Vorobeychik and J. Letchford. Securing interdependent assets. *Autonomous Agents and Multi-Agent Systems*, 29(2):305–333, 2015.

[29] J. Wallinga and P. Teunis. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American Journal of Epidemiology*, 160(6):509–516, 2004.

[30] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

[31] V. A. Zorich and R. Cooke. Mathematical analysis i. 2004.