Adversarial Detection with Model Interpretation

Ninghao Liu
Texas A&M University
nhliu43@tamu.edu

Hongxia Yang
Alibaba Group
yang.yhx@alibaba-inc.com

Xia Hu
Texas A&M University
xiahu@tamu.edu

ABSTRACT

Machine learning (ML) systems have been increasingly applied in web security applications such as spammer detection, malware detection and fraud detection. These applications have an intrinsic adversarial nature where intelligent attackers can adaptively change their behaviors to avoid being detected by the deployed detectors. Existing efforts against adversaries are usually limited by the type of applied ML models or the specific applications such as image classification. Additionally, the working mechanisms of ML models usually cannot be well understood by users, which in turn impede them from understanding the vulnerabilities of models nor improving their robustness. To bridge the gap, in this paper, we propose to investigate whether model interpretation could potentially help adversarial detection. Specifically, we develop a novel adversary-resistant detection framework by utilizing the interpretation of ML models. The interpretation process explains the mechanism of how the target ML model makes prediction for a given instance, thus providing more insights for crafting adversarial samples. The robustness of detectors is then improved through adversarial training with the adversarial samples. A data-driven method is also developed to empirically estimate costs of adversaries in feature manipulation. Our approach is model-agnostic and can be applied to various types of classification models. Our experimental results on two real-world datasets demonstrate the effectiveness of interpretation-based attacks and how estimated feature manipulation cost would affect the behavior of adversaries.

KEYWORDS

Adversarial Detection, Machine Learning Interpretation, Spammer Detection

ACM Reference Format:
Ninghao Liu, Hongxia Yang, and Xia Hu. 2018. Adversarial Detection with Model Interpretation. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/3219819.3220027

1 INTRODUCTION

Machine learning models are increasingly employed in security-related applications such as spammer detection, malware detection and biometric authentication, to distinguish malicious instances (e.g., spammers) from normal ones [4, 15, 36, 39]. Different from traditional classification tasks, where data distributions are stationary and testing data usually follows the same distribution as the training data, security-related applications involve the adversarial competition between classifiers and malicious entities. Sophisticated and adaptive adversaries can change their behavior patterns to evade the detection in order to survive, thus undermining the effectiveness of the deployed classifiers [50]. Actively preparing the detectors against possible evolution of attacks, instead of passively reacting after discovering these changes, would improve the functionality and robustness of real-world systems.

The emerging field of developing ML detectors under adversarial settings has attracted increasing attentions. Existing methods can generally be categorized into three groups. First, some previous work applies feature selection and feature engineering methods [15, 31, 53]. These methods can build more accurate ML classifiers, but they do not fully eliminate threats from the evolution of adversaries. Second, some previous work formalizes the problem as a game between detectors and adversaries [2, 9, 10, 17], but the specific modeling process varies when different types of ML classifiers are involved. Third, along with the popularity of deep neural networks (DNNs) and its vulnerability to adversaries [48], various approaches have been developed for DNNs to defend against adversaries [22, 28, 35, 38] using fast gradient method or its variations. Some defensive strategies include adversarial training [22, 48], defensive distillation [44] and feature squeezing [51]. However, techniques on DNNs may not be directly applied to other classifiers where gradients of the objective function are not directly obtainable, such as decision trees and nearest neighbor classifiers. In addition, web security is moving beyond block-and-tackle techniques towards understanding the vulnerability of web systems and potential causes of attacks. By better understanding the underlying working mechanism of machine learning models, more robust detectors could be developed.

It has been observed that enabling interpretation of ML models would significantly help analyzing the performance and functionality of models in many applications [18, 30, 46]. As we know, although achieving superior prediction performance in many tasks, ML models are usually being treated as black-boxes. To tackle this issue, some recent work tries to identify how individual decisions are made [3, 30, 46] or to provide intuitive descriptions to the concepts learned by the models [16, 26]. Emerging attempts of interpreting ML models open new opportunities for adversarial detection, since understanding intrinsic mechanisms how ML models work could be helpful in discovering the irrationality and weakness of models in the decision making process. Motivated by these recent advances, in this work, we investigate whether interpretation could provide directions for enhancing the robustness of classifiers against adversaries. This is a challenging task due to the distinct characteristics of the problem. First, it is still an under studied problem that how

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220027

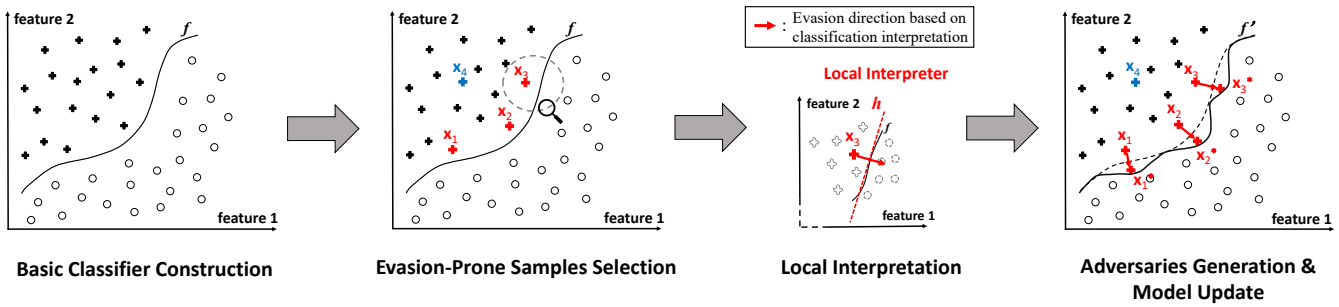

Figure 1: The framework of interpretation-based attack and defense based on evasion-prone samples. The dashed instances in the third image represent probing samples to explore the decision regions of the target classifier.

the interpretation of ML models could be effectively modeled and integrated for improving robustness of the models. A unified approach is needed as many learning models may have significantly different working mechanisms. Second, in the process of training adversary-robust classifiers, a large number of adversarial samples need to be generated. This is usually very computationally expensive. To tackle the aforementioned challenges, we propose a novel adversarial detection framework with model interpretation.

Specifically, in this paper, we design a general and efficient ML-based detection framework which is robust to adversaries. We proactively predict adversarial samples based on a set of local interpreters of the target detection model. The robustness of ML models is then improved through adversarial training [22] with the adversarial samples. The efficiency of this interpretation-based attack-defense pipeline promoted by considering only the low-confidence regions of ML classifiers. Our framework is model-agnostic as the interpretation technique is general to any type of classifier. Finally, we design a data-driven method to estimate the costs of perturbing different features. It is based on the observation that some features can be easily manipulated by adversarial spammers without affecting their malicious activities, while others are rarely touched by them. The main contributions of this paper are summarized as follows:

- We propose a novel adversarial detection approach, in which adversarial samples are crafted by incorporating the interpretation of ML models. The proposed approach is general to be applied to different types of classifiers.
- We design an efficient algorithm that reduces the number of adversarial samples needed in training adversary-robust classifiers.
- We design a series of experiments to comprehensively evaluate the effectiveness of attack and defense using different interpretation models and under different constraints.

2 PRELIMINARIES

Notations We first introduce the notations and application scenarios in this work. Let $\{\mathbf{X} \in \mathbb{R}^{N\times M}, \mathbf{y} \in \mathbb{R}^N\}$ be the data about user instances without considering adversaries, where N denotes the number of users and M is the feature dimension. The vector $\mathbf{x}_n = \mathbf{X}[n,:]$ $(1 \le n \le N)$ represents instance n in the feature space, $y_n = \mathbf{y}[n]$ is the label, and x_b denotes the b-th feature. In this work, we use *social spammer* detection [4, 15, 39, 50] as the scenario to introduce our framework, but the methodologies can be easily generalized to detection tasks against other malevolent entities. We set $y_n = 1$ for spammers and $y_n = 0$ for normal users. Then our goal is to build a spammer classifier $f(\mathbf{x}) : \mathbb{R}^M \to \{0, 1\}$, which is robust to future adversaries that may evolve (i.e., change their features) to evade the detection of f. We use existing spammer samples to predict the probable perturbation of adversaries. Let $\mathbf{x}^*$ be the adversary sample in feature space after behavior shifts of the original $\mathbf{x}$, where $\Delta\mathbf{x} = \mathbf{x}^* - \mathbf{x}$ denotes the perturbation in the feature space corresponding to the evasive actions taken by the adversary. An adversary attempts to make the detector classify it as a normal account. We use $\mathbf{X}^* \in \mathbb{R}^{N'\times M}$ to denote the data matrix for adversary accounts, and N' is the number of adversaries considered. It is worth noting that, in this paper, "adversaries" and "spammers" both refer to the malicious accounts in systems, but adversaries are a specific subset of spammers who actively attempt to evade the inspection of detectors. In practice, adversaries can either poison the training data before a classifier is established [1, 50] or try to avoid being detected by the deployed classifier, while in this paper we only focus on the latter scenario.

Adversarial Training We use adversarial training [22] as the fundamental defensive technique to improve the robustness of spammer classifiers against adversaries. The classifier is trained with a mixture of original and adversarial data instances. The overall loss function $\tilde{L}$ is formulated as below:

$$\tilde{L} = \alpha \cdot L(f, \mathbf{X}, \mathbf{y}) + (1 - \alpha) \cdot \sum_{\mathbf{x}^* \in \mathbf{X}^*} l(f, \mathbf{x}^*, 1), \tag{1}$$

where L and l measure dataset-level and instance-level classification errors, respectively. The parameter α controls the tradeoff between original and adversarial instances. We set $\alpha = 0.5$ in our experiments. The labels in the second term are fixed as 1 since they refer to spammers. In deep neural networks, it has been shown that $\mathbf{X}^*$ acts as the regularization term to increase model generality [48]. From another perspective, in our problem, $\mathbf{X}^*$ contains additional information of malicious instances after evolution, who are likely to reduce the effectiveness of the existing classifier. Unlike previous adversarial classification models that target certain types of classifiers [8, 10, 17, 54], the above formulation is independent of the type of the classifier f to be deployed.

3 ADVERSARY STRATEGY BASED ON MODEL INTERPRETATION

The adversarial training of detectors requires speculated adversarial instances as part of the training data, which however are not

directly available. To solve this problem, we develop a framework of interpretation-based adversary strategy to approximate the possible adversarial actions of spammers. The framework is illustrated in Figure 1. Given a classifier f, a special subset of spotted malicious instances are chosen as the seeds (red points in the figure), and a local interpreter is then built to explain why a seed $\mathbf{x}$ is regarded as malicious by f. Adversarial samples $\mathbf{x}^*$ are then generated by perturbing each seed towards its evasion direction determined by the interpreter. The perturbation cost is constrained by the capability of adversaries. Finally, adversarial training with both original data and adversarial instances is implemented to obtain the new detector robust to adversaries.

3.1 Evasion-Prone Samples Selection

Spammer instances have different chances to evade detection, depending on the decision confidence assigned by the classifier. A classifier becomes vulnerable if many spammer instances fall into its low-confidence decision regions, since some small perturbation of these instances may result in misclassification of the detector. We want to pay more attention to the *evasion-prone* (EP) samples that are more likely to shift across the decision boundary. From the spammers' perspective, it requires less effort for them to wash the maliciousness off the instances, thus making it feasible for controlling a large number of accounts. From the classifiers' perspective, evasion-prone accounts can provide more information about how to improve the current model against the adversarial actions of spammers. In this paper, EP samples are used as the *seeds* for generating adversarial samples. An example can be found in Figure 1, where $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ are evasion-prone samples, but not for $\mathbf{x}_4$.

We use the uncertainty score u_n [47] to measure the classification confidence for a given instance $\mathbf{x}_n$ assigned by f. The concrete form of u_n varies according to the specific type of classifiers. For example, $u_n = 1 - P(y_n = 1|\mathbf{x}_n)$ for probabilistic models or neural networks whose output range is constrained, and $P(y_n = 1|\mathbf{x}_n)$ could also be the spammer class probability predicted by decision trees and ensemble models [12]. For linear models, $u_n = -(\mathbf{w} \cdot \mathbf{x}_n + b)$, where $\mathbf{w}$ and b are model parameters. In this work, we choose instances with high u_n scores as seeds for generating adversarial samples.

There are two widely adopted assumptions about whether adversaries have full knowledge of ML models. In some cybersecurity problems, ML models on the server side are assumed to be confidential [35, 49, 52]. In this case, attackers build their local substitute model after making enough queries to the server, generate adversarial samples on the local model, and rely on the transferability [48] of the adversarial samples to have them still effective on the original server models. In this paper, however, we aim to solve a classifier-oriented problem, so the full knowledge of the classifier is assumed to be known [2, 14, 17]. In this setting, the generated adversaries can target the most critical vulnerabilities of the classifier and initiate more threatening attacks. Note that our work can also be easily applied in black-box attacks by simply approximating the original model with the attacker-side substitute.

3.2 Local Interpretation Based Attack

For each evasion-prone seed, we need to find out its most sensitive perturbation direction for crafting the adversarial sample. Most existing methods perturb seeds along the direction of local gradients of the loss function [6, 22, 42]. There are two problems for this class of methods. First, broader conditions of classification regions cannot be perceived by the local gradient, so gradient-based perturbation may be misled into unsupported regions. For example, on the left side of Figure 2, instances in red and blue regions will be recognized as spammers and normal users, respectively. The two samples moving along the red arrows will not be correctly directed towards blue regions to evade detection. Second, for some classifiers such as decision trees and ensemble methods, we cannot directly compute the gradient of loss function since a continuous prediction function is not available. For example, as shown on the right side of Figure 2, the outputs of the decision tree are simply discrete predictions rather than continuous values, where gradients are not directly obtainable.

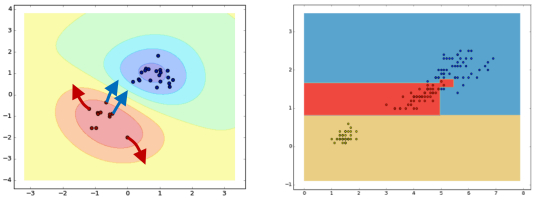

Figure 2: Left: Gradient descent based evasions targeting an SVM classifier with RBF kernels [6]. Right: An example of decision regions of a three-class decsion tree classifier.

We now introduce a new adversary samples crafting method based on the interpretation of ML models. Recent advances in ML interpretation enable end users to understand and effectively manage the systems built upon ML models. Adversarial attacks will be more effectively generated if the operation mechanism of ML models is utilized. Therefore, we propose to simulate attacks under the guidance of the model interpretation about how input data instances are classified as benign or malicious. The proposed method is model-agnostic and can be applied to estimate perturbation direction for any model.

Formally, given a seed instance $\mathbf{x}^s$ and the global classifier f, we define an explainable model $h \in H$ to locally approximate the prediction mechanism of f around $\mathbf{x}^s$. Here H denotes the class of explainable models, such as decision trees and linear models, or local vector flows constructed by Parzen window estimators [3]. The output of $h(\cdot) : \mathbb{R}^M \to \mathbb{R}$ measures the malicious score of the input instance. Let $l(f, h, \mathbf{x}^s)$ measure how close h is in approximating f in the space around $\mathbf{x}^s$, then the explanatory model h can be obtained by solving $\text{argmin}_{h \in H}\, l(f, h, \mathbf{x}^s) + \beta \cdot c(h)$, where $c(h)$ measures the complexity of h. If h is chosen from linear classifiers, for example, then we can choose $c(h) = \|\mathbf{w}\|_1$, where $\mathbf{w}$ denotes the weight vector of linear models. In this paper, we define

$$l(f, h, \mathbf{x}^s) = \sum_{\mathbf{x}' \in X'} \exp\left(\frac{-\|\mathbf{x}^s - \mathbf{x}'\|_2^2}{\eta^2}\right) (f(\mathbf{x}') - h(\mathbf{x}'))^2, \quad (2)$$

which is a weighted sum of approximation errors over a set of instances $\mathbf{x}'$ sampled around the seed $\mathbf{x}^s$, and η is the parameter controlling weight decay (parameter settings are discussed in experiments) [46]. The evasion-prone seeds, as we choose in the previous section, facilitate the interpretation process, because they are close

to the classification boundary so that it is easier when sampling $\mathbf{x}'$ to get both positive and negative labels for classification problems. The existence of negative samples prevents perturbation from going to unsupported regions shown in Figure 2. After obtaining the local explainable model h, we rely on it to provide directions about how to perturb the seed $\mathbf{x}^s$.

Attack Strategy: With the local explanatory model h that measures the maliciousness of a given instance, the adversarial sample $\mathbf{x}^*$ corresponding to the seed $\mathbf{x}^s$ can be obtained by solving:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \ h(\mathbf{x}), \quad s.t. \ \mathbf{x} \in \Omega(\mathbf{x}^s), \tag{3}$$

where $\Omega(\mathbf{x}^s)$ restricts the disturbance range around $\mathbf{x}^s$. The output $h(\mathbf{x}^*)$ is minimized so that the malicious score of the adversarial sample is maximally suppressed, which means we assume adversaries will try to stay as safe as possible by spending all of their budget. This strategy is different from the one applied in [14, 48], where adversaries try to minimize the perturbation as long as evasions are achieved. It is mainly because: (1) the former strategy considers the practical situation where adversaries have limited budget but want to achieve long-term survivals; (2) adversarial spammers assigned with low malicious scores are more threatening to the detector, so they are more helpful for model adjustment. The motivations of our strategy are that, on one hand, interpretation-based strategy has a broader understanding of f than gradient-based methods, so it can effectively lead perturbation into regions of lower maliciousness scores. On the other hand, the locality of h ensures its proximity to the global model f around $\mathbf{x}^s$, so that the adversarial samples targeting h are likely to be transferred to f [49].

Evasion Constraint: The constraint is defined as $\Omega(\mathbf{x}^s) = \{\mathbf{x} : d(\mathbf{x}, \mathbf{x}^s) \leq \zeta\}$. We will use l_2 and l_1 norms as two distance metrics for $d()$.

- l_2 norm constraint: We first use the commonly applied l_2 norm $d(\mathbf{x}, \mathbf{x}^s) = \|\mathbf{x} - \mathbf{x}^s\|_2$ to constrain attacks. Here we do not differentiate among the varying costs in perturbing different features. We consider this constraint in order to compare with other state-of-the-art attacking methods in terms of effectively choosing the perturbation direction. For the l_2 norm constraint, we will use LASSO as the local interpretation model, i.e., h is chosen from linear models and $c(h) = \|\mathbf{w}\|_1$.
- l_1 norm constraint: We use l_1 norm constraint to model more practical attacking scenarios, where $\Omega(\mathbf{x}^s) = \{\mathbf{x} : \|(\mathbf{x} - \mathbf{x}^s) \circ \mathbf{e}\|_1 \leq \zeta\}$ and $\circ$ denotes element-wise multiplication. Here we assume each adversary is assigned with a universal cost budget ζ. And $e_b \in \mathbf{e}$ denotes the effort cost for one unit change of feature x_b. Adversary workers are constrained by their capability, i.e., the maximum total effort they would like to spend in manipulating each instance. The cost of changing the value of x_b can be related to the feature's robustness [41, 53]. A larger e_b suggests it is effort-consuming to change x_b for reversing the classification result. For l_1 norm constraint, we will try two local interpretation models: LASSO and Vector Flow model [3]. Since LASSO is a linear predictor, for each seed and its local interpretation vector $\mathbf{w}$, only the feature b with the largest $\frac{|w_b|}{e_b}$ will be manipulated. For Vector Flow model, it is constructed with a number of Parzen windows to the samples in decision regions. The problem to solve is formulated as:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \ \frac{\sum\limits_{i, y_i=1} k_\sigma(\mathbf{x} - \mathbf{x}'_i) - \sum\limits_{i, y_i=0} k_\sigma(\mathbf{x} - \mathbf{x}'_i)}{\sum\limits_{i} k_\sigma(\mathbf{x} - \mathbf{x}'_i)} \tag{4}$$
$$s.t. \ \|(\mathbf{x} - \mathbf{x}^s) \circ \mathbf{e}\|_1 \leq \zeta,$$

where $k_\sigma(\cdot)$ is the Gaussian kernel, and $\mathbf{x}'_i$ denotes the sample for building the local interpretation model. The only hyper-parameter σ can be determined through validation, where some samples are chosen to build Parzen windows and some others are generated for validation. We adopt the iterative gradient descent algorithm [6] to solve for $\mathbf{x}^*$ in Equation 4.

3.3 Adversary Costs Estimation

In practice, different features have varying costs to be manipulated. The costs refer to the profit loss of attackers or the efforts paid for initiating adversarial actions. Such costs originate from several aspects. First, one malicious worker in real world usually controls a large number of spammer accounts [5, 21]. It is harder to mimic certain aspects of normal users' behaviors, such as having diverse content in post streams or creating a lot of reviews recognized as useful by other customers. Second, spammers have different goals from legitimate users, in terms that they try to spread malicious or false content to deceive others and gain profit. For example, spammers seldom interact with other users online and they tend to include many URLs in their posts. Third, some designed features in traditional spammer detection are simply shallow artifacts extracted from datasets rather than intrinsic properties or behavior representations of accounts. Spammers can easily manipulate these features without much effort.

To estimate feature robustness, a straightforward way is to find the closed-form formula for each feature and explicitly compute the difficulty of changing its value [17, 53], or to unify the cost to a constant value [11]. These approaches, however, can only be applied to a limited number of straightforwardly designed features. Therefore, we design an empirical measure by relying on evasive spammers themselves to indicate the difficulty of controlling different aspects of their behaviors. We estimate an adversary's cost of manipulating feature x_b as:

$$e_b = \begin{cases} 1/|\overline{x_b^{ep}} - \overline{x_b^+}|, & \text{if } sign(\overline{x_b^{ep}} - \overline{x_b^+}) = sign(\overline{x_b^-} - \overline{x_b^+}) \\ e_{MAX}, & \text{Otherwise} \end{cases}, \tag{5}$$

where $\overline{x_b^-}$ denotes the average value of x_b for normal instances, $\overline{x_b^{ep}}$ is averaged over evasion-prone samples, and $\overline{x_b^+}$ averages x_b for spammers without the evasion-prone samples. The intuition of this equation is that, if evasion-prone samples have significantly shifted their x_b values to make them less malicious, then it indicates the cost of manipulating x_b is small, and vice versa. In addition, if the change direction of x_b^{ep} deviates from that of normal users, then we assume spammers are not willing to manipulate x_b and we set e_b with a large value e_{MAX}. In our experiments, we simply set $e_{MAX} = \infty$. More complex scenarios such as the correlation between features are beyond our scope in this study. We use evasion-prone samples here because they possess relatively strong incentives to evade the

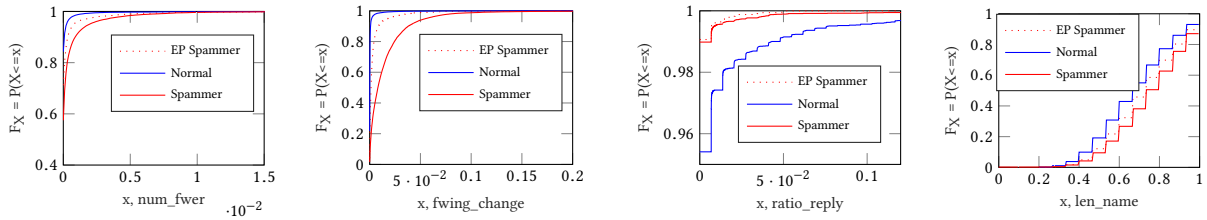

Figure 3: CDF of example features for overall non-spammers, lazy spammers and EP spammers in the Twitter dataset.

detection and have already taken some actions, so they provide information of how spammers may evolve.

Visualizations of the shift of EP instances are depicted in Figure 3, where the cumulative distribution function (CDF) of some example features are plotted. We can observe that, for *num_fwer* (number of followers), *fwing_change* (change speed of followings) and *len_name* (length of username), EP spammers are more similar to normal users as their CDF curves are closer than the lazy spammers. Especially, we find that $|\overline{x_b^{ep}} - \overline{x_b^+}|/|\overline{x_b^-} - \overline{x_b^+}| \approx 85\%$ for x_b = *fwing_change*, so *fwing_change* may not be a robust feature with respect to potential attacks. However, for *ratio_reply*, the plot shows that the distribution difference becomes more remarkable for EP spammers, so it is not preferred to be manipulated by spammers. The feature manipulation costs are taken into consideration in our attack strategies under l_1 norm constraint. The effectiveness of the attacks, defenses and the roles of different features will be further discussed in our experiments.

4 EXPERIMENTS

In this section, we evaluate the performance of the proposed method on several real-world datasets. First, we will introduce the experimental settings. Second, we will measure the effectiveness of attacks generated by the proposed method compared with other baseline attacking methods. Third, we will measure the performance of adversarial training using the proposed method compared with the baseline defending methods.

4.1 Experimental Settings

Here we introduce the datasets extracted from several web platforms applied in our experiments, the basic classifiers as targets of adversaries, and the baseline methods for dealing with adversaries.

Datasets: We use two public datasets for evaluation, including the Twitter dataset [31] and the YelpReview dataset [40]. The Twitter dataset, collected via social honeypot techniques, consists of the profile information and post streams of content polluters and legitimate users. The YelpReview dataset consists of reviewers' information and review posts about restaurants on Yelp. The labels of Twitter users and Yelp posts are available. We set Yelp reviewers with more than 10% spam posts as spammers. After preprocessing, for Twitter dataset, we have 41, 499 users and each user instance has 28 features, and for YelpReview dataset we have 16, 941 reviewers and each reviewer has 16 features.

Target Models: Spammer detection using traditional classification models has been extensively studied [4, 15, 31, 32]. Following previous work, to cover diverse types of models, we apply Logistic Regression (LR), Support Vector Machines (SVM), Random Forest (RF), Neural Network (NN) and Ensemble method (ESM) to detect spammers in our datasets. The performances of different models are shown in Table 1 (outside parentheses). The parameters of the models are determined via 5-fold cross-validation. YelpReview dataset is more complicated than Twitter, so the resultant classifiers for YelpReview are more complex. For examples, the weights of regularization terms in LR and SVM are set smaller for YelpReview, and the number of layers and the size of each layer are also larger in NN. These trained classifiers will be targeted by various attacking strategies, and will be improved by defensive methods.

Baseline Methods: The methods applied in our experiments can be categorized into two classes, i.e., attacking methods and defensive methods, as introduced below.

- Gradient Descent Attacking method (GDA) [6]: An attacking method moving samples along the direction of gradient vectors. The gradients or subgradients of the loss function are needed. Similar attacking strategies can also be found in [42].
- Fast Gradient Sign method (FGS) [22]: An attacking method which perturbs a legitimate sample $\mathbf{x}$ to $\mathbf{x} + \epsilon sign(\nabla_{\mathbf{x}} l(f, \mathbf{x}, y))$, where $sign(\nabla_{\mathbf{x}} l(f, \mathbf{x}, y))$ denotes the perturbation direction determined by the the sign of the cost function's gradient. FGS has been widely applied in deep neural network models.
- Per-worker Optimal Evasion (POE) [50]: An attacking method where different combinations of feature perturbation are tested until the target classifier misclassifies the perturbed instance or the maximum number of attempts is reached.
- Defensive Distillation (DD) [44]: A defensive method by transfering the knowledge learned by a complex model f to a compact model f'. The soft labels generated from complex models f are used to train the new models [23]. It has been shown that, by making f' have the same model structure with f, the distilled model f' is more robust to adversarial samples than f.

However, it is hard to directly apply gradient-based methods (e.g., GDA and FGS) to attack tree-based models or ensemble models. Motivated by the black-box attacking strategy introduced in [43], we first approximate RF and ESM classifiers with a neural network as the substitute and then apply GDA and FGS for attacking.

4.2 Effectiveness of the Attacking Strategy

We evaluate the effectiveness of the proposed method by checking whether the crafted adversarial samples could successfully evade the detection of target classifiers. For local interpretation, we set $d_{pos-neg}/\sqrt{M}$ as the variance of multivariate Gaussian distributions for sampling $\mathbf{x}'$ around each seed, where $d_{pos-neg}$ is the seed's distance to the closest non-spammer instance and M is the

	Twitter			YelpReview		
	Prec	Recall	F1	Prec	Recall	F1
LR	0.910 (0.867, 0.869)	0.889 (0.914, 0.915)	0.899 (0.890, 0.892)	0.794 (0.788, 0.776)	0.854 (0.890, 0.912)	0.823 (0.836, 0.838)
SVM	0.936 (0.911, 0.907)	0.928 (0.946, 0.944)	0.932 (0.929, 0.925)	0.905 (0.893, 0.887)	0.882 (0.890, 0.898)	0.893 (0.892, 0.893)
RF	**0.952 (0.952, 0.952)**	**0.962 (0.968, 0.965)**	**0.957 (0.960, 0.959)**	**0.988 (0.989, 0.985)**	**0.951 (0.961, 0.964)**	**0.969 (0.975, 0.974)**
NN	0.894 (0.879, 0.890)	0.958 (0.965, 0.968)	0.925 (0.920, 0.927)	0.937 (0.858, 0.905)	0.884 (0.919, 0.950)	0.910 (0.887, 0.927)
ESM	0.944 (0.927, 0.925)	0.943 (0.956, 0.961)	0.944 (0.941, 0.943)	0.970 (0.945, 0.959)	0.940 (0.956, 0.965)	0.955 (0.951, 0.962)

Table 1: Spammer detection performance of the proposed framework. Each triple represents: performance *without* adversarial training, performance *with* l_2 adversarial training, and performance *with* l_1 adversarial training.

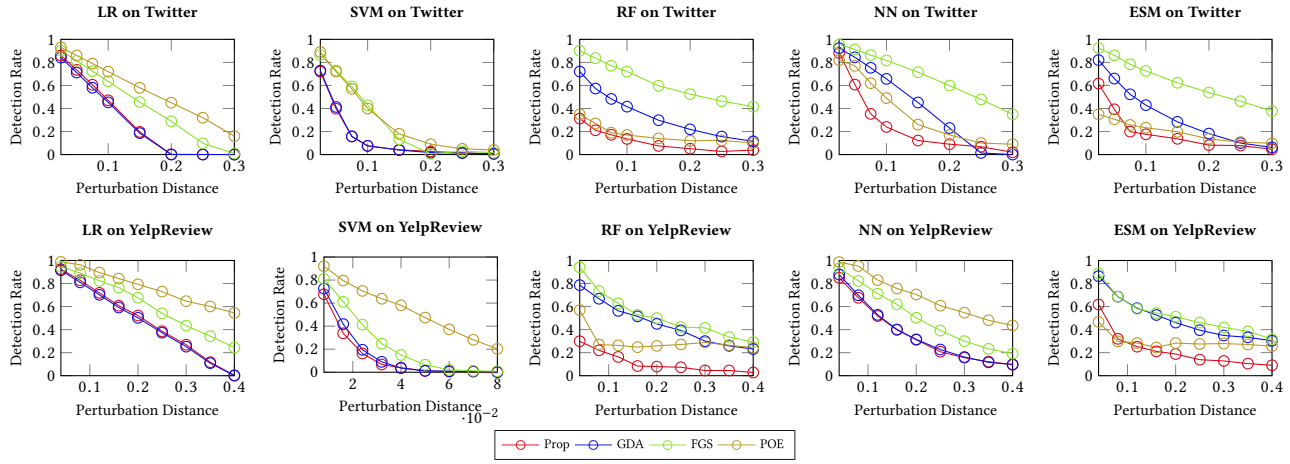

Figure 4: L_2 attack effectiveness evaluation with different perturbation distances on different types of classifiers.

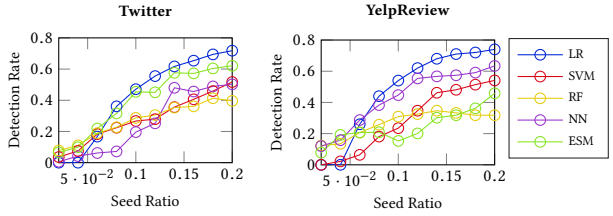

Figure 5: Attack effectiveness evaluation with different percentage of evasion-prone samples.

number of features. The weight decay η is set as $+\infty$ in all experiments, so we treat all samples $\mathbf{x}'$ equally. The baseline attacking methods are implemented for comparison.

4.2.1 Effect of evasion-prone samples selection. To show the effect of applying evasion-prone samples, we measure the effectiveness of attacks with and without using evasion-prone samples as the seeds. The results averaged on five runs are shown in Figure 5. The x axis represents the portion of top spammers used as seeds for generating adversarial samples, and the y axis shows the detection rates of the target classifier. Data instances of high uncertainty scores are regarded as evasion-prone. Instances are ranked according to the uncertainty score, where the top ones have higher scores. For better visualization, the adversarial perturbation distance is fixed as $0.1 \times d^{avg}_{pos-neg}$ for Twitter data and $0.2 \times d^{avg}_{pos-neg}$ for YelpReview data (except $0.04 \times d^{avg}_{pos-neg}$ in SVM), where $d^{avg}_{pos-neg}$ is the average of distance from seeds to their closest non-spammer neighbors. The figures show that, as we include more instances whose labels are more certain to the classifier, the success rate of evasion by the adversarial samples decreases (i.e., the detection accuracy of the classifier increases). If we randomly choose seeds for adversaries crafting, the evasion rate (not shown in figures) is usually less than 50%. Therefore, carefully choosing the seeds for generating adversarial samples will increase the effectiveness of attacks, thus providing more information about the weakness of the classifier.

4.2.2 Effectiveness of l_2 attack. In this part, we explore how effectively the interpretation-based attack strategy under l_2 norm constraint would impair the performance of classifiers. We will compare the proposed method with baseline methods on different classifiers. For visualization, we choose top 4% evasion-prone samples as seeds for SVM and RF, 6% for ESM, 9% for NN and LR, which are applied to all attacking methods. The results are shown in Figure 4. The x coordinate means using $x \times d^{avg}_{pos-neg}$ as the perturbation distance. In general, the proposed method outperforms the baseline methods as it achieves greater evasion rate (i.e., lower classification accuracy). Its performance is stable across different types of classifiers as the interpretation process is model-agnostic. As the perturbation distance increases, more adversarial samples evade the detection of the target classifier. The proposed method has similar performance as baseline methods (e.g., GDA) for some classifiers (e.g., LR, SVM). It indicates that in certain cases the gradient vector may also act as the local interpreter. In some cases,

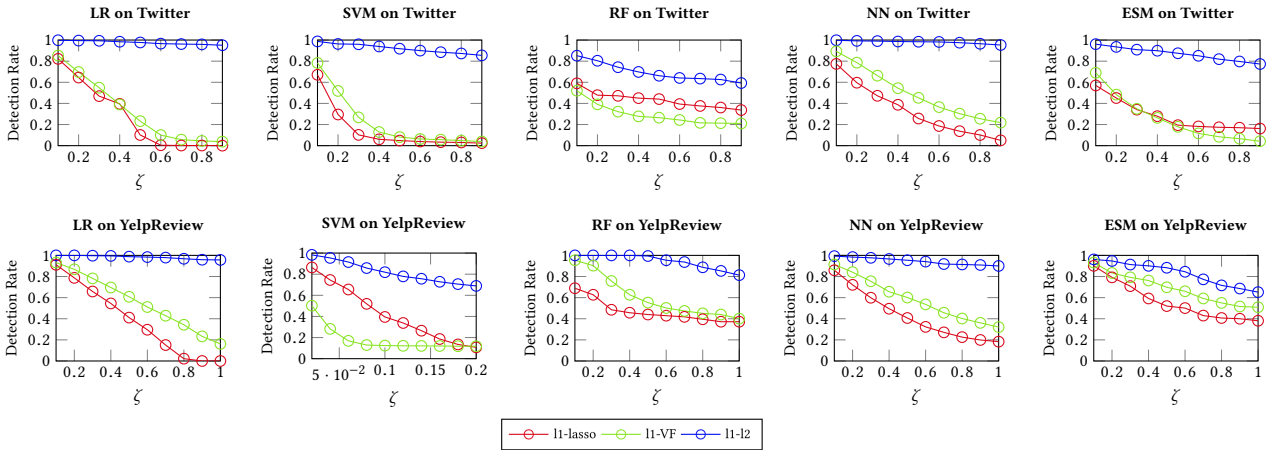

Figure 6: L_1 attack effectiveness evaluation with different perturbation distances on different types of classifiers.

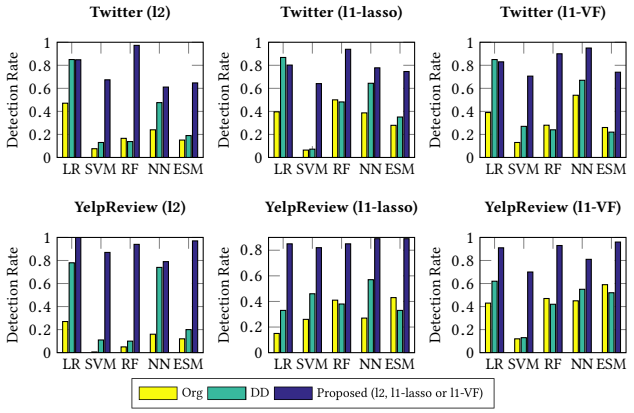

Figure 7: Defense effectiveness evaluation of proposed methods compared to Defensive Distillation.

due to randomness, the POE method achieves good performance when the perturbation distance is small, but is outperformed by some of other methods as perturbation distance increases. The performances of GDA on RF and ESM are worse than those on other classifiers. The probable reason could be that they are actually not directly attacking the original classifiers but the substitute models.

4.2.3 Effectiveness of interpretation-based l_1 attack. We evaluate the attack performance of the proposed method with different local interpretation models under the l_1 norm constraint. Here feature manipulation costs are considered. The results are shown in Figure 6. Here the "*l1-l2*" method first generates adversarial samples under l_2 constraint, and then normalize the samples with respect to the costs specified by ζ. It is not a surprise that *l1-lasso* and *l1-VF* significantly outperform *l1-l2*, since the former ones select features of small costs to manipulate while the latter wastes effort in perturbing robust features which are of little reward. In general, the attack effectiveness of *l1-lasso* is greater than that of *l1-VF* as *l1-lasso* achieves lower detection rates on more classifiers. Though not shown in the figures, *l1-lasso* is more "concentrated" than *l1-VF* in terms that it is more focused on manipulating only a small number of features, and these features have low cost e_b. Let p_b denote the probability that feature b is perturbed by adversaries and $\mathbf{p} = [p_1, ..., p_B]$, $entropy(\mathbf{p}_{lasso}) = 0.11$ and 1.3 respectively in Twitter and YelpReview data, while $entropy(\mathbf{p}_{VF}) = 1.28$ and 1.87 for the two datasets. After inspecting the adversarial samples, we found that for Twitter data, nearly all adversaries choose to perturb the *fwing_change* feature. This is because its e_b is small (i.e., the feature is easy to be manipulated) and the gradient with respect to this feature is large (i.e., classifiers heavily rely on this feature to make decisions). In practice, slowing down the speed of following more users will not significantly affect the malevolent activities of spammers, and this feature tends to be controlled by adversaries. For YelpReview data, the manipulated features are relatively diverse across different samples. Some examples include *speedReviews* and *reviewCount*. It indicates that spammers can suppress their posting activities to a safety zone, which is relatively easier than changing other features such as *usefulCount* and *scoreDeviation* as these features are not fully under spammers' control.

4.3 Effectiveness of the Defensive Strategy

To evaluate the effectiveness of defense against adversaries, we will check whether the adversarial samples can be captured by the new classifiers after applying defensive strategies, and whether the classification results on the original data are not negatively affected by the shift of decision boundary. Some approaches like Defensive Distillation are already defensive methods, while the correspondent defensive method for an attacking strategy is to use its adversarial samples as part of the retraining data to obtain new classifiers.

4.3.1 Comparison to Defensive Distillation. We first compare the proposed method to Defensive Distillation concerning the ability to capture adversarial samples. Both l_2 and l_1 (using LASSO and VF as the local model) norms are involved. The results are shown in Figure 7. The perturbation distance in l_2 attack is $0.1 \times d^{avg}_{pos\_neg}$ for Twitter and $0.3 \times d^{avg}_{pos\_neg}$ for YelpReview, while the perturbation cost in l_1 attack is 0.4 for Twitter and 0.7 for YelpReview, except that the perturbation in SVM is further multiplied by 0.2 for visualization. In general, the capture rate of adversarial samples increases after applying defensive methods. The proposed method has better

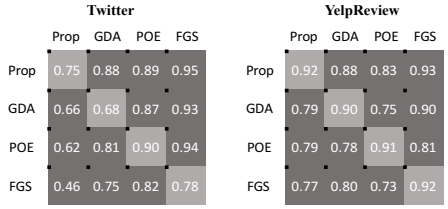

Figure 8: Evaluation of adversarial training effectiveness through cross attacking and defense using $\mathbf{l}_2$ attacks.

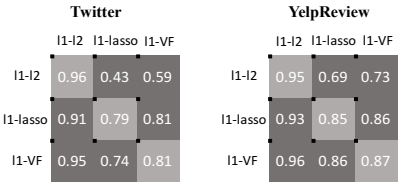

Figure 9: Evaluation of adversarial training effectiveness through cross attacking and defense using $\mathbf{l}_1$ attacks.

overall performance than Defensive Distillation. DD has good performance in NN and LR (could be seen as a one-layer NN), while it does not perform well in RF and ensemble methods. The proposed method is well adapted to a wide range of classification models, so it is helpful to proactively predict the evolution of malevolent entities and take it into consideration for classifier reconstruction.

4.3.2 Effectiveness of Adversarial Training. We first re-evaluate the performance of new models on the old test data, to check whether the prediction results are affected after adversarial training. Other settings remain the same as the previous part. The results are shown in Table 1. The performances before and after using adversarial training are shown inside and outside parentheses, respectively. Compared with the original classifiers, the new classifiers have higher Recall and lower Precision, because the adversarial samples used in training expand the region in which an instance is classified as a spammer. The F1 scores do not significantly change. RF has the best detection performance and shows good compatibility to adversarial training. In addition, the increases in Recall and F1 score using l_1 adversarial samples are greater than using l_2 samples. The possible reason is that the consideration of feature manipulation costs leads the attacks towards the distribution of crafty spammer instances that were missed by the old classifiers.

We then compare the defense performance of adversarial training using adversarial samples generated by different attacking methods. We adopt the cross-attack schema as the evaluation framework [42]. The results are shown in Figure 8 and Figure 9 for l_2 and l_1 constrained attacks, respectively. Each row indicates the adversarial samples used for retraining classifiers, while each column corresponds to the attacking strategy to evade the retrained classifiers. Each entry records the average detection rate of attacking samples over all types of classifiers. In Figure 8, we observe that the classifiers trained using the proposed method is more robust to other attacks (the entry values in the *Prop* row are larger), and the attack initiated by the proposed method is more threatening to the new classifiers (the entry values in the *Prop* column are smaller). Note that high detection rates on diagonal entries are due to the fact that the training and test adversarial samples are drawn from the same distribution, so they do not necessarily indicate good defensive performances. An interesting phenomenon observed from Figure 9 is that, although $l1_{VF}$ does not achieve comparable attacking success as $l1_{lasso}$, it sometimes has better defensive performance. This is possibly because, as we discussed using *entropy* in 4.2.3, $l1_{lasso}$ concentrates on manipulating only the most vulnerable features to gain the greatest attack reward. However, the features manipulated by $l1_{VF}$ are more diverse, meaning that more attacking scenarios are considered. The relation between attack and defense is like that of lance and shield. Effective attacks provide valuable guidance to defense deployment, but an effective defense method should cover various risk scenarios.

5 RELATED WORK

Classification models are increasingly used in security-related applications like spam filtering, biometric authentication and malware detection [4, 15, 25, 36, 39]. These applications have an adversarial nature since sophisticated malicious entities may manipulate their behaviors to evade being detected. Traditional ML-based detectors do not consider such adversarial settings. There are two forms of attacks, evasion attacks and poisoning attacks [7, 50]. In the former case, adversaries modify their behavior to fool the established system, while the latter tampers with the training data. We focus on evasion attacks in our paper.

Adversarial detection has been investigated using game theories, where adversaries and the classifier act as opposite players [11, 17]. Typically, the single-shot version of adversarial classification game is considered, and an equilibrium is achieved to obtain the solution [10]. A problem with these methods is that they are usually designed specifically for certain type of classifiers. As deep learning (DL) has become popular on many ML problems, recent studies show that DL is vulnerable to adversarial samples [14, 24, 48]. Fast gradient sign method is one of the most popular approaches for creating adversarial instances [22, 38]. Some other attacking methods include box-constrained L-BFGS method [48], iterative least likely method [28], Jacobian-based saliency map attack [45], adversarial samples generation based on an ensemble of deep models [35]. Some typical defensive methods include defensive distillation [44], region-based classification [13] and feature squeezing [51].

Despite widespread applications, some ML models remain mostly black boxes to end users. Preliminary work on interpretation for ML models can be divided into two categories: 1) interpreting or reorganizing the working mechanisms or the learned concept of the model [12, 20, 27, 33, 37]; 2) extracting the signaling features or rules based on which individual predictions are made [3, 16, 19, 34, 46]. For the former category, example-based explanation is one of the most widely used approaches [26, 27, 29]. For prediction instances interpretation, the goal is to locally approximate the behavior of prediction models [3], or to select significant features and rules that lead to the prediction results [30, 46].

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel and general adversarial detection framework by utilizing the interpretation of ML models. The framework applies the adversarial training method, where anticipated adversarial samples are used as part of the training data. We utilize

the local interpretation of detection models to unveil the weakness of existing detectors and provide directions for adversarial perturbation. The efficiency of the proposed framework is improved through using evasion-prone instances as the seeds which are then perturbed for crafting adversarial samples. We consider l_2 and l_1 norms as distance measures to form the perturbation constraint. An empirical measure is designed to estimate the cost of manipulating each feature. Experiments on spammer detection are conducted to evaluate the effectiveness of attack and defense of the proposed framework. Our future work includes using model interpretation for developing defensive strategies, extending current work to deal with high-dimensional raw data, and adapting the methods for datasets with relational links between instances.

ACKNOWLEDGMENTS

The work is, in part, supported by DARPA (#N66001-17-2-4031) and NSF (#IIS-1718840). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Scott Alfeld, Xiaojin Zhu, and Paul Barford. 2016. Data Poisoning Attacks against Autoregressive Models.. In *AAAI*.
[2] Scott Alfeld, Xiaojin Zhu, and Paul Barford. 2017. Explicit Defense Actions Against Test-Set Attacks.. In *AAAI*.
[3] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research* (2010).
[4] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. 2010. Detecting spammers on twitter. In *CEAS*.
[5] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*.
[6] Battista Biggio, Igino Corona, and Others. 2013. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.
[7] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2014. Security evaluation of pattern classifiers under attack. *TKDE* (2014).
[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2011. Support vector machines under adversarial label noise. In *ACML*.
[9] Michael Brückner, Christian Kanzow, and Tobias Scheffer. 2012. Static prediction games for adversarial learning problems. *JMLR* (2012).
[10] Michael Brückner and Tobias Scheffer. 2009. Nash equilibria of static prediction games. In *NIPS*.
[11] Michael Brückner and Tobias Scheffer. 2011. Stackelberg games for adversarial prediction problems. In *KDD*.
[12] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *KDD*.
[13] Xiaoyu Cao and Neil Zhenqiang Gong. 2017. Mitigating Evasion Attacks to Deep Neural Networks via Region-based Classification. In *ACSAC*.
[14] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE.
[15] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *WWW*.
[16] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *NIPS*.
[17] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. 2004. Adversarial classification. In *KDD*.
[18] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. 2018. Towards Explanation of DNN-based Prediction with Guided Feature Inversion. In *KDD*.
[19] Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic, Pang-Ning Tan, Vipin Kumar, Jaideep Srivastava, and Paul Dokas. 2004. Minds-minnesota intrusion detection system. *Next generation data mining* (2004).
[20] Jun Gao, Ninghao Liu, Mark Lawley, and Xia Hu. 2017. An Interpretable Classification Framework for Information Extraction from Online Healthcare Forums. *Journal of Healthcare Engineering* (2017).
[21] Saptarshi Ghosh, Bimal Viswanath, et al. 2012. Understanding and combating link farming in the twitter social network. In *WWW*.
[22] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES. *stat* 1050 (2015), 20.
[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *stat* 1050 (2015), 9.
[24] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).
[25] Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 219–230.
[26] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *NIPS*.
[27] Been Kim, Cynthia Rudin, and Julie A Shah. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *NIPS*.
[28] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).
[29] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* (2015).
[30] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *KDD*.
[31] Kyumin Lee, Brian David Eoff, and James Caverlee. 2011. Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter.. In *ICWSM*.
[32] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *IJCAI*.
[33] Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu. 2018. On Interpretation of Network Embedding via Taxonomy Induction. In *KDD*.
[34] Ninghao Liu, Donghwa Shin, and Xia Hu. 2017. Contextual Outlier Interpretation. *arXiv preprint arXiv:1711.10589* (2017).
[35] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. In *ICLR*.
[36] E Mariconti, L Onwuzurike, P Andriotis, E De Cristofaro, G Ross, and G Stringhini. 2017. MamaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. In *NDSS*.
[37] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for Interpreting and Understanding Deep Neural Networks. *arXiv preprint arXiv:1706.07979* (2017).
[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*.
[39] Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *WWW*.
[40] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013. Fake review detection: Classification and analysis of real and pseudo reviews. *Technical Report UIC-CS-2013–03, University of Illinois at Chicago, Tech. Rep.* (2013).
[41] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. 2010. Efficient and robust feature selection via joint L2,1-norms minimization. In *NIPS*.
[42] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).
[43] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2016. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697* (2016).
[44] Nicolas Papernot, Patrick McDaniel, and Others. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *SP*.
[45] Nicolas Papernot, Patrick McDaniel, and Others. 2016. The limitations of deep learning in adversarial settings. In *EuroS&P*.
[46] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *KDD*.
[47] Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52, 55-66 (2010), 11.
[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[49] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *USENIX Security*.
[50] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. 2014. Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers.. In *Usenix Security*.
[51] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *NDSS*.
[52] Weilin Xu, Yanjun Qi, and David Evans. 2016. Automatically evading classifiers. In *Proceedings of the 2016 Network and Distributed Systems Symposium*.
[53] Chao Yang, Robert Chandler Harkreader, and Guofei Gu. 2011. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *Recent Advances in Intrusion Detection*. Springer, 318–337.
[54] Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Bowei Xi. 2012. Adversarial support vector machine learning. In *KDD*.