Information Sciences 573 (2021) 396–411

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Data poisoning against information-theoretic feature selection

Heng Liu *, Gregory Ditzler

Department of Electrical and Computer Engineering, University of Arizona, 1230 E Speedway Blvd, Tucson, AZ 85721, USA

ARTICLE INFO

Article history:
Received 14 September 2020
Received in revised form 16 April 2021
Accepted 21 May 2021
Available online 26 May 2021

Keywords:
Feature selection
Information theory
Adversarial learning

ABSTRACT

A typical assumption made in machine learning is that a learning model does not consider an adversary's existence that can subvert a classifier's objective. As a result, machine learning pipelines exhibit vulnerabilities in an adversarial environment. Feature Selection (FS) is an essential preprocessing stage in data analytics and has been widely used in security-sensitive machine learning applications; however, FS research in adversarial machine learning has been largely overlooked. Recently, empirical works demonstrated that the FS is also vulnerable in an adversarial environment. In the past decade, although the research community has made extensive efforts to promote the classifiers' robustness and develop countermeasures against adversaries, only a few contributions investigated FS's behavior in a malicious environment. Given that machine learning pipelines increasingly rely on FS to combat the "curse of dimensionality" and overfitting, insecure FS can be the "Achilles heel" of data pipelines. In this contribution, we explore the weaknesses of information-theoretic FS methods by designing a generic FS poisoning algorithm. We also show the transferability of the proposed poisoning method across seven information-theoretic FS methods. The experiments on 16 benchmark datasets demonstrate the efficacy of our proposed poisoning algorithm and the existence of transferability.

1. Introduction

Real-world data analytics increasingly adopt machine learning techniques in numerous applications given their superior benchmark performance (e.g., recommendation systems, speech recognition, image classification, etc.). In machine learning pipelines, apart from the generalization performance, security and robustness of a model are also essential. In the past decade, the security concern of machine learning and data science pipelines has become an increasingly important issue [16,7,9]. For example, in spam filtering, attackers own authorized email accounts can arbitrarily manipulate emails in their inbox, i.e., part of the training data used to train the classifier. As a result, in security-sensitive fields (e.g., spam filtering, malware detection), classifiers are updated periodically to cope with more advanced attacks. In general, an attack refers to the malicious behavior that incurs a loss in generalization on a machine learning pipeline. An attack can be mathematically formulated as follows [5]:

$$\max_{\mathcal{A}\in\Phi(\mathcal{A})} \mathcal{W}(\mathcal{A};\theta) : \theta \in \Theta \tag{1}$$

In (1), given the attacker's knowledge $\theta \in \Theta$ and the available adversarial examples $\Phi(\mathcal{A})$, an adversary's goal is to find the adversarial examples $\mathcal{A}$ that incurs the maximal loss: $\mathcal{W}(\mathcal{A};\theta)$.

* Corresponding author.
E-mail addresses: hengl@email.arizona.edu (H. Liu), ditzler@arizona.edu (G. Ditzler).

https://doi.org/10.1016/j.ins.2021.05.049
0020-0255/

Recent contributions proposed rules and algorithms to generate malicious samples. One such approach generates negligible adversarial perturbations using gradient descent to degrade the performance of classifiers [5]. Moreover, Goodfellow et al. demonstrated that many classifiers share part of their adversarial input space, which allows transferable malicious samples across different classifiers [16]. Meanwhile, the research community is also actively investigating the proposals to secure classifiers and create defensive countermeasures. For example, Zhou et al. proposed randomness in Deep Neural Networks (DNN) to defend against black-box image attacks [45]. Yang et al. adopted various input transformation techniques to filter the adversarial perturbations in DNN-based audio analysis applications [43].

Although the adversarial machine learning has been studied for many years since it first emerged in 2004 [5], the majority of the research community's contributions focused on designing secure classifiers but neglected the risk of FS in an adversarial environment. Early empirical works show that using insecure FS methods in a malicious environment can cause the classifiers that depend on FS to fail [44,41]. For example, an insecure FS method reduces the dataset complexity and exposes more evasion opportunities. Given the growing sophistication and variability of attacks, it is reasonable to expect that an adversary can exploit other weaknesses in machine learning pipelines via the FS stage.

FS is the technique of finding the minimal subset of features that allows for the maximal predictive power [19]. In the past decades, FS has become a crucial preprocessing stage to overcome issues such as "the curse of dimensionality" and overfitting [13]. Moreover, the FS techniques are particularly useful to reduce the data size because many classifiers scale in complexity with the dimensionality of the data. FS also facilitates the understanding of the dataset's structure by eliminating irrelevant and redundant features. Therefore, FS is widely used in nearly all data science pipelines.

FS methods are broadly categorized into three groups: wrappers, embedded, and filters. Wrapper methods assess a feature subset's utility by optimizing a particular classifier's performance and can generally achieve the minimal generalization loss; however, in a high computational cost even with powerful computing systems. For example, SVM-RFE optimizes feature subset by learning an SVM and recursively eliminates the least contributing feature [18]. Embedded methods select features by jointly optimizing a classification loss and feature selector objective. For example, Lasso uses an ℓ_1 regularization term to induce sparsity to select features [38]. Wrapper and embedded methods both require a classifier (i.e., classifier dependent). Filter methods, on the other hand, evaluate feature subset using classifier-independent criteria (e.g., mutual information, Pearson's correlation coefficient, K-L divergence, etc.).

There are few contributions that investigates FS's security and vulnerability. In [41], Xiao et al. revealed the risk of embedded FS algorithms (e.g., Lasso and Elastic-net) by designing targeted FS attack algorithms. In [44], Zhang et al. proposed an adversary-aware wrapper FS method that jointly optimizes the feature's informativeness and robustness. On the other hand, although the filter FS methods are widely applied in numerous data pipelines for its independence of classifiers, filter FS's vulnerability and security remain unknown, especially for information-theoretic filter FS methods. In [30], we proposed a heuristic FS attack algorithm that targets the mRMR method using a greedy optimization method; however, we did not experiment with how well the attacks generated against mRMR transfer to other information-theoretic FS algorithms.

In this paper, we focus on revealing the vulnerability of generic, such as those based on the likelihood filter framework [6], information-theoretic FS approaches. Specifically, we first develop two mutual information-based attack algorithms that specifically target information-theoretic objectives. These attacks are determined by solving a constrained optimization problem. Further, we proposed an algorithm to solve this optimization problem that will output adversarial data to fool greedy forward selection FS algorithms. We used publicly available datasets to benchmark our approach's efficacy and show that the generated attacks are transferable to other FS algorithms (i.e., black-box attacks). Finally, we discuss our approach's trade-offs when applied to a diverse set of benchmarks.

The rest of this paper is organized as follows: In Section 2 we review the related works in FS and adversarial machine learning, then we analytically propose an information-theoretic FS poisoning algorithm in Section 3. We present the experimental evaluation in Section 4 and draw the conclusion in Section 5.

2. Related works

2.1. Information-theoretic feature selection

Information-theoretic FS uses a classifier-independent score (e.g., KL divergence, mutual information, etc.) to evaluate a feature subset's utility. There are many information-theoretic FS methods proposed based on different trade-offs between features' relevancy with the class label and redundancy with other features. In this section, we revisit the state-of-the-art information-theoretic FS objectives.

In [26], Lewis et al. proposed to select the top K features with the largest mutual information (MIM) w.r.t. the class label. MIM assumes that all the features are independent. Unfortunately, this assumption is typically not true. Thus, MIM is generally sub-optimal because it fails to account for feature redundancy. In general, it is equally important to maximize feature relevancy and minimize feature redundancy. For example, Battiti et al. proposed to subtract a feature's mutual information w.r.t. previously selected features from its relevancy to penalize the redundancy. In [2], Battiti proposed the Mutual Infor-

mation Feature Selection (MIFS) method. In MIFS, the penalization of feature redundancy is controlled by a hyper-parameter $\gamma(\gamma > 0)$. The MIFS objective is given by:

$$J_{MIFS}(F_i) = I(F_i;Y) - \gamma \sum_{F_j \in \mathcal{S}} I(F_i;F_j) \tag{2}$$

In (2), the candidate feature F_i's score is the combination of its relevancy w.r.t. the class label (i.e., $I(F_i;Y)$) and its redundancy w.r.t. the previously selected features (i.e., $\sum_{F_j \in \mathcal{S}} I(F_i;F_j)$, $\mathcal{S}$ is the previously selected feature subset) controlled by $\gamma > 0$. Peng et al. further proposed that γ should be inversely proportional to the selected subset size: $\frac{1}{|\mathcal{S}|}$ (mRMR) [36], the result objective is given by (3).

$$J_{mRMR}(F_i) = I(F_i;Y) - \frac{1}{|\mathcal{S}|} \sum_{F_j \in \mathcal{S}} I(F_i;F_j) \tag{3}$$

The explicitly defined trade-off parameter γ, even the dynamic γ in mRMR, does not guarantee the best relevancy-redundancy trade-off and often leads to drastically different results. Therefore, Yang and Moody [42] implicitly leverage the relevance-redundancy trade-off by using Joint Mutual Information (JMI) to maximize a candidate feature's complementary information which is given by:

$$J_{JMI}(F_i) = \sum_{F_j \in \mathcal{S}} I(F_i, F_j; Y) \tag{4}$$

In [29], Lin and Tang show that in contrast to minimizing the feature redundancy, the conditional redundancy between unselected features and already selected features given the class label should also be maximized. Lin and Tang proposed the Conditional Informax Feature Extraction (CIFE) method given by:

$$J_{CIFE}(F_I) = I(F_i;Y) - \sum_{F_j \in \mathcal{S}} I(F_i;F_j) - I(F_i;F_j|Y) \tag{5}$$

Brown et al. proposed a unified framework based on Conditional Likelihood Maximization (CMI) that concludes various objectives published since 1992 [6]. The objective is given by (6). Brown et al. showed that various FS objectives are special cases of (6) fit with different β and γ. Brown et al. also categorized various information-theoretic FS objectives as linear and non-linear combinations of information-theoretic terms. Besides the above discussed linear objectives (e.g., mRMR, CMI, and JMI), there are also non-linear objectives that are popular.

$$J_{cmi}(F_i) = I(F_i;Y) - \gamma \sum_{F_j \in \mathcal{S}} I(F_i;F_j) + \beta \sum_{F_j \in \mathcal{S}} I(F_i;F_j|Y) \tag{6}$$

In [14], Fleuret proposed the Conditional Mutual Information Maximization (CMIM) which selects features that maximize their mutual information with the class label conditioned on selected features. It ensures the selection of features that are both individually informative and pairwise non-redundant. In [21], Jakulin proposed a similar heuristic Interaction Capping (ICAP) using the conditional redundancy term. The objectives are given by (7) and (8), respectively.

$$J_{CMIM}(F_i) = \min_{F_j \in \mathcal{S}} I(F_i;Y|F_j) = I(F_i;Y) - \max_{F_j \in \mathcal{S}} \{I(F_i;F_j) - I(F_i;F_j|Y)\} \tag{7}$$

$$J_{ICAP}(F_i) = I(F_i;Y) - \sum_{F_j \in \mathcal{S}} \max\{0, (I(F_i;F_j) - I(F_i;F_j|Y))\} \tag{8}$$

Unlike the heuristics discussed above, the Double Input Symmetrical Relevancy (DISR) criteria uses a normalization term on the information-theoretic terms to offset the inherent bias towards high arity features. Meyer and Bontempi proposed the DISR objective [32] which is given by:

$$J_{DISR}(F_i) = \sum_{F_j \in \mathcal{S}} \frac{I(F_i, F_j; Y)}{H(F_i, F_j, Y)} \tag{9}$$

Apart from the feature evaluation criteria, there is also the search procedure that is important. In practice, information-theoretic FS is performed using a *Greedy Forward Selection* as the intuitive exhaustive searching is NP-hard. In Algorithm 1, greedy forward selection algorithms initialize the selected subset $\mathcal{S}_0$ at step 0 as the feature with the maximal mutual information with class label Y. Then at each step t, given the previously selected subset $\mathcal{S}_{t-1}$ and the candidate subset $\mathcal{F}$, the candidate $F \in \mathcal{F}$ that maximizes $J(F;\mathcal{S}_{t-1})$ is selected and removed from $\mathcal{F}$. We chose the analyze the greedy feature set optimizer with information-theoretic objectives because of its wide use in feature selection.

Algorithm 1: Greedy Forward Selection

Input: FS objective $J(*)$; Number to select k; Feature set $\mathcal{F}$; Class label Y
Initialization: $\mathcal{S} = \varnothing$
$F_0 = \arg\max_{F \in \mathcal{F}} I(F; Y)$
$\mathcal{S}_0 = \{F_0\}$
for $t = 1 : k - 1$ **then**
  $F_t = \arg\max_{F \in \mathcal{F}} J(F; \mathcal{S}_{t-1})$
  $\mathcal{S}_t = \mathcal{S}_{t-1} \bigcup \{F_t\}$; $\mathcal{F} = \mathcal{F} \setminus F_t$
end for

2.2. Adversarial machine learning

Adversarial machine learning gained a significant amount of momentum in the field after the exploitation of perturbation attacks was easy to generate against deep learning [37]. However, the very first seminal works in adversarial machine learning date back to 2004, when Dalvi et al. [9], Lowd and Meek [31] studied adversaries in the environment of spam filtering. Huang et al. later proposed a taxonomy that categorizes different attacks from three axises [20]: *Influence* (contaminating training stage or testing stage), *Specificity* (targeted or indiscriminate attacks), *Security Violation* (*Integrity*, *Availability*, or *Privacy* violations). Following notations in Section 1, the attacks $\Phi(\mathcal{A})$ can be categorized into *poisoning* and *evasion* attacks based on which stage the attack takes place (*Influence*). A poisoning attack targets tampering the training performance of a classifier. The evasion attacks, on the other hand, seek to contaminate the testing data and evade detection for malicious data samples. Moreover, an adversary may have different levels of knowledge $\theta \in \Theta$ about the victim's machine learning pipeline (i.e., training/testing data, learning algorithm, hyper-parameters, etc.). Specifically, the knowledge varies from *Perfect Knowledge* (P.K.) to *Zero Knowledge* (Z.K.); the corresponding attacks are white-box attack and black-box attack, respectively (*Specificity*). There are three types of *Security Violations*. *Integrity* is violated if malicious behaviors are performed without compromising normal system operation. *Availability* is violated if the functionality of the system is compromised, causing a denial of service. *Privacy* is violated if the attacker is able to obtain information about the system's users by reverse-engineering the attacked system.

We now review contributions in adversarial machine learning. Over the past few years, perhaps the most famous and influential work is [37], where Szegedy et al. revealed the vulnerability of trained DNNs to imperceptible malicious perturbations in their input samples. Goodfellow et al. followed with a Fast Gradient Sign Method (FGSM), which approximates the cost function in the neighborhood of legitimate samples to allow for faster generation of adversarial samples [17]. Apart from the deep learning field, the classical adversarial machine learning field also attracted tremendous research attention and efforts. Kearns and Li explored the formalization of the worst-case error against learned binary classifiers and proposed methods for bounding the error rate tolerable by arbitrary learning methods [23]. Lowd and Meek proposed an attack algorithm against spam filters by reverse-engineering the linear classifiers [31]. In addition to supervised learning tasks, some unsupervised learning methods also face security issues in adversarial settings. For example, some obfuscation attacks can inject and hide the adversarial examples while keeping the clustering results unchanged [1].

There are also numerous counter-measures to defend against various attacks that vary from different perspectives (e.g., attack objective, knowledge of the victim model, etc.). In [3], Bhagoji et al. proposed to improve classifiers' robustness by balancing the information carried among features via PCA-based feature transformations. Similarly, Guo et al. adopted PCA for image analysis applications (e.g., total variance minimization, image quilting) to defend against image attacks (e.g., FGSM). In [20], Huang et al. proposed two heuristic rules (e.g., data sanitization, periodic re-training) to defend against poisoning attacks. However, these heuristic rules sometimes can be impractical and naïve. For example, periodic re-training is computationally expensive on large datasets. There are also generic strategies that improve the classifiers' security in the face of an adversary. For example, increasing the difficulty of poisoning/evasion. In [4], Biggio et al. proposed a model-robustness evaluation framework using intelligent data re-sampling. These generic defensive techniques are particularly useful when facing indiscriminate attacks such as black-box attack and transferable attack [35,34].

In a recent literature survey [5], Biggio et al. reviewed ten years of adversarial machine learning research. Specifically, Biggio et al. summarized the apparently-different contributions and proposed three model-independent *Golden Rules* as generic countermeasures against adversaries: *know your adversary*, *be proactive*, and *protect yourself*. These *Golden Rules* incorporate the "security by design" principle which proactively models potential attacks to improve model robustness.

2.3. Adversarial feature selection

FS is an important stage in machine learning pipelines to reduce the classifier complexity and alleviate the "curse of dimensionality". In security-sensitive applications, both the classification stage and the FS stage face security risks. However, despite the extensive contributions made to improve the classifiers' security, the security concern of FS did not draw the attention of researchers until recent years.

Li and Vorobeychik demonstrated the severe shortcoming in FS when facing feature cross-substitution attack in an adversarial setting [27]. Wang et al. grouped different attack algorithms against feature selection as dense and sparse attacks [39]. The authors showed that sparse feature attacks are best defended by classifiers that use ℓ_1 regularization. Li et al. formulated the attack algorithm against Lasso as a bi-level optimization problem [28]. To account for the associated cumbersome computational complexity, the authors resorted to the projected gradient descent method. The proposed attack algorithm circumvents the non-differentiability of the ℓ_1 norm as the authors rearranged the attack model as a linear inequality constrained quadratic programming problem. Adversarial training has been shown to slightly beneficial to linear FS models such as Lasso [12].

Xiao et al. analytically revealed the underlying risk of the embedded FS algorithms (e.g., Lasso, Elastic-net) by designing a targeted poisoning algorithm against Lasso [41]. Experiments demonstrated that the malicious samples can result in a significant classification accuracy degradation. The poisoning algorithm is designed according to the attack formulation (1) proposed by Biggio et al. Specifically, the attacker seeks to maximize Lasso's loss by including carefully crafted malicious samples. The malicious sample $\mathbf{x}_a$ is generated by maximizing:

$$\max_{\mathbf{x}_a \in \mathbb{R}^d} \mathcal{W} = \frac{1}{|\mathcal{D}|+1} \sum_{\mathbf{x}_j \in \mathcal{D} \bigcup \{\mathbf{x}_a\}} \ell\big(y_j, f(\mathbf{x}_j)\big) + \lambda \Omega(\omega) \tag{10}$$

In (10), $f(\mathbf{x}_j)$ is the predicted class label of $\mathbf{x}_j$. Note that $\sum \ell\big(y_j, f(\mathbf{x}_j)\big)$ is the summed losses on $\mathcal{D} \bigcup \{\mathbf{x}_a\}$ ($\mathcal{D}$ is the original dataset) and depends on the malicious sample $\mathbf{x}_a$. $\lambda\Omega(\omega)$ is the penalization on model coefficients ω (i.e., $\|\omega\|_1$).

While Xiao et al.'s method can successfully downgrade Lasso's accuracy, experiments showed that the adversarial samples can be easily detected. Frederickson et al. modified Xiao's objective by adding a penalization term ($\phi\Lambda(x_j)$) to minimize the distance between malicious and benign samples [15]. Frederickson et al.'s objective is given by:

$$\max_{\mathbf{x}_j} \mathcal{W}\prime = \frac{1}{|\mathcal{D}|+1} \sum_{\mathcal{D} \bigcup \{\mathbf{x}_a\}} \ell\big(y_j, f(\mathbf{x}_j)\big) + \lambda\Omega(\omega) - \phi\Lambda(x_j) \tag{11}$$

As methods are being designed to attack FS, there is also research to build FS methods that are also resilient against adversaries. In [44], Zhang et al. proposed a secure wrapper FS technique that can be adversary-aware in the testing stage by maximizing the generalization and security objectives simultaneously. The objective is given in (12). The author incorporated a distance metric to quantify a feature's robustness (i.e., the hardness of evasion).

$$\pi^* = \arg\max_{\pi \in \{0,1\}^d} G(\pi) + \lambda S(\pi), \;\; s.t. \sum_{k=1}^{d} \pi_k = m \tag{12}$$

In (12), $\pi \in \{0,1\}^d$ is a binary-valued d-dimensional vector representing whether each feature has been selected or not. $G(\pi)$ and $S(\pi)$ represent classifier's generalization power and security to evasion, respectively. m is selected feature number.

Motivated by a similar concept, Wu and Li enhanced the mRMR feature selection method's security against evasion by jointly considering the features' usefulness to prediction and robustness against evasion [40]. The feature's robustness is measured by the distance between a malicious sample and its M nearest benign neighbors (M is a user-defined term).

3. Filter feature selection poisoning

The embedded and wrapper FS methods mostly rely on optimization theory to solve for the best feature subset. The information-theoretic filter FS, on the other hand, measures the feature subset's utility via a mutual information score. Note that the mutual information works on Random Variable (R.V.) instead of scalars. In this section, we first propose two algorithms that allow us to manipulate mutual information between R.V.'s using constrained optimization. Then we design a poisoning algorithm against information-theoretic FS. Note that most estimators evaluate the mutual information quantity on discretized R.V. values [6]. Thus, we here incorporate the same R.V. discretization assumption.

3.1. Manipulation of mutual information: problem setting

Given two Random Variables (R.V.'s): F and Y, the marginal outcome spaces are defined as $f \in \mathcal{F}$ and $y \in \mathcal{Y}$, respectively (conventionally, probability theory uses "sample space", we here use "outcome space" to avoid confusion with terms such as "data sample"). Similarly, we denote the joint outcome space for (F, Y) as $\Psi = \{(f, y) : (f, y) \in \mathcal{F} \otimes \mathcal{Y}, \mathbb{P}(f, y) \neq 0\}$. Note that Ψ excludes the zero-probability joint outcomes in $\mathcal{F} \otimes \mathcal{Y}$. Using the above notations, the entropy-based Mutual Information (MI) of (F, Y) is calculated as: $I(F;Y) = H(F) + H(Y) - H(F,Y)$ where entropies are calculated based on probabilities: $H(F) = -\sum_{f \in \mathcal{F}} \mathbb{P}(f) \log \mathbb{P}(f), H(F,Y) = -\sum_{(f,y) \in \Psi} \mathbb{P}(f,y) \log \mathbb{P}(f,y)$ [8].

Our first task is that we seek to manipulate (decrease or increase) the MI of (F, Y) by injecting malicious data samples $\hat{\mathcal{D}}$ to the training dataset $\mathcal{D}$. Let N denote the training dataset size (i.e., $D_i \in \mathcal{D}, 1 \leqslant i \leqslant N$). We use $D_i \in \widehat{\mathcal{D}}, i > N$ to index the malicious data samples. Note that although $\mathcal{D}$ and $\widehat{\mathcal{D}}$ consist of many features, we now consider only feature F and class label Y. We require the injected data samples satisfy the following constraints: (a) the marginal outcomes of F and Y in $\widehat{\mathcal{D}}$ are from $\mathcal{F}$

and $\mathcal{Y}$, respectively; (b) the joint outcomes of (F, Y) in $\widehat{\mathcal{D}}$ are from Ψ. The reasoning for the constraints is that the feature selector can deem an unrecognized marginal/joint outcome as an impossible outcome based on *a priori* knowledge. We formulate the above constraints mathematically:

- The marginal outcome spaces in malicious data $\widehat{\mathcal{D}}$ for F and Y are denoted as $\mathcal{F}_2$ ($\mathcal{F}_2 \subset \mathcal{F}$) and $\mathcal{Y}_2$ ($\mathcal{Y}_2 \subset \mathcal{Y}$), respectively. The complementary subsets are: $\mathcal{F}_1 = \mathcal{F} \setminus \mathcal{F}_2, \mathcal{Y}_1 = \mathcal{Y} \setminus \mathcal{Y}_2$.
- (F, Y)'s joint outcome space in $\widehat{\mathcal{D}}$ is Ψ.

3.2. Manipulation of mutual information

We now analytically derive a solution for the first task. As the MI is calculated probabilistically, we first analyze how the distributions and MI are influenced by injecting malicious data samples $\widehat{\mathcal{D}}$. Then we shed light on how to determine $\widehat{\mathcal{D}}$ such that MI can be manipulated. We use $\mathbb{P}(\cdot)$ and $\mathbb{P}\prime(\cdot)$ to denote the probabilities before/after $\widehat{\mathcal{D}}$ is injected, respectively (i.e., $\mathbb{P}(\cdot)$ for $\mathcal{D}$ and $\mathbb{P}\prime(\cdot)$ for $\mathcal{D} \bigcup \widehat{\mathcal{D}}$).

We use ρ to denote the number of injected malicious data samples ($D_i \in \widehat{\mathcal{D}}, N < i \leqslant N + \rho$). Let $\lambda(\rho) = \frac{N}{N+\rho}$, then we have $\lambda(\rho) \leqslant 1$ because $\rho \geqslant 0$. To simplify notations in later sections, we use λ directly to imply it is a function of ρ. We first express marginal probabilities $\mathbb{P}\prime(\cdot)$'s for $f \in \mathcal{F}_1, y \in \mathcal{Y}_1$ in terms of $\mathbb{P}(\cdot)$'s after ρ malicious data samples injected (note that marginal outcomes $f \in \mathcal{F}_1, y \in \mathcal{Y}_1$ are not used in $\widehat{\mathcal{D}}$):

- $\forall f \in \mathcal{F}_1, \mathbb{P}'(f) = \frac{N\mathbb{P}(f)}{N+\rho} = \lambda\mathbb{P}(f)$
- $\forall y \in \mathcal{Y}_1, \mathbb{P}'(y) = \frac{N\mathbb{P}(y)}{N+\rho} = \lambda\mathbb{P}(y)$

To calculate the MI of (F, Y) after injection, we still need the distributions $\mathbb{P}'(\cdot)$ for $(f, y) \in \Psi$ and $f \in \mathcal{F}_2, y \in \mathcal{Y}_2$ after injection. Let $I'(F; Y)$ represent the MI after $\mathcal{D}$ has been poisoned. Because we require the joint outcome spaces are identical between $\mathcal{D}$ and $\widehat{\mathcal{D}}$ in our setting, therefore, the first task's goal is equivalent to solving for injected number m_i for each joint outcome $(f_i, y_i) \in \Psi$. Note that i is the unique index for each $(f_i, y_i) \in \Psi$, thus, $1 \leqslant i \leqslant |\Psi|$. Moreover, m_i satisfies the constraints: $0 \leqslant m_i \leqslant \rho, \sum_{1 \leqslant i \leqslant |\Psi|} m_i = \rho$. Mathematically, the joint probability of the joint outcome $(f_i, y_i) \in \Psi$ is re-scaled to $\mathbb{P}'(f_i, y_i) = \frac{\mathbb{P}(f_i, y_i) * N + m_i}{N+\rho}$ after injection.

For a marginal outcome $\forall f \in \mathcal{F}_2$, the probability is re-scaled to $\mathbb{P}'(f) = \frac{\mathbb{P}(f) * N + \sum_{i \in T_f} m_i}{N+\rho}$ where $T_f = \{i : f_i = f, \ \forall (f_i, y_i) \in \Psi\}$. Similarly, for the marginal outcome $\forall y \in \mathcal{Y}_2$, the probability is re-scaled to $\mathbb{P}'(y) = \frac{\mathbb{P}(y) * N + \sum_{i \in T_y} m_i}{N+\rho}$ where $T_y = \{i : y_i = y, \ \forall (f_i, y_i) \in \Psi\}$. We here intuitively explain T_f, T_y. In Table 1, we give individual $(f_i, y_i) \in \Psi$ and corresponding m_i. As we can see, although the rows (f_i, y_i)'s are unique (i.e., $\forall (f_i, y_i), (f_j, y_j) \in \Psi, 1 \leqslant i, j \leqslant |\Psi| : \ i \neq j \iff (f_i, y_i) \neq (f_j, y_j)$), the individual columns may contain duplicate f_i's or y_i's (e.g., $\exists (f_i, y_i), (f_j, y_j) \in \Psi, 1 \leqslant i, j \leqslant |\Psi|, i \neq j : \ f_i = f_j$ or $y_i = y_j$). Therefore, T_f and T_y collects the m_i's such that $f_i = f$ ($y_i = y$) where $(f_i, y_i) \in \Psi$.

With all the probabilities $\mathbb{P}'(\cdot)$ calculated, we can explicitly express the individual and joint entropies in the following equations.

$$H'(F) = -\sum_{f \in \mathcal{F}_1} \lambda\mathbb{P}(f) \log(\lambda\mathbb{P}(f)) - \sum_{f \in \mathcal{F}_2} \mathbb{P}'(f) \log \mathbb{P}'(f) \tag{13}$$

$$= -\sum_{f \in \mathcal{F}_1} \lambda\mathbb{P}(f) \log(\lambda\mathbb{P}(f)) - \sum_{f \in \mathcal{F}_2} \frac{\mathbb{P}(f) * N + \sum_{i \in T_f} m_i}{N+\rho} \log \frac{\mathbb{P}(f) * N + \sum_{i \in T_f} m_i}{N+\rho} \tag{14}$$

$$H'(Y) = -\sum_{y \in \mathcal{Y}_1} \lambda\mathbb{P}(y) \log(\lambda\mathbb{P}(y)) - \sum_{y \in \mathcal{Y}_2} \mathbb{P}'(y) \log \mathbb{P}'(y)$$

$$= -\sum_{y \in \mathcal{Y}_1} \lambda\mathbb{P}(y) \log(\lambda\mathbb{P}(y)) - \sum_{y \in \mathcal{Y}_2} \frac{\mathbb{P}(y) * N + \sum_{i \in T_y} m_i}{N+\rho} \log \frac{\mathbb{P}(y) * N + \sum_{i \in T_y} m_i}{N+\rho} \tag{15}$$

$$H'(F, Y) = -\sum_{(f_i, y_i) \in \Psi} \frac{\mathbb{P}(f_i, y_i) * N + m_i}{N+\rho} \log \frac{\mathbb{P}(f_i, y_i) * N + m_i}{N+\rho}$$

Table 1
Injected $(f_i, y_i) \in \Psi$.

f_1	y_1	m_1
f_2	y_2	m_2
...	...	...
$f_{\lvert\Psi\rvert}$	$y_{\lvert\Psi\rvert}$	$m_{\lvert\Psi\rvert}$

In the above derivations, we express the entropies after injection in terms probabilities before injection and m_i's ($1 \leqslant i \leqslant |\Psi|$) for a given ρ. Now we formulate the objective in (16) where σ takes values in $\{1, -1\}$, 1 indicates (16) is a maximization and vice versa for a given ρ. We can see our objective is a constrained optimization problem.

$$\begin{aligned} &\arg \max_{m_1, m_2, \cdots, m_{|\Psi|}} \left(H'(F) + H'(Y) - H'(F,Y)\right) * \ \sigma \\ &\text{s.t.} \sum_{1 \leqslant i \leqslant |\Psi|} m_i = \rho, \ 0 \leqslant m_i \leqslant \rho \end{aligned} \tag{16}$$

3.3. Manipulation of two mutual information

We now consider a new task consisting of two features F, F_p (subscript p indicates $F_p \neq F$) and class label Y where $I(F_p; Y) > I(F; Y)$ on training dataset $\mathcal{D}$. Our task is to maximize the difference $I(F;Y) - I(F_p;Y)$ by injecting malicious data samples $\widehat{\mathcal{D}}$ to training dataset $\mathcal{D}$. Similarly, the marginal outcome spaces for F, Y and F_p are defined as $f \in \mathcal{F}, y \in \mathcal{Y}$ and $f_p \in \mathcal{F}_p$, respectively. The joint outcome space of (F,Y) is $(f,y) \in \Psi, \Psi = \{(f,y) : (f,y) \in \mathcal{F} \otimes \mathcal{Y}, \mathbb{P}(f,y) \neq 0\}$. The joint outcome space of (F_p, Y) is $(f_p, y) \in \Phi, \Phi = \{(f_p, y) : (f_p, y) \in \mathcal{F}_p \otimes \mathcal{Y}, \mathbb{P}(f_p, y) \neq 0\}$. Using information theory, we have a simplified objective (17):

$$\begin{aligned} I(F;Y) - I(F_p;Y) &= H(F) + H(Y) - H(F,Y) - \left(H(F_p) + H(Y) - H(F_p,Y)\right) \\ &= H(F) - H(F,Y) - H(F_p) + H(F_p,Y) \end{aligned} \tag{17}$$

We here follow the same strategy and notations (e.g., $\widehat{\mathcal{D}}, \mathcal{D}, N, \rho, \lambda, \mathbb{P}(\cdot), \mathbb{P}'(\cdot)$, and m_i) from previous discussion. We require the injected data samples to satisfy the following constraints: (a) the marginal outcomes of F, F_p, and Y in $\widehat{\mathcal{D}}$ are from $\mathcal{F}, \mathcal{F}_p$, and $\mathcal{Y}$, respectively, (b) the triple outcomes of (F, F_p, Y) in $\widehat{\mathcal{D}}$ is from $\Delta = \{(f, f_p, y) \in \mathcal{F} \otimes \mathcal{F}_p \otimes \mathcal{Y} : \mathbb{P}(f,y) \neq 0 \text{ or } \mathbb{P}(f_p, y) \neq 0\}$. Note that although $D_i \in \widehat{\mathcal{D}}$ consists of three R.V.'s, the second constraint is in terms of joint probabilities of two R.V.'s (F and Y, or F_p and Y). The second constraint is to reduce the solution space and the detectability of generated malicious data samples. The above constraints can be formulated mathematically as follows:

- The marginal outcome spaces of F, F_p and Y in $\widehat{\mathcal{D}}$ are denoted as $\mathcal{F}_2$ ($\mathcal{F}_2 \subset \mathcal{F}$), $\mathcal{F}_{p2}$ ($\mathcal{F}_{p2} \subset \mathcal{F}_p$) and $\mathcal{Y}_2$ ($\mathcal{Y}_2 \subset \mathcal{Y}$), respectively. The complementary subsets are: $\mathcal{F}_1 = \mathcal{F} \setminus \mathcal{F}_2, \mathcal{F}_{p1} = \mathcal{F}_p \setminus \mathcal{F}_{p2}, \mathcal{Y}_1 = \mathcal{Y} \setminus \mathcal{Y}_2$.
- (F, F_p, Y)'s triple outcome in $\widehat{\mathcal{D}}$ is from $\Delta = \{(f, f_p, y) \in \mathcal{F} \otimes \mathcal{F}_p \otimes \mathcal{Y} : \mathbb{P}(f,y) \neq 0 \text{ or } \mathbb{P}(f_p, y) \neq 0\}$.

We now solve for the distributions after injection. For the marginal outcomes $\forall f \in \mathcal{F}_1, \forall f_p \in \mathcal{F}_{p1}$, the probabilities $\mathbb{P}'(\cdot)$ after injection can be solved similarly as in previous discussion (i.e., simply re-scaled by λ).

- $\forall f \in \mathcal{F}_1, \mathbb{P}'(f) = \frac{N\mathbb{P}(f)}{N+\rho} = \lambda \mathbb{P}(f)$
- $\forall f_p \in \mathcal{F}_{p1}, \mathbb{P}'(f_p) = \frac{N\mathbb{P}(f_p)}{N+\rho} = \lambda \mathbb{P}(f_p)$

For the rest marginal and joint distributions after injection, we follow the same strategy from the discussion in Section 2-B. The goal here is equivalent to solving for the corresponding injected numbers m_i's for each $(f_i, f_{pi}, y_i) \in \Delta$. Thus, m_i satisfies $0 \leqslant m_i \leqslant \rho$ and $1 \leqslant i \leqslant |\Delta|$. Therefore, for the marginal outcome $\forall f \in \mathcal{F}_2$, the probability after injection is re-scaled to $\mathbb{P}'(f) = \frac{\mathbb{P}(f) * N + \sum_{i \in T_f} m_i}{N+\rho}$ where $T_f = \{i : f_i = f, \ \forall (f_i, f_{pi}, y_i) \in \Delta\}$. Similarly for $\forall f_p \in \mathcal{F}_{p2}$, the probability is re-scaled to $\mathbb{P}'(f_p) = \frac{\mathbb{P}(f_p) * N + \sum_{i \in T_{f_p}} m_i}{N+\rho}$ where $T_{f_p} = \{i : f_{pi} = f_p, \ \forall (f_i, f_{pi}, y_i) \in \Delta\}$.

We now solve the joint distributions. For the joint outcome $(f,y) \in \Psi$, the joint probability is re-scaled to $\mathbb{P}'(f,y) = \frac{\mathbb{P}(f,y) * N + \sum_{i \in T_{f,y}} m_i}{N+\rho}$ where $T_{f,y} = \{i : y_i = y, f_i = f, \ \forall (f_i, f_{pi}, y_i) \in \Delta\}$. Similarly, for the joint outcome $(f_p, y) \in \Phi$, the joint probability is re-scaled to $\mathbb{P}'(f_p, y) = \frac{\mathbb{P}(f_p, y) * N + \sum_{i \in T_{f_p,y}} m_i}{N+\rho}$ where $T_{f_p,y} = \{i : y_i = y, f_{pi} = f_p, \ \forall (f_i, f_{pi}, y_i) \in \Delta\}$. $T_f, T_{f_p}, T_{f,y}$, and $T_{f_p,y}$ are all defined similarly as in previous subsection (see Table 2).

With all the distributions $\mathbb{P}'(\cdot)$ after injection calculated, we can explicitly express the individual and joint entropies in the following equations.

$$H'(F) = -\sum_{f \in \mathcal{F}_1} \lambda \mathbb{P}(f) \log(\lambda \mathbb{P}(f)) - \sum_{f \in \mathcal{F}_2} \mathbb{P}'(f) \log \mathbb{P}'(f) \tag{18}$$

Table 2
Injected $(f_i, f_{pi}, y_i) \in \Delta$.

f_1	f_{p1}	y_1	m_1
f_2	f_{p2}	y_2	m_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$f_{\lvert\Delta\rvert}$	$f_{p\lvert\Delta\rvert}$	$y_{\lvert\Delta\rvert}$	$m_{\lvert\Delta\rvert}$

$$= -\sum_{f\in\mathcal{F}_1}\lambda\mathbb{P}(f)\log(\lambda\mathbb{P}(f)) - \sum_{f\in\mathcal{F}_2}\frac{\mathbb{P}(f)*\ N+\sum_{i\in T_f} m_i}{N+\rho}\log\frac{\mathbb{P}(f)*\ N+\sum_{i\in T_f} m_i}{N+\rho} \tag{19}$$

$$\begin{aligned} H'(F_p) &= -\sum_{f_p\in\mathcal{F}_{p1}}\lambda\mathbb{P}(f_p)\log(\lambda\mathbb{P}(f_p)) - \sum_{f_p\in\mathcal{F}_{p2}}\mathbb{P}'(f_p)\log\mathbb{P}'(f_p) \\ &= -\sum_{f_p\in\mathcal{F}_{p1}}\lambda\mathbb{P}(f_p)\log(\lambda\mathbb{P}(f_p)) - \sum_{f_p\in\mathcal{F}_{p2}}\frac{\mathbb{P}(f_p)*\ N+\sum_{i\in T_{f_p}} m_i}{N+\rho}\log\frac{\mathbb{P}(f_p)*\ N+\sum_{i\in T_{f_p}} m_i}{N+\rho} \end{aligned} \tag{20}$$

$$H'(F,Y) = -\sum_{(f,y)\in\Psi}\frac{\mathbb{P}(f,y)N+\sum_{i\in T_{f,y}} m_i}{N+\rho}\log\frac{\mathbb{P}(f,y)N+\sum_{i\in T_{f,y}} m_i}{N+\rho}$$

$$H'(F_p,Y) = -\sum_{(f_p,y)\in\Phi}\frac{\mathbb{P}(f_p,y)N+\sum_{i\in T_{f_p,y}} m_i}{N+\rho}\log\frac{\mathbb{P}(f_p,y)N+\sum_{i\in T_{f_p,y}} m_i}{N+\rho} \tag{21}$$

Now we formulate the new task as a constrained optimization in objective (22).

$$\begin{aligned} &\arg\max_{m_1,m_2,\cdots,m_{\lvert\Delta\rvert}} \left(H'(F) - H'(F_p) - H'(F,Y) + H'(F_p,Y)\right) \\ &s.t.\ \sum_{1\leqslant i\leqslant\lvert\Delta\rvert} m_i = \rho,\ 0\leqslant m_i\leqslant\rho \end{aligned} \tag{22}$$

The constrained optimization problems in (16) and (22) are not guaranteed to be convex. The optimization can be solved using quadratic programming. Specifically, we first use the interior-point method (available in python and Matlab) to solve m_i's in the continuous range, then we round the results.

3.4. Pseudo code

For the two mutual information manipulation tasks from previous sections, we propose two algorithms Mutual Information Poisoning (MIP) and Dual Mutual Information Poisoning (DMIP), respectively.

Algorithm 2: MIP pseudo-code

Inputs: R.V.'s F, Y of length N.
Hyper-parameters: Operation $\sigma = +1/-1$, Injection sample limit π.
Goal: $\sigma * \left(I(F';Y') - I(F;Y)\right) \geqslant \delta$.
Initialization: Calculate $\mathbb{P}(f) : \forall f \in \mathcal{F}, \mathbb{P}(y) : \forall y \in \mathcal{Y}, \mathbb{P}(f,y) : \forall (f,y) \in \Psi$, Origin = $I(F;Y), j = 1$.
Variable Initialization: Optimization variable $\vec{\mathbf{M}} = [m_1, m_2, \cdots]$, Initial value $\vec{\mathbf{M}}_0 = [0, 0, \cdots]$.
for $j \leqslant \pi$ **do**
  $F', Y' = \arg\max_{\vec{\mathbf{M}}} (H'(F) + H'(Y) - H'(F,Y)) * \ \sigma$
    s.t. $\sum_{1\leqslant i\leqslant\lvert\Psi\rvert} m_i = j,\ 0 \leqslant m_i \leqslant j$
  if $\lvert I(F';Y') - Origin\rvert \geqslant \delta$ **then**
    Return F', Y', j
    Break
  end if
  $j += 1$
end for
Outputs: Poisoned R.V.'s F', Y' **and budget spent** j.

Algorithm 3: DMIP pseudo-code

1: **Inputs**: R.V.'s F, F_p, Y of length N.
2: **Hyper-parameters**: Injection sample limit π.
3: **Goal**: $I(F';Y') - I(F_p';Y') > 0$
4: **Initialization**: Calculate $\mathbb{P}(f)$ for $f \in \mathcal{F}$, $\mathbb{P}(y)$ for $y \in \mathcal{Y}$, $\mathbb{P}\left(f_p\right)$ for $f_p \in \mathcal{F}_p$, $\mathbb{P}(f,y)$ for $(f,y) \in \Psi$, $\mathbb{P}\left(f_p,y\right)$ for $\left(f_p,y\right) \in \Phi, j = 1$.
5: **Variable Initialization**: Optimization variable $\vec{\mathbf{M}} = [m_1, m_2, \cdots]$, Initial value $\vec{\mathbf{M}}_0 = [0, 0, \cdots]$.
6: **for** $j \leqslant \pi$ **do**
7: $F', F_p', Y' =$
8: $\arg\max_{\vec{\mathbf{M}}} H'(F) - H'(F_p) - H'(F,Y) + H'(F_p,Y)$
9: **s.t.** $\sum m_i = j,\ 0 \leqslant m_i \leqslant j$
10: **if** $I(F';Y') - I(F_p';Y') > 0$ **then**
11: Return F', F_p', Y', j
12: Break
13: **end if**
14: $j += 1$
15: **end for**
16: **Outputs**: Poisoned R.V.'s F', F_p', Y' and budget spent j.

We give the MIP and DMIP pseudo codes in Algorithms 2 and 3, respectively. In Algorithm 2, the MIP algorithm takes in a feature F and class label Y. Each of these inputs are of length N. MIP outputs a poisoned feature F', its corresponding label Y' and the budget j. Note the budget is the number of samples needed to poison the feature and class label. MIP also has hyper-parameters that need to be chosen by the user. First there is the sign σ that takes value $+1$ or -1 and sign indicates increasing or decreasing $I(F;Y)$, respectively. π is the maximal injected sample number allowed, i.e., the length of output F' and Y' is less or equal to $N + \pi$. The last part of Algorithm 2 is the level of the desired manipulation which is denoted by δ. Note that although δ is a user-defined term, δ is the algorithm's goal instead of a hyper-parameter as it is not a component of MIP algorithm (see (16)). In Algorithm 3, DMIP takes in two features F, F_p and class label Y of length N and outputs the poisoned F' feature F_p' and label Y' as well as the budget used j. The hyper-parameter π is the maximal injection sample number allowed, i.e., the length of output F', F_p', Y' is less or equal to $N + \pi$.

We only explain the MIP algorithm in detail, since MIP and DMIP are formulated similarly. In step 1, MIP takes in the following inputs: R.V.'s F and Y of length N; desired MI operation σ ($+1/-1$ means increase/decrease); level of desired manipulation δ; injection budget π. In step 2, the original probability distributions $\mathbb{P}(\cdot)$ are calculated; the *Origin* defines the original MI $I(F;Y)$. In step 3, we initialize the target variables $\vec{\mathbf{M}} = [m_1, m_2, \cdots]$ and its initial value is set to $[0, 0, \cdots]$. The injection budget π is usually given arbitrarily. Thus, we want to find the minimal cost $j \leqslant \pi$ such that the objective is optimized. In step 4, we start at $j = 1$ to optimize the objective (step 5, 6) and increment j by 1 (step 11) at every step until the mutual information is increased/decreased by a desired manipulation level δ (step 7). Then we return the poisoned R.V.'s F', Y' and the budget j (step 8). We then stop the process (step 9).

Here we use a toy example to show MIP and DMIP algorithms' utility on randomly generated vectors F, F_p, and Y of length 1000 from a uniform distribution. We also demonstrate the effect of varying hyper-parameters. In Fig. 1a and 1b, the y-axis is the poisoned individual MI $I(F;Y)$ or the difference $I(F;Y) - I(F_p;Y)$, respectively. The x-axis is the incremental injection sample number $j = (10 \sim 200)$ where the injection sample limit π is arbitrarily chosen as $\pi = 200$.

In Fig. 1a, we use $I^+(F;Y)$ and $I^-(F;Y)$ to denote the increased or decreased MI by MIP algorithm where $\sigma = +1$ or -1, respectively. We use a horizontal dashed line to denote δ which is the algorithm's stop threshold determined by user. As we observed in Fig. 1a, $I^+(F;Y)$ is progressively increased for $\sigma = +1$ as j increases. Thus, if we increase $\pi, I^+(F;Y)$ will eventually converge to its theoretical upper bound $\min\{H(F), H(Y)\}$. We also observed that $I^-(F;Y)$ is progressively decreased for $\sigma = -1$ and $I^-(F;Y)$ reaches the lower bound 0 at $j = 50$. Thus, increasing π after $\pi = 50$ will not change $I^-(F;Y)$. In Fig. 1b, the difference $I(F;Y) - I(F_p;Y)$ is increased by DMIP algorithm. As we observed in Fig. 1b, $I(F;Y)$ is driven up and $I(F_p;Y)$ is decreased when the difference $I(F;Y) - I(F_p;Y)$ is maximized as j increases. Moreover, if we increase $\pi, I(F;Y) - I(F_p;Y)$ will eventually converge to its theoretical upper bound $\min\{H(F), H(Y)\}$.

We briefly discuss the computational complexity of the MIP and DMIP algorithms. In each algorithm, we iteratively solve a nonlinear constrained optimization (Eqs. (16), (22))) to find the minimal j malicious data samples when the objective function is maximized. We solve the constrained optimization using the popular interior-point method which is a repetitive procedure (see [22] for details). In the interior-point method, the algorithm iterates $O(\sqrt{n}\log\frac{n\mu_0}{\epsilon})$ steps where each step performs a newton method and parameter updating. Here n is the number of variables to optimize in (16) and (22) and

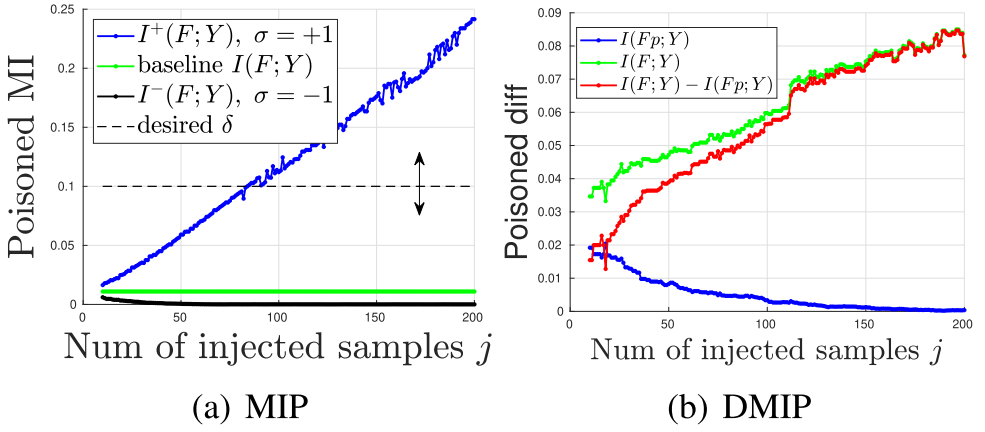

Fig. 1. The utility of MIP and DMIP algorithms. Fig. 1a shows the MI is poisoned as more malicious samples injected. In Fig. 1b, we can see as more malicious samples injected, $I(F;Y)$ is driven up and $I(F_p;Y)$ is dragged down when the difference $I(F;Y)-I(F_p;Y)$ is maximized.

$n=|\vec{\mathbf{M}}|$, μ_0 is the intermediate barrier parameter's initial value, and ϵ is a constant where the final objective function value after $O(\sqrt{n}\log\frac{n\mu_0}{\epsilon})$ steps is within ϵ region of the optimal value. Therefore, the time complexity of the constrained optimization is $O(C\sqrt{n}\log\frac{n\mu_0}{\epsilon})|_{n=|\vec{\mathbf{M}}|}$ where C is the constant time for each interior-point method iteration, and the time complexity of the MIP and DMIP algorithms is $O(C\pi\sqrt{n}\log\frac{n\mu_0}{\epsilon})|_{n=|\vec{\mathbf{M}}|}$ (π is the injection limit and $0<j\leqslant\pi$).

It is worth noting that the variable number satisfies that $|\vec{\mathbf{M}}|\leqslant|\mathcal{F}\otimes\mathcal{Y}|$ in (16) and $|\vec{\mathbf{M}}|\leqslant|\mathcal{F}\otimes\mathcal{F}_p\otimes\mathcal{Y}|$ in (22). These upper bounds on variable number can be quite large when the random variable's outcome space is large. Corresponding, we designed to neglect the joint outcomes with zero probabilities in the training data $\mathcal{D}$. Thus, the variable number and the time complexity are both reduced.

3.5. A filter FS poisoning algorithm

In the previous section, we proposed the MIP and DMIP algorithms to enable the manipulation of MI terms using constrained optimization. In this section, we present a poisoning algorithm that targets generic information-theoretic FS methods by incorporating the DMIP Algorithm. We refer to the poisoning algorithm as the DMIP poisoning algorithm.

Information-theoretic FS typically uses a greedy forward optimizer to avoid the intractable exhaustive search algorithm [6]. However, the greedy optimization has weaknesses that can be exploited by an adversary. For example, mistakes made at the beginning can lead to future selection mistakes because the information-theoretic FS evaluates candidates' redundancy w.r.t. the previously selected features. In the DMIP poisoning algorithm, we seek to poison the first selected feature by information-theoretic FS methods, which is simply the feature with the maximal MI.

Given the training dataset, assume F is an arbitrary feature and F_p is the feature having the maximal MI with class label Y, i.e., $F_p=\arg\max\{I(F;Y)\}$. If $F\neq F_p$, the poisoning algorithm's objective is to drive up $I(F;Y)$ and drag down $I(F_p;Y)$ simultaneously such that $I(F;Y)-I(F_p;Y)>0$ (i.e., $\delta=0$ in Algorithm 3). After the above poisoning objective is optimized, and the resulting malicious data samples are injected into the training dataset, F will be the first selected feature that we refer to as the "fake best" feature.

In the DMIP poisoning algorithm, we only poison two features (e.g., F and F_p) and the class label Y in the above poisoning objective. The remaining feature values are determined by randomly sampling data points from $\mathcal{D}$ with substituting F, F_p and Y with $\widehat{\mathcal{D}}$. In our experiment, we show such scheme can reduce the detectability of injected malicious data samples.

Adversarial data samples sometimes exhibit transferability across various machine learning models [17,34]. Information-theoretic FS algorithms use a greedy optimization that start with the feature that has the maximal MI with class label. Therefore, the malicious data samples generated by the DMIP poisoning algorithm are naturally transferable among generic information-theoretic FS algorithms. In our experiment, we also discuss the transferability.

4. Experiments

In this section, we present the evaluation of the DMIP poisoning algorithm. In the experiments, we first apply the DMIP poisoning algorithm on 16 benchmark datasets to generate the poisoned datasets. Note that the FS only happens in the training stage; thus, only the training portion of each benchmark is poisoned. Next, we run seven state-of-the-art information-theoretic FS algorithms (i.e., MIFS, mRMR, JMI, CIFE, CMIM, ICAP, and DISR) on each pair of benign and poisoned datasets. We then evaluate the DMIP poisoning algorithm's impact on the FS stage and the classification stage: First, the consistency

between the FS results from each pair of benign and poisoned datasets; Second, the testing classification error rates using KNN ($K = 5$) and decision tree.

Table 3 shows the benchmark datasets [6]. Note the benchmark datasets are discretized to perform the FS process. We use the discretized datasets from [6]. The discretization process has fixed-width bins. In this contribution, we assume a *Limited Knowledge (L.K.)* setting, i.e., the attacker has access to the training data. We select 15% of the features from each dataset. All experiments are averaged over 30 bootstrap trials. The algorithms are implemented in Matlab.

4.1. FS consistency and classification accuracy evaluation

We first report the consistencies between the FS results from each pair of benign and poisoned datasets. Note that the DMIP poisoning algorithm requires a manually chosen "fake best" feature to substitute the authentic best feature. We experiment with all features in feature space as the initially chosen "fake best" except the authentic best feature. For high-dimensional datasets with more than 500 features, we only use the features with the top-500 MI with the class label. For each benchmark, we report all of the consistency measurements associated with various initially chosen "fake best". We also report the malicious data sample injection number for each choice of "fake best". We use the Kuncheva index (range in $[-1, 1]$) to measure the degree of consistency [25]. Kuncheva stability/consistency index was used to measure the agreement between two FS results [25] and a high value implies high agreement. The Kuncheva index is a widely used measurement of FS algorithms' stability [11,33,24]. Mathematically, for two subsets $\mathcal{A} \subset \mathcal{F}$ and $\mathcal{B} \subset \mathcal{F}$, such that $m = |\mathcal{A} \cap \mathcal{B}|$ and $|\mathcal{A}| = |\mathcal{B}| = z$, where $1 \leqslant z \leqslant |\mathcal{F}| = p$, the Kuncheva index is formally given by:

$$\text{Consistency}(\mathcal{A}, \mathcal{B}) = \frac{mp - z^2}{z(p - z)} \in [-1, +1]$$

We report the consistency measurements for seven FS algorithms. In Fig. 2, we show the FS selection consistency (between benign and poisoned datasets) for each benchmark. In each plot, the x-axis represents the different "fake best" features descendingly sorted by their MI w.r.t. the class label Y. The left y-axis indicates the consistency measurements (a lower consistency indicates more poisoning impact), and the right y-axis is the poisoning ratio (poisoning data number over training data number). The first observation is that the poisoning ratio (i.e., number of malicious samples) increases as the "fake best" feature's MI decreases. The second observation is that, although the poisoning impacts (i.e., consistency measurements) on each benchmark vary across different "fake-best" features and FS algorithms, the DMIP poisoning algorithm can successfully contaminate each FS algorithm's selection result. Moreover, in our experiments, we require the poisoning ratio to be less than 1. Thus, for some "fake best" features whose required poisoning ratio exceeded the maximal poisoning budget, we simply put the corresponding consistency as 1 and poisoning ratio as 100% (e.g., "landsat", "penglung", "waveform").

In many machine learning pipelines, classifiers' accuracy is the desired objective. In the next experiment, we evaluate for seven state-of-the-art FS algorithms. We report the test classification error comparisons (with and without DMIP poisoning) after the FS stage using KNN ($K = 5$) and decision tree. Note that the DMIP leads to different poisoned FS consistencies for various "fake best" features used. Thus, in this experiment, the classification evaluation used the poisoned FS selected subset that leads to the minimal consistency. Table 4 presents the testing classification error comparisons for 16 benchmark datasets and seven FS algorithms, the number of malicious data samples are shown in parenthesis. We mark the failed poisoning cases in boldface. In Table 4, the first observation is that the DMIP poisoning algorithm can incur a higher test classification error on both classifiers, and this conclusion holds true for all seven FS algorithms. The second observation is that the classification error varies across different benchmark datasets. For example, "splice" and "krvskp" both have 1000+ adversarial samples, but the error increment for "splice" is higher than on "krvskp". The reasoning behind this result is that the filter FS doesn't account for the classification directly. Next, we use the Wilcoxon signed-rank sum test [10] to evaluate the significance of the experiment results. The Wilcoxon signed-rank sum test assumes commensurability of score differences and is usually safer to use as it does not assume normal distributions. Also, the Wilcoxon signed-rank sum test is insensitive to outliers [10]. In Table 4, we present the Wilcoxon signed-rank sum test statistics for each classification error comparison. As we can see, the test statistics are between $0 \sim 7$. For a confidence level of $\alpha = 0.01$ and 16 datasets, the difference between the

Table 3
Benchmark datasets.

Dataset	#features	#samples	Dataset	#features	#samples
lungcancer	56	32	pengcolon	2000	47
breast	30	427	penglung	325	55
congress	16	435	semeion	256	1195
heart	13	203	sonar	60	156
ionosphere	34	264	spect	22	201
krvskp	36	2397	splice	60	2382
landsat	36	4827	waveform	40	3750
parkinsons	22	147	wine	13	134

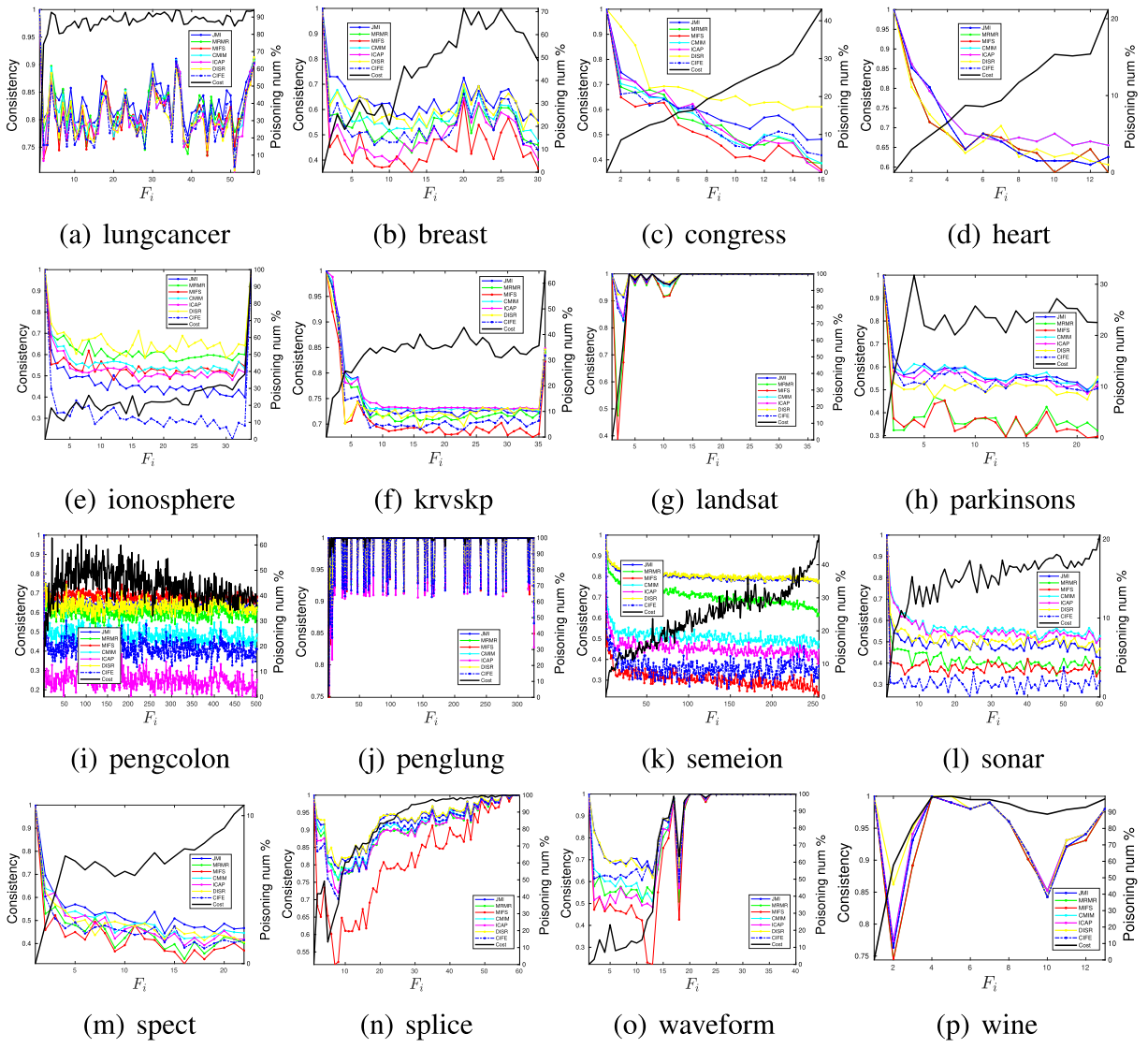

Fig. 2. We plot the consistency curves for 16 datasets and seven FS objectives when choosing various choices of "fake-best", we also give the corresponding injection number in the black curve.

classification errors is significant if the test statistic is equal to or less than 19. Therefore, the DMIP poisoning algorithm incurs a significantly higher classification error for seven FS objectives.

We also discuss the variational differences in performances while dealing with small, medium, and large datasets. Note that we can categorize the sizes of the benchmark datasets in Table 3 from the feature axis and sample axis. Suppose we choose to categorize from the feature axis. In that case, the benchmark datasets consist of three groups: (i) one large dataset that has 2000 features, (ii) two medium datasets that have $200 \sim 400$ features, (iii) 13 small datasets that have less than 100 features. In Fig. 2, we can observe that five datasets ("ionosphere", "pengcolon", "semeion", "sonar", "waveform") from all three groups have similar minimal poisoned FS consistency scores (about 0.2) as well as similar FS consistency score ranges. In Table 4, we can observe that four datasets ("lungcancer", "pengcolon", "splice", "wine") have similar maximal poisoned classification error increments (above 0.11) as well as similar error increase ranges for both KNN and decision tree classifiers. Although the above four datasets are not from the medium group, we do not deem this an outlier as the filter FS methods do not directly account for classification accuracy. Therefore, the poisoning impact levels on the FS consistency and classification accuracy are consistent and not influenced by the feature size. We can draw the same conclusion if we choose to categorize the benchmark datasets from the sample axis.

Table 4
The testing error rate comparisons for seven FS objectives: JMI, mRMR, MIFS, CIFE, CMIM, ICAP and DISR on 16 benchmark datasets, the malicious samples generated by DMIP generally incurs higher testing error, the exceptions are marked in boldface. The critical value of wilcoxon ranked sum test for 16 datasets is 19 with confidence level $\alpha = 0.01$.

Datasets	**KNN (N = 5)**													
	JMI		**mRMR**		**MIFS**		**CIFE**		**CMIM**		**ICAP**		**DISR**	
	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP
lungcancer	0.354	0.379(18)	0.371	0.496(19)	0.400	0.446(16)	0.408	0.483(17)	0.367	0.467(19)	0.413	0.463(17)	0.317	0.446(18)
breast	0.058	0.072(182)	0.050	0.069(143)	0.071	0.098(203)	0.077	**0.077(148)**	0.047	0.065(175)	0.070	0.073(146)	0.065	0.068(149)
congress	0.046	0.069(130)	0.046	0.123(126)	0.048	0.134(118)	0.044	0.078(126)	0.046	0.112(133)	0.046	0.108(134)	0.052	0.070(118)
heart	0.250	0.266(40)	0.250	0.305(38)	0.250	0.305(38)	0.250	0.266(40)	0.252	0.265(40)	0.252	0.265(40)	0.261	0.304(39)
ionosphere	0.107	0.121(81)	0.103	0.128(93)	0.118	0.127(55)	0.111	0.128(92)	0.106	0.119(70)	0.113	0.119(68)	0.098	0.115(96)
krvskp	0.070	0.080(1045)	0.070	0.089(899)	0.065	0.116(782)	0.063	0.072(1072)	0.071	0.075(943)	0.070	0.077(948)	0.061	0.063(1063)
landsat	0.115	**0.114(800)**	0.112	0.119(381)	0.111	0.120(403)	0.109	**0.109(575)**	0.117	**0.110(510)**	0.109	**0.108(575)**	0.121	**0.118(456)**
parkinsons	0.112	0.151(32)	0.130	0.170(25)	0.150	0.181(34)	0.106	0.135(33)	0.121	0.147(30)	0.107	0.138(29)	0.142	0.163(37)
pengcolon	0.164	0.282(40)	0.171	0.287(40)	0.180	0.238(32)	0.256	0.311(34)	0.171	0.289(39)	0.244	0.367(38)	0.176	0.269(39)
penglung	0.052	0.080(15)	0.050	0.081(16)	0.048	0.091(15)	0.083	0.115(14)	0.050	0.091(15)	0.054	0.078(14)	0.054	0.083(15)
semeion	0.239	0.246(641)	0.226	**0.204(768)**	0.159	0.185(426)	0.272	**0.258(247)**	0.141	0.161(475)	0.146	0.156(419)	0.247	0.253(658)
sonar	0.181	**0.178(27)**	0.212	0.218(27)	0.236	0.246(30)	0.203	0.222(25)	0.190	**0.181(27)**	0.184	**0.177(26)**	0.178	**0.178(27)**
spect	0.224	0.233(17)	0.228	**0.226(19)**	0.224	0.233(18)	0.228	0.232(15)	0.221	0.230(18)	0.216	0.236(16)	0.223	0.242(17)
splice	0.106	0.155(1873)	0.106	0.164(1888)	0.143	0.257(1586)	0.130	0.152(1605)	0.105	0.157(1826)	0.119	0.153(1570)	0.106	0.154(1815)
waveform	0.148	0.163(1181)	0.147	0.177(1281)	0.213	0.239(995)	0.163	**0.161(1019)**	0.154	0.162(851)	0.160	0.163(719)	0.150	0.158(1076)
wine	0.092	0.233(58)	0.109	0.210(54)	0.109	0.210(54)	0.092	0.233(58)	0.092	0.232(56)	0.092	0.232(56)	0.076	0.241(77)
test statistic	**3**		**7**		**0**		**5**		**6.5**		**6.5**		**2.5**	
	Decision Tree													
	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP	Benign	DMIP
lungcancer	0.354	0.404(18)	0.346	0.429(19)	0.367	0.471(16)	0.408	0.425(17)	0.350	0.463(19)	0.358	0.450(17)	0.350	0.463(18)
breast	0.059	0.078(182)	0.047	0.078(143)	0.067	0.104(203)	0.066	0.083(148)	0.049	0.067(175)	0.061	0.074(146)	0.061	0.078(149)
congress	0.039	0.078(130)	0.041	0.136(126)	0.042	0.151(118)	0.037	0.101(126)	0.038	0.130(133)	0.039	0.134(134)	0.043	0.086(118)
heart	0.228	0.250(40)	0.221	0.263(38)	0.221	0.263(38)	0.228	0.250(40)	0.227	0.246(40)	0.227	0.246(40)	0.229	0.256(39)
ionosphere	0.079	0.107(81)	0.085	0.136(93)	0.095	0.120(55)	0.071	0.113(92)	0.076	0.093(70)	0.076	0.104(68)	0.087	0.108(96)
krvskp	0.061	0.117(1045)	0.061	0.115(899)	0.064	0.144(782)	0.056	0.137(1072)	0.061	0.138(943)	0.060	0.136(948)	0.061	0.111(1063)
landsat	0.104	0.105(800)	0.104	0.108(381)	0.103	0.111(403)	0.099	0.100(575)	0.107	**0.104(510)**	0.099	0.100(575)	0.108	0.110(456)
parkinsons	0.115	0.133(32)	0.131	0.156(25)	0.139	0.182(34)	0.109	0.158(33)	0.121	0.147(30)	0.115	0.154(29)	0.142	0.151(37)
pengcolon	0.171	0.258(40)	0.171	0.213(40)	0.153	0.220(32)	0.180	0.207(34)	0.164	0.207(39)	0.156	0.204(38)	0.160	0.260(39)
penglung	0.100	0.139(15)	0.100	0.150(16)	0.100	0.146(15)	0.096	0.144(14)	0.102	0.146(15)	0.096	0.130(14)	0.094	0.131(15)
semeion	0.248	0.271(641)	0.249	0.264(768)	0.214	0.253(426)	0.257	0.269(247)	0.205	0.230(475)	0.202	0.232(419)	0.252	0.289(658)
sonar	0.176	0.197(27)	0.203	0.206(27)	0.203	0.221(30)	0.179	0.193(25)	0.187	**0.169(27)**	0.185	**0.185(26)**	0.180	0.192(27)
spect	0.223	0.239(17)	0.209	0.231(19)	0.204	0.223(18)	0.216	0.234(15)	0.216	0.226(18)	0.201	0.238(16)	0.227	0.244(17)
splice	0.051	0.169(1873)	0.051	0.168(1888)	0.074	0.233(1586)	0.066	0.154(1605)	0.051	0.172(1826)	0.057	0.155(1570)	0.051	0.167(1815)
waveform	0.141	0.161(1181)	0.143	0.176(1281)	0.201	0.234(995)	0.151	0.163(1019)	0.144	0.160(851)	0.152	0.159(719)	0.144	0.157(1076)
wine	0.098	0.258(58)	0.105	0.248(54)	0.105	0.248(54)	0.098	0.258(58)	0.098	0.263(56)	0.098	0.263(56)	0.083	0.277(77)
test statistic	**0**		**0**		**0**		**0**		**6.5**		**0**		**0**	

4.2. Discussion

In this section, we discuss the experiment results from two perspectives. First, we discuss the detectability of generated malicious samples. Second, we explain the large number of malicious samples required for some datasets.

The detectability refers to the malicious sample's similarity to the benign sample. Easily detected malicious samples are deemed as outliers and filtered by data preprocessing. In DMIP poisoning algorithm, for each malicious data sample, the marginal and joint outcomes of the poisoned two features and the class label are from the training dataset's outcome spaces. The remaining feature values are determined by randomly sampling data points from the training dataset. Therefore, the malicious data samples can blend into the benign data samples and are indistinguishable.

Secondly, we evaluated the number of poisoning samples required by the DMIP poisoning algorithm. We observe in Fig. 2 that the DMIP poisoning algorithm requires a significant amount of poisoning data in some benchmarks (e.g., "lungcancer", "penglung"). The first reason of our observation is that the DMIP poisoning algorithm targets changing the underlying data distribution rather than "pushing" data points across the decision boundary. The second reason is due to the constraints in the DMIP algorithm (see Section 3-C). We use an experiment to explain our point. We first relax the poisoning constraints, i.e., (F, F_p, Y)'s outcome in $\widehat{\mathcal{D}}$ is from $\Delta\prime = \{(f, f_p, y) : (f, f_p, y) \in \mathcal{F} \otimes \mathcal{Y} \otimes \mathcal{F}_p\}$. Then for four benchmark datasets, we greedily search the data samples one by one in $\Delta\prime$ that best optimize the DMIP objective. We compare the required poisoning number with the DMIP poisoning in Fig. 3. As we can see, the high poisoning ratio has to do with the required constraints.

In the DMIP poisoning algorithm, we did not use the greedy search because it is theoretically ill-founded and lacks the consideration for low detectability (i.e., constraints in Section 3-C). Moreover, although we only poisoned two features and the class label in this contribution. The DMIP poisoning algorithm can be extended to N random variables scenario ($N > 3$) for novel FS poisoning methods. However, the greedy searching space in such case will increase exponentially and becomes computationally intractable.

4.3. Research merits

In this section, we discuss the research merits of the proposed method. We focus on the academic merits and practical potentials in the real-world. Although FS plays a vital role in almost all data analysis pipelines, their behaviors in adversarial settings have been largely overlooked, especially for the information-theoretic feature selection methods. This paper is the first contribution that proposed a generic poisoning algorithm against the information-theoretic feature selection methods to the best of our knowledge. Another academic merit in this paper is that we analytically proposed two techniques that allow for manipulating MI among multiple R.V.'s. Additionally, this contribution exposed the risk of greedy forward optimization and provided insights for developing robust and secure information-theoretic feature selection techniques.

Apart from the academic merits, the proposed method also has practical potentials for real-world applications. First, this paper provided a practical method to generate malicious data that cause machine learning models to fail. For example, the spam filtering applications usually analyze the textual emails through the bag-of-words technique. Given that the bag-of-words technique usually generates a dimensionality that amounts to hundreds of thousands, it is often necessary to apply feature selection to reduce feature dimensionality for better generalizability. Thus, the adversary can use the DMIP algorithm to alter the word choices in spam emails such that (i) the feature selection procedure misses the informative features, (ii) the spam emails evade the classification procedure. Secondly, machine learning pipelines can use the DMIP algorithm to evaluate the robustness and design of countermeasures proactively. In the spam filtering example, motivated by [44], the robustness of a spam filtering model can be measured by the minimal injected sample number required by the DMIP algorithm.

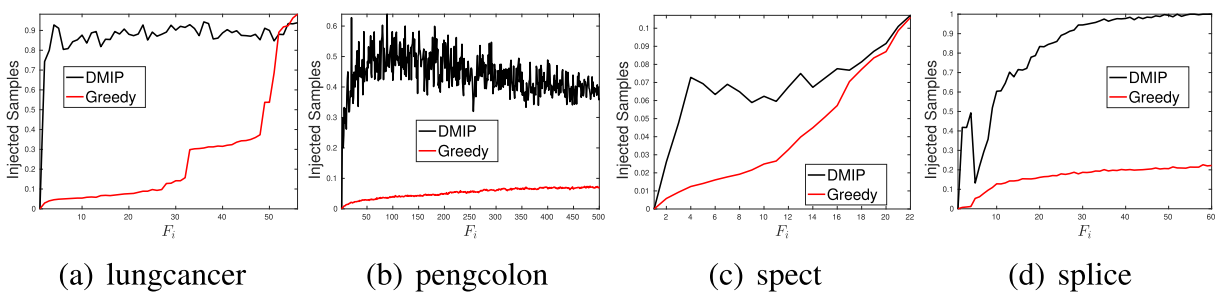

Fig. 3. We plot the required injection number comparison for DMIP and greedy injection.

5. Conclusions

Adversarial machine learning is an emerging field and an increasingly important research topic given the wide spread use of machine learning in application-driven environments. Novel attack algorithms have been proposed for many real-world applications (e.g., audio analysis system, image analysis, etc.). However, as a critical preprocessing stage in machine learning pipelines, FS's behavior in an adversarial environment has largely been under-explored. In this contribution, we revisited the information-theoretic feature selection methods, and adversarial machine learning can impact these preprocessing methods. We also presented a generic poisoning algorithm against information-theoretic feature selection techniques that use a greedy selection procedure. Our key contributions are as follows: First, we used R.V.'s probability distributions as proxies and analytically designed the MIP and DMIP algorithms to manipulate the individual MI and MI among multiple R.V.'s, respectively. Second, we exploited the greedy forward selection in information-theoretic FS methods and proposed a generic poisoning algorithm against information-theoretic FS based on DMIP algorithm. Third, we experimented with the proposed poisoning algorithm against seven state-of-the-art FS objectives on 16 benchmark datasets. The experimental results demonstrated that the proposed poisoning algorithm has a substantial poisoning impact on both the feature selection result and the subsequent classification accuracy. Fourth, the experimental results showed that the poisoning algorithm has transferability by design across different FS objectives. Fifth, we showed that the generated adversarial samples have low detectability.

Apart from the contributions, there is one limitation in this work that can motivate future works. The proposed poisoning algorithm exploited the greedy forward optimization weakness, which is widely-used in information-theoretic FS methods. Thus, when different feature subset optimization procedures are used (e.g., backward elimination), new poisoning algorithms are needed correspondingly. Moreover, another future work is to design a more secure and robust filter FS objective.

CRediT authorship contribution statement

Heng Liu: Conceptualization, Methodology, Software, Writing - original draft, Validation. **Gregory Ditzler:** Conceptualization, Methodology, Writing - original draft, Validation, Conceptualization, Methodology, Software, Writing - original draft, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by grants from the Department of Energy #DE-NA0003946, National Science Foundation CAREER #1943552, and Army Research Lab W56KGU-20-C-0002. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] B. Battista, I. Pillai, S.R. Bulò, D. Ariu, M. Pelillo, F. Roli, Is data clustering in adversarial settings secure, in: ACM Workshop on Artificial Intelligence and Security, 2013.
[2] R. Battiti, Using mutual information for selecting features in supervised neural net learning, IEEE Trans. Neural Networks (1994).
[3] A.N. Bhagoji, D. Cullina, C. Sitawarin, P. Mittal, Enhancing robustness of machine learning systems via data transformations, in: Conference on Information Sciences and Systems (CISS), 2018.
[4] B. Biggio, G. Fumera, F. Roli, Security evaluation of pattern classifiers under attack, IEEE Trans. Knowl. Data Eng. 26 (4) (2013) 984–996.
[5] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, Pattern Recognition. (2018).
[6] G. Brown, A. Pocock, M.-J. Zhao, M. Luján, Conditional likelihood maximisation: A unifying framework for information theoretic feature selection, J. Mach. Learn. Res. 13 (2012) 27–66.
[7] N. Carlini, D. Wagner, Audio adversarial examples: Targeted attacks on speech-to-text, in: IEEE Security and Privacy Workshops (SPW), 2018.
[8] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley-Interscience, 2006.
[9] N. Dalvi, Sanghai S. Mausam, D. Verma, Adversarial classification, in: International Conference on Knowledge Discovery and Data Mining, 2004, pp. 99–108.
[10] J. Demšar, Statistical comparisions of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[11] G. Ditzler, R. Polikar, G. Rosen, A bootstrap based neyman–pearson test for identifying variable importance, IEEE Trans. Neural Netw. Learn. Syst 26 (4) (2015) 880–886.
[12] G. Ditzler, A. Prater, Fine tuning lasso in an adversarial environment against gradient attacks, in: IEEE Symposium Series on Computational Intelligence, 2017.
[13] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd edition., John Wiley & Sons Inc, 2001.
[14] F. Fleuret, Fast binary feature selection with conditional mutual information, J. Mach. Learn. Res. (2004).
[15] C. Frederickson, M. Moore, G. Dawson, R. Polikar, Attack strength vs. detectability dilemma in adversarial machine learning, in: International Joint Conference on Neural Networks, 2018.
[16] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, in: International Conference on Machine Learning, 2013.

[17] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: International Conference on Learning RepresentationsRepresentations, 2014.
[18] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res 3 (2003) 1157–1182.
[19] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (2002) 389–422.
[20] L. Huang, A. Joseph, B. Nelson, B. Rubinstein, J.D. Tygar, Adversarial maching learning, in: ACM Worskshop on Artificial Intelligence and Security, 2011, pp. 43–58.
[21] Jakulin, A. (2005). Machine Learning Based on Attribute Interactions. PhD thesis, University of Ljubljana, Slovenia. .
[22] N. Karmarkar, A new polynomial-time algorithm for linear programming, in: Proceedings of the sixteenth annual ACM symposium on Theory of computing, 1984, pp. 302–311.
[23] M. Kearns, M. Li, Learning in the presence of malicious errors, SIAM J. Comput. 22 (1993) 807–837.
[24] T. Khoshgoftaar, A. Fazelpour, H. Wang, R. Wald, A survey of stability analysis of feature subset selection techniques, in: International Conference on Information Reuse and Integration, 2013, pp. 424–431.
[25] L.I. Kuncheva, A stability index for feature selection, in: International Conference on Artifical Intelligence and Application, 2007, pp. 390–395.
[26] D.D. Lewis, Feature selection and feature extraction for text categorization, in: Proceedings of the Workshop on Speech and Natural Language, 1992, pp. 212–217.
[27] B. Li, Y. Vorobeychik, Feature cross-substitution in adversarial classification, in: Advances in Neural Information Processing Systems, 2014.
[28] Li, F., Lai, L., and Cui, S. (2020). On the adversarial robustness of feature selection using lasso. In IEEE International Workshop on Machine Learning for Signal Processing..
[29] D. Lin, X. Tang, Conditional infomax learning: An integrated framework for feature extraction and fusion, in: European Conference on Computer Vision, 2006.
[30] H. Liu, G. Ditzler, Data poisoning attacks against mrmr, in: International Conference on Acoustics Speech and Signal Processing, 2019.
[31] D. Lowd, C. Meek, Adversarial learning, in: Knowledge and Data Discovery, 2005.
[32] P. Meyer, G. Bontempi, On the use of variable complementarity for feature selection in cancer classification, in: Evolutionary Computation and Machine Learning in Bioinformatics, 2006.
[33] S. Nogueira, G. Brown, Measuring the stability of feature selection with applications to ensemble methods, in: International Workshop on Multiple Classifier Systems, 2015.
[34] Papernot, N., McDaniel, P., and Goodfellow, I. (2016a). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv:1605.07277..
[35] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., and Swami, A. (2016b). Practical black-box attacks against deep learning systems using adversarial examples. arXiv:1602.02697..
[36] H. Peng, F. Long, C. Ding, Feature selection based on mutual information: criteria of max–dependency, max–relevance, and min–redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238.
[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: International Conference on Learning Representations, 2013.
[38] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Society 58 (1) (1996) 267–288.
[39] F. Wang, W. Liu, S. Chawla, On sparse feature attacks in adversarial learning, in: IEEE International Conference on Data Mining, 2014, pp. 1013–1018.
[40] M. Wu, Y. Li, Adversarial mrmr againts evasion attacks, in: International Joint Conference on Neural Networks, 2018.
[41] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, F. Roli, Is feature selection secure against training data poisoning?, in: International Conference on Machine Learning, 2015
[42] Yang, H.H. and Moody, J. (1999). Feature selection based on joint mutual information. In Advances in Intelligent Data Analysis..
[43] Z. Yang, B. Li, P.-Y. Chen, D. Song, Characterizing audio adversarial examples using temporal dependency, in: International Conference on Learning Representations, 2019.
[44] F. Zhang, P.P.K. Chan, B. Biggio, D. Yeung, F. Roli, Adversarial feature selection against evasion attacks, IEEE Trans. Cybern. 46 (3) (2016) 766–777.
[45] Zhou, Y., Kantarcioglu, M., and Xi, B. (2018). Breaking transferability of adversarial samples with randomness. In arXiv preprint arXiv:1805.04613. .