Learning and Planning in the Feature Deception Problem

Zheyuan Ryan Shi[1], Ariel D. Procaccia[2], Kevin S. Chan[3], Sridhar Venkatesan[4], Noam Ben-Asher[3], Nandi O. Leslie[3], Charles Kamhoua[3], and Fei Fang[1]

[1] Carnegie Mellon University
[2] Harvard University
[3] Army Research Laboratory
[4] Perspecta Labs Inc.

Abstract. Today's high-stakes adversarial interactions feature attackers who constantly breach the ever-improving security measures. Deception mitigates the defender's loss by misleading the attacker to make suboptimal decisions. In order to formally reason about deception, we introduce the *feature deception problem (FDP)*, a domain-independent model and present a learning and planning framework for finding the optimal deception strategy, taking into account the adversary's preferences which are initially unknown to the defender. We make the following contributions. (1) We show that we can uniformly learn the adversary's preferences using data from a modest number of deception strategies. (2) We propose an approximation algorithm for finding the optimal deception strategy given the learned preferences and show that the problem is NP-hard. (3) We perform extensive experiments to validate our methods and results. In addition, we provide a case study of the credit bureau network to illustrate how FDP implements deception on a real-world problem.

1 Introduction

The world today poses more challenges to security than ever before. Consider the cyberspace or the financial world where a defender is protecting a collection of targets, e.g. servers or accounts. Despite the ever-improving security measures, malicious attackers work diligently and creatively to outstrip the defense [23]. Against an attacker with previously unseen exploits and abundant resources, the attempt to protect any target is almost surely a lost cause [10]. However, the defender could induce the attacker to attack a less harmful, or even fake, target. This can be seen as a case of deception.

Deception has been an important tactic in military operations for millenia [14]. More recently, it has been extensively studied in cybersecurity [13,9]. At the start of an attack campaign, attackers typically perform reconnaissance to learn the configuration of the machines in the network using tools such as Nmap [16]. Security researchers have proposed many deceptive measures to manipulate a machine's response to these probes [12,2], which could confound and mislead an attempt to attack. In addition, honey-X, such as honeypots, honey users, and honey files have been developed to attract the attackers to attack these fake targets [30]. For example, it is reported that country A once created encrypted but fake files with names of country B's military systems and

Feature	Observed value	Actual value
Operating system	Windows 2016	RHEL 7
Service version	v1.2	v1.4
IP address	10.0.1.2	10.0.2.1
Open ports	22, 445	22, 1433
Round trip time for probes [28]	16 ms	84 ms

Table 1. Example features in cybersecurity

marked them to be shared with country A's intelligence agency [18]. Using sensitive filenames as bait, country A successfully lured country B's hackers to these decoy targets.

Be it commanding an army or protecting a computer network, a common characteristic is that the attacker gathers information about the defender's system to make decisions, and the defender can (partly) control how her system appears to the surveillance. We formalize this view, abstract the collected information about the defender's system that is relevant to attacker's decision-making as features, and propose the *feature deception problem (FDP)* to model the strategic interaction between the defender and the attacker.

It is evident that the FDP model could be applied to many domains by appropriately defining the relevant set of features. To be concrete, we will ground our discussion in cybersecurity, where an attacker observes the features of each network node when attempting to fingerprint the machines (example features shown in the left column of Table 1) and then chooses a node to compromise. Attackers may have different preferences over feature value combinations when choosing targets to attack. If an intruder has an exploit for Windows machines, a Linux server might not be attractive. If the attacker is interested in exfiltration, he might choose a machine running database services. If the defender knows the attacker's preferences, she could strategically configure important machines appear undesirable or configure the honeypots to appear attractive to the attacker, by changing the observed value of the features, e.g. Table 1. However, to make an informed decision, she needs to first learn the attacker's preferences.

Our Contributions Based on our proposed FDP model, we provide a learning and planning framework and make three key contributions. First, we analyze the sample complexity of learning attacker's preferences. We prove that to learn a classical subclass of preferences that is typically used in the inverse reinforcement learning and behavioral game theory literature, the defender needs to gather only a polynomial number of data points on a linear number of feature configurations. The proof leverages what we call the *inverse feature difference* matrix (IFD), and shows that the complexity depends on the norm of this matrix. If the attacker is aware of the learning, they may try to interfere with the learning process by launching the data-poisoning attack, a typical threat model in adversarial machine learning. Using the IFD, we demonstrate the robustness of learning in FDP against this kind of attack. Second, we study the planning problem of finding the optimal deception strategy against learned attacker's preferences. We show that it is NP-hard and propose an approximation algorithm. In addition, we perform extensive experiments to validate our results. We also conduct a case study to illustrate how our FDP framework implements deception on the network of a credit bureau.

2 The Feature Deception Problem

In an FDP, a defender aims to protect a set N of n targets from an adversary. Each target $i \in N$ has a set M of m features. The adversary observes these features and then chooses a target to attack. The defender incurs a loss $u_i \in [-1, 1]$ if the adversary chooses to attack target i.[5] The defender's objective is to minimize her expected loss. Now, we introduce several key elements in FDP. We provide further discussions on some of the assumptions in FDP in the final section.

Features Features are the key element of the FDP model. Each feature has an *observed* value and an *actual* value. The actual value is given and fixed, while the defender can manipulate the observed value. Only the observed values are visible to the adversary. This ties into the notion of deception, where one may think of the actual value as representing the "ground truth" whereas the observed value is what the defender would like the attacker to see. Since deception means manipulating the attacker's perceived value of a target, not the actual value, changing the observable values does not affect the defender's loss u_i at each target.

Table 1 shows an example in cybersecurity. In practice, there are many ways to implement deception. For example, a node running Windows (actual feature) manages to reply to reconnaissance queries in Linux style using tools like OSfuscate. Then the attacker might think the node is running Linux (observed feature). For IP deception, Jafarian et al. [11] and Chiang et al. [4] demonstrate methods to present to the attacker a different IP from the actual one. In addition, when we "fake open" a port with no real vulnerable service runs on it, an attack on the underlying service will fail. This could be done with command line tools or existing technologies like Honeyd [24].

Feature representation We represent the observed feature values of target i by a vector $x_i = (x_{ik})_{k \in M} \in [0, 1]^m$. We denote their corresponding actual values as $\hat{x}_i \in [0, 1]^m$. We allow for both continuous and discrete features. In practice, we may have categorical features, such as the type of operating system, and they can be represented using one-hot encoding with binary features.

Feasibility constraints For a feature k with actual value $\hat{x}_{ik}$, the defender can set its observed value $x_{ik} \in C(\hat{x}_{ik}) \subseteq [0, 1]$, where the feasible set $C(\hat{x}_{ik})$ is determined by the actual value. For continuous features, we assume $C(\hat{x}_{ik})$ takes the form $[\hat{x}_{ik} - \tau_{ik}, \hat{x}_{ik} + \tau_{ik}] \cap [0, 1]$ where $\tau_{ik} \in [0, 1]$. This captures the feasibility constraint in setting up the observed value of a feature based on its actual value. Take the round trip time (RTT) as an example. Shamsi et al. fingerprint the OS using RTT of the SYN-ACK packets [28]. Typical RTTs are in the order of few seconds (Fig. 4 [28]), while a typical TCP session is 3-5 minutes. Thus, perturbing RTT within a few seconds is reasonable, but greater perturbation is dubious.

For binary features, $C(\hat{x}_{ik}) \subseteq \{0, 1\}$. In addition to these feasibility constraints for individual features, we also allow for linear constraints over multiple features, which

[5] Typically, the loss u_i is non-negative, but it might be negative if, for example, the target is set up as a decoy or honeypot, and allows the defender to gain information about the attacker.

could encode natural constraints for categorical features with one-hot encoding, e.g. $\sum_{k\in M'} x_{ik} = 1$, with $M' \subseteq M$ being the subset of features that collectively represent one categorical feature. They may also encode the realistic considerations when setting up the observed features. For example, $x_{ik_1} + x_{ik_2} \leq 1$ could mean that a Linux machine $(x_{ik_1} = 1)$ cannot possibly have ActiveX available $(x_{ik_2} = 1)$.

Budget constraint Deception comes at a cost. We assume the cost is additive across targets and features: $c = \sum_{i\in N} \sum_{k\in M} c_{ik}$, where $c_{ik} = \eta_{ik}|x_{ik} - \hat{x}_{ik}|$. For a continuous feature k, η_{ik} represents the cost associated with unit of change from the actual value to the observable value. In the example of RTT deception, defender's cost is the packet delay which can be considered linear. If k is binary, η_{ik} defines the cost of switching states. The defender has a budget B to cover these costs. We note that, though we introduce these explicit forms of feasibility constraints and cost structure, our algorithms in the sequel are not specific to these forms.

Defender strategies The defender's strategy is an observed feature configuration $x = \{x_i\}_{i\in N}$. The defender uses only pure strategies.

Attacker strategies The attacker's pure strategy is to choose a target $i \in N$ to attack. Since human behavior is not perfectly rational and the attacker may have preferences that are unknown to the defender a priori, we reason about the adversary using a general class of bounded rationality models. We assume the attacker's utilities are characterized by a score function $f : [0, 1]^m \to \mathbb{R}_{>0}$ over the observed feature values of a target. Given observed feature configuration $x = \{x_i\}_{i\in N}$, he attacks target i with probability $\frac{f(x_i)}{\sum_{j\in N} f(x_j)}$. f may take any form and in this paper, we assume that it can be parameterized by or approximated with a neural network with parameter w. In some of the theoretical analyses, we focus on a subclass of functions

$$f_w(x_i) = \exp\left(\sum\nolimits_{k\in M} w_k x_{ik}\right). \tag{1}$$

We omit the subscript w when there is no confusion. This functional form is commonly used to approximate the agent's reward or utility function in inverse reinforcement learning and behavioral game theory, and has been empirically shown to capture many attacker preferences in cybersecurity [1]. For example, the tactics of advanced persistent threat group APT10 [25] are driven by: (1) final goal: they aim at exfiltrating data from workstation machines; (2) expertise: they employ exploits against Windows workstations; (3) services available: their exploits operate against file sharing and remote desktop services. Thus, APT10 prefer to attack machines with Windows OS running a file-sharing service on the default port. Each of these properties is a "feature" in FDP and a score function f in Eq (1) can assign a greater weight for each of these features. It can also capture more complex preferences by using hand-crafted features based on domain knowledge. For example, APT10 typically scan for NetBIOS services (i.e., ports 137 and 138), and Remote Desktop Protocol services (i.e., ports 445 and 3389) to identify systems that they might get onto [25]. Instead of treating the availability of ports as features, we may design a binary feature indicating whether each of the service is available (representing an "OR" relationship of the port availability features). We

also show a more efficient way to approximately handle combinatorial preferences in Section 5.4. In addition, this score function also captures fully rational attackers in the limit.

The ultimate goal of the defender is to find the optimal feature configuration against an unknown attacker. This can be decomposed into two subtasks: *learning* the attacker's behavior model from attack data and *planning* how to manipulate the feature configuration to minimize her expected loss based on the learned preferences. In the following sections, we first analyze the sample complexity of the learning task and then propose algorithms for the planning task.

3 Learning the Adversary's Preferences

The defender learns the adversary's score function f from a set of d labeled data points each in the format of (N, x, y) where N is the set of targets and x is the observed feature configuration of all targets in N. The label $y \in N$ indicates that the adversary attacks target y.

In practice, there are two ways to carry out the learning stage. First, the defender can learn from historical data. Second, the defender can also actively collect data points while manipulating the observed features of the network. This is often done with honeynets [30], i.e. a network of honeypots.

No matter which learning mode we use, it is often the case, e.g. in cybersecurity, that the dataset contains multiple data points with the same x, since changing the defender configuration frequently leads to too much overhead. In addition, at the learning stage, only the observed feature values x matter because the attacker does not observe the actual feature values $\hat{x}$. The feasibility constraints $C(\hat{x}_{ik})$ on each feature still apply. Yet, they are irrelevant during learning because we use either historical data that satisfy these constraints, or honeypots for which these constraints are vacuous.

To analyze the sample complexity of learning the adversary's preferences, we focus on the classical form score function f in Eq (1). We show that, in an FDP with m features, the defender can learn the attacker's behavior model correctly with high probability, using only m observed feature configurations and a polynomial number of samples. We view this condition as very mild, because even if the network admin's historical dataset does not meet the requirement, she could set up a honeynet to elicit attacks, where she can control the feature configurations of each target [30]. It is still not free for the defender to change configurations, but attacks on honeynet do not lead to actual loss since it runs in parallel with the production network.

To capture the multiple features in FDP, we introduce the *inverse feature difference matrix* $(A^{st})^{-1}$. Specifically, given observed feature configurations $x^1, \ldots, x^m$, for any two targets $s, t \in N$, let A^{st} be the $m \times m$ matrix whose (i, j)-entry is $a^{st}_{ij} = x^i_{sj} - x^i_{tj}$. A^{st} captures the matrix-level correlation among feature configurations. We use the matrix norm of $(A^{st})^{-1}$ to bound the learning error.

For feature configuration x, let $D^x(t) = \frac{f(x_t)}{\sum_{i \in N} f(x_i)}$ be the attack probability on target t. We assume $\rho := \min_{x,t} D^x(t) > 0$. Let $\alpha = \min_{s \neq t} ||(A^{st})^{-1}||$, where $||\cdot||$ is the matrix norm induced by the L^1 vector norm, i.e. $||(A^{st})^{-1}|| = \sup_{y \neq 0} \frac{|(A^{st})^{-1} y|}{|y|}$. Our result is stated as the following theorem.

Theorem 1. *Consider m observed feature configurations $x^1, x^2, \ldots, x^m \in [0,1]^{mn}$. With $\Omega(\frac{\alpha^4 m^4}{\rho\epsilon^2} \log \frac{nm}{\delta})$ samples for each of the m feature configurations, with probability $1-\delta$, we can learn a score function $\hat{f}(\cdot)$ with uniform multiplicative error ϵ of the true $f(\cdot)$, i.e., $\frac{1}{1+\epsilon} \leq \frac{f(x_i)}{\hat{f}(x_i)} \leq 1+\epsilon, \forall x_i$.*

Proof. Let $\hat{D}^x(t) = \frac{\hat{f}(x_t)}{\sum_{i \in N} \hat{f}(x_i)}$. We leverage a known result from behavioral game theory [8]. It cannot be directly translated to sample complexity guarantee in FDP because of the correlation among feature configurations, but we use it to reason about attack probabilities in proving Theorem 1.

Lemma 1. *[8] Given observable features $x \in [0,1]^{mn}$, and $\Omega(\frac{1}{\rho\epsilon^2} \log \frac{n}{\delta})$ samples, we have $\frac{1}{1+\epsilon} \leq \frac{\hat{D}^x(t)}{D^x(t)} \leq 1+\epsilon$ with probability $1-\delta$, for all $t \in N$.*

Fix $\epsilon, \delta > 0$. From Eq. (1), for each x^i where $i = 1, 2, \ldots, m$, we have

$$\sum_{j=1}^{m} w_j (x^i_{sj} - x^i_{tj}) = \ln \frac{D^{x^i}(s)}{D^{x^i}(t)}, \quad \forall s, t \in N, s \neq t$$

Let

$$b^{st} = (\ln \frac{D^{x^1}(s)}{D^{x^1}(t)}, \ldots, \ln \frac{D^{x^m}(s)}{D^{x^m}(t)})^T.$$

The system of equations above can be represented by $A^{st} w = b^{st}$. It is known that $||A^{st}|| = \max_{1 \leq j \leq m} \sum_{i=1}^{m} |a^{st}_{ij}|$. In our case, the feature values are bounded in $[0,1]$ and thus $|a^{st}_{ij}| \leq 1$. This yields $||A^{st}|| \leq m$. Now, choose s, t such that $||(A^{st})^{-1}|| = \alpha$. Suppose A^{st} is invertible.

Let $\epsilon' = \frac{\epsilon}{4\alpha^2 m^2}$ and $\delta' = \frac{\delta}{m}$. Suppose we have $\Omega(\frac{1}{\rho\epsilon'^2} \log \frac{n}{\delta'})$ samples. From Lemma 1, for any node $r \in N$ and any feature configuration x^i where $i = 1, 2, \ldots, m$, $\frac{1}{1+\epsilon'} \leq \frac{\hat{D}^{x^i}(r)}{D^{x^i}(r)} \leq 1+\epsilon'$ with probability $1-\delta'$. The bound holds for all strategies simultaneously with probability at least $1 - m\delta' = 1 - \delta$, using a union bound argument. In particular, for our chosen nodes s and t, we have

$$\frac{1}{(1+\epsilon')^2} \leq \frac{\hat{D}^{x^i}(s)}{\hat{D}^{x^i}(t)} \frac{D^{x^i}(t)}{D^{x^i}(s)} \leq (1+\epsilon')^2, \quad \forall i = 1, \ldots, m$$

Define $\hat{b}^{st}$ similarly as b^{st} but using empirical distribution $\hat{D}$ instead of true distribution D. Let $e = \hat{b}^{st} - b^{st}$. Then, for each $i = 1, \ldots, m$, we have

$$-2\epsilon' \leq 2 \ln \frac{1}{1+\epsilon'} \leq e_i = \ln \frac{\hat{D}^{x^i}(s) D^{x^i}(t)}{\hat{D}^{x^i}(t) D^{x^i}(s)} \leq 2 \ln(1+\epsilon') \leq 2\epsilon'$$

Therefore, we have $|e| \leq 2\epsilon' m$. Let $\hat{w}$ be such that $A^{st}\hat{w} = \hat{b}^{st}$, i.e. $\hat{w} - w = (A^{st})^{-1}e$. Observe that

$$\frac{|(A^{st})^{-1}e|/|(A^{st})^{-1}b^{st}|}{|e|/|b^{st}|} \leq \max_{\tilde{e},\tilde{b}^{st}\neq 0} \frac{|(A^{st})^{-1}\tilde{e}|/|(A^{st})^{-1}\tilde{b}^{st}|}{|\tilde{e}|/|\tilde{b}^{st}|}$$
$$= \max_{\tilde{e}\neq 0} \frac{|(A^{st})^{-1}\tilde{e}|}{|\tilde{e}|} \max_{\tilde{b}^{st}\neq 0} \frac{|\tilde{b}^{st}|}{|(A^{st})^{-1}\tilde{b}^{st}|} = \max_{\tilde{e}\neq 0} \frac{|(A^{st})^{-1}\tilde{e}|}{|\tilde{e}|} \max_{y\neq 0} \frac{|A^{st}y|}{|y|} = ||(A^{st})^{-1}|| \cdot ||A^{st}||$$

This leads to

$$\begin{aligned} |(A^{st})^{-1}e| &\leq ||(A^{st})^{-1}|| \cdot ||A^{st}|| \cdot |e| \cdot \frac{|(A^{st})^{-1}b^{st}|}{|b^{st}|} \\ &\leq ||(A^{st})^{-1}|| \cdot ||A^{st}|| \cdot |e| \cdot \max_{\tilde{b}^{st}\neq 0} \frac{|(A^{st})^{-1}\tilde{b}^{st}|}{|\tilde{b}^{st}|} \\ &= ||(A^{st})^{-1}||^2 \cdot ||A^{st}|| \cdot |e| \leq \alpha^2 m(2\epsilon' m) \end{aligned}$$

For any observable feature configuration x,

$$\left| \left(\sum_{j=1}^{m} w_j x_{ij} \right) - \left(\sum_{j=1}^{m} \hat{w}_j x_{ij} \right) \right| \leq \sum_{j=1}^{m} |\hat{w}_j - w_j| = |(A^{st})^{-1}e| \leq \alpha^2 m(2\epsilon' m) = \frac{\epsilon}{2}$$

Therefore,

$$\frac{1}{1+\epsilon} \leq \frac{f(x_i)}{\hat{f}(x_i)} \leq 1 + \epsilon. \quad \square$$

It is easy to see that we do not have to use the same pair of targets (s, t) for every feature configuration. In fact, this result can be easily adapted to allow for each feature configuration being implemented on a different system with a different set and number of targets. Instead of defining A^{st} and b^{st}, we could define A and b, where row i of A and i-th entry of b correspond to feature configuration x^i and targets (s^i, t^i). If feature configuration x^i is implemented on a system with n_i targets, we need $\Omega(\frac{1}{\rho\epsilon'^2} \log \frac{n_i}{\delta'})$ samples from this system, and then the argument above still holds.

The α in Theorem 1 need not be large, especially if the defender can select the feature configurations to collect data and elicit preferences. Consider a sequence of m feature configurations $x^1, \ldots, x^m$, and focus on targets 1 and 2. For each x^j, let the features on target 1 be identical to target 2, except for the j-th feature, where $x^j_{1j} = 1$ and $x^j_{2j} = 0$. This leads to $A^{12} = I$, and $\alpha \leq 1$. This also shows that it is not hard to set up the configurations such that A^{st} is nonsingular.

An adversary who is aware of the defender's learning procedure might sometimes intentionally attack without following his true score function f, to mislead the defender. The following theorem states that the defender can still learn an approximately correct f even if the attacker contaminates a γ fraction of the data.

Theorem 2. *In the setting of Theorem 1, if the attacker modifies a $\gamma \leq \frac{\epsilon\rho}{4\alpha m}$ fraction of the data points for each feature configuration, the function f can be learned within multiplicative error 3ϵ.*

Proof. Fix two nodes s, t. Recall that in Theorem 1, without data poisoning, we learned the weights w by solving the linear equations $A^{st}\tilde{w} = \tilde{b}^{st}$ based on the empirical distribution of attacks, where $\tilde{b}^{st} = (\ln \frac{\tilde{D}^{x^1}(s)}{\tilde{D}^{x^1}(t)}, \ldots, \ln \frac{\tilde{D}^{x^m}(s)}{\tilde{D}^{x^m}(t)})$. [6] Denote a parallel system of equations $A^{st}\hat{w} = \hat{b}^{st}$ which uses the poisoned data. We are interested in bounding $|\hat{w} - \tilde{w}| = |(A^{st})^{-1}(\hat{b}^{st} - \tilde{b}^{st})|$. Consider the k-th entry in the vector $\hat{b}^{st} - \tilde{b}^{st}$:

$$|(\hat{b}^{st} - \tilde{b}^{st})_k| = \left| \ln \frac{\hat{D}^{x^k}(s)}{\hat{D}^{x^k}(t)} \frac{\tilde{D}^{x^k}(t)}{\tilde{D}^{x^k}(s)} \right|$$

To simplify the notations, we denote $\tilde{D}^{x^k}(t) = \gamma_t^k$ and $\tilde{D}^{x^k}(s) = \gamma_s^k$, and without loss of generality, assume $\gamma_t^k \leq \gamma_s^k$. To find an upper bound of RHS of the above equation, we define function $g(\gamma_1, \gamma_2) = \frac{\gamma_t^k(\gamma_s^k + \gamma_1)}{\gamma_s^k(\gamma_t^k - \gamma_2)}$, and define function $h(\gamma_1, \gamma_2) = |\ln g(\gamma_1, \gamma_2)|$. The constraint that the attacker can only change γ fraction of the points translates into $|\gamma_1|, |\gamma_2|, |\gamma_1 - \gamma_2| \leq \gamma$. Since g is increasing in γ_1 and γ_2, g attains maximum at $(\gamma_1, \gamma_2) = (\gamma, \gamma)$ and minimum at $(\gamma_1, \gamma_2) = (-\gamma, -\gamma)$, which are the only two possible maxima of h. Observe that $g(\gamma, \gamma) \geq 1$ and $g(-\gamma, -\gamma) \leq 1$. It then suffices to compare $g(\gamma, \gamma)$ with $1/g(-\gamma, -\gamma)$:

$$\frac{1/g(-\gamma, -\gamma)}{g(\gamma, \gamma)} = \frac{\gamma_s(\gamma_t + \gamma)}{\gamma_t(\gamma_s - \gamma)} \frac{\gamma_s(\gamma_t - \gamma)}{\gamma_t(\gamma_s + \gamma)} = \frac{\gamma_s^2\gamma_t^2 - \gamma_s^2\gamma^2}{\gamma_t^2\gamma_s^2 - \gamma_t^2\gamma^2} \leq 1$$

Therefore, $h(\gamma_1, \gamma_2)$ is maximized at $(\gamma_1, \gamma_2) = (\gamma, \gamma)$. From here, we obtain

$$|(\hat{b}^{st} - \tilde{b}^{st})_k| \leq \ln \frac{(\gamma_s^k + \gamma)\gamma_t^k}{(\gamma_t^k - \gamma)\gamma_s^k} = \ln \left(\left(1 + \frac{\gamma}{\gamma_s^k}\right) \left(1 + \frac{\gamma}{\gamma_t^k - \gamma}\right) \right) \leq \frac{\gamma}{\gamma_s^k} + \frac{\gamma}{\gamma_t^k - \gamma}.$$

Recall that

$$\frac{\left|(A^{st})^{-1}(\hat{b}^{st} - \tilde{b}^{st})\right|}{\left|\hat{b}^{st} - \tilde{b}^{st}\right|} \leq \sup_{y \neq 0} \frac{|(A^{st})^{-1}y|}{|y|} = ||(A^{st})^{-1}|| = \alpha$$

Thus, we get

$$|\hat{w} - \tilde{w}| = |(A^{st})^{-1}(\hat{b}^{st} - \tilde{b}^{st})| \leq \alpha \left|\hat{b}^{st} - \tilde{b}^{st}\right| \leq \alpha \sum_{k=1}^{m} \left(\frac{\gamma}{\gamma_s^k} + \frac{\gamma}{\gamma_t^k - \gamma} \right)$$

Note that by Lemma 1, we have $\gamma_t^k \geq \frac{\rho}{1+\epsilon'} \geq \frac{\rho}{2}$. Since we assumed that $\gamma \leq \frac{\epsilon\rho}{4\alpha m} \leq \frac{\epsilon\rho}{4}$, we know that $\gamma \leq \gamma_t/2$. Thus, we get

$$|\hat{w} - \tilde{w}| \leq \alpha \sum_{k=1}^{m} \left(\frac{\gamma}{\gamma_s^k} + \frac{2\gamma}{\gamma_t^k} \right) \leq \frac{3\epsilon(1+\epsilon')}{4} \leq \frac{3}{4}\epsilon \left(1 + \frac{1}{4}\epsilon\right)$$

[6] Refer to the proof of Theorem 1 for the notations used.

From here, using the triangle inequality, we have

$$|\hat{w} - w| \leq |\hat{w} - \tilde{w}| + |\tilde{w} - w| \leq \frac{3}{4}\epsilon\left(1 + \frac{1}{4}\epsilon\right) + \frac{\epsilon}{2} \leq \frac{3}{2}\epsilon$$

Thus, in the end, we get

$$\frac{1}{1+3\epsilon} \leq \frac{f(x_i)}{\hat{f}(x_i)} \leq 1 + 3\epsilon. \quad \square$$

For a general score function f_w, gradient-based optimizers such as RMSProp can be applied to learn w through maximum-likelihood estimation.

$$w = \arg\max_{w'} \sum\nolimits_{j\in[d]} \left[L^j_{w'}(N^j, x^j, y^j)\right]$$

$$L^j_{w'}(N^j, x^j, y^j) = \log(f_{w'}(x^j_{y^j})) - \log(\sum\nolimits_{i\in N^j} f_{w'}(x^j_i))$$

However, it is not guaranteed to find the optimal solution given the non-convexity of L.

4 Computing the Optimal Feature Configuration

We now embark on our second task: assuming the (learned) adversary's behavior model, compute the optimal observed feature configuration to minimize the defender's expected loss. For any score function, the problem can be formulated as the following mathematical program (MP).

$$\min_{x} \quad \frac{\sum_{i\in N} f(x_i)u_i}{\sum_{i\in N} f(x_i)} \tag{2}$$

$$s.t. \quad \sum_{i\in N}\sum_{k\in M} \eta_{ik}|x_{ik} - \hat{x}_{ik}| \leq B \tag{3}$$

$$\text{Categorical feature constraints} \tag{4}$$

$$x_{ik} \in C(\hat{x}_{ik}) \qquad \forall i \in N, k \in M \tag{5}$$

This MP is typically non-convex and very difficult to solve. We show that the decision version of FDP is NP-complete. Hence, finding the optimal feature configuration is NP-hard. In fact, this holds even when there is only a single binary feature and the score function f takes the form in Eq. (1).

Theorem 3. *FDP is NP-complete.*

Proof. We reduce from the Knapsack problem: given $v \in [0,1]^n$, $\omega \in \mathbb{R}^n_+$, $\Omega, V \in \mathbb{R}_+$, decide whether there exists $y \in \{0,1\}^n$ such that $\sum_{i=1}^n v_i y_i \geq V$ and $\sum_{i=1}^n \omega_i y_i \leq \Omega$.

We construct an instance of FDP. Let the set of targets be $N = \{1, \ldots, n+1\}$, and let there be a single binary feature, i.e. $M = \{1\}$ and $x_{i1} \in \{0,1\}$ for each $i \in N$. Since there is only one feature, we abuse the notation by using $x_i = x_{i1}$. Suppose each target's actual value of the feature is $\hat{x}_i = 0$. Consider a score function f with

$f(0) = 1$ and $f(1) = 2$. For each $i \in N$, let $u_i = (1 - v_i)/\delta$ if $i \neq n + 1$, and $u_{n+1} = (1 + V + \sum_{i=1}^{n} v_i)/\delta$. Choose a large enough $\delta \geq 1$ so that $u_{n+1} \leq 1$. For each $i \in N$, let $\eta_i = \omega_i$ if $i \neq n + 1$, and $\eta_{n+1} = 0$. Finally, let the budget $B = \Omega$.

For a solution y to a Knapsack instance, we construct a solution x to the above FDP where $x_i = y_i$ for $i \neq n+1$, and $x_{n+1} = 0$. We know $\sum_{i \in N} \eta_i |x_i - \hat{x}_i| = \sum_{i \in N} \eta_i x_i \leq B$ if and only if $\sum_{i=1}^{n} \omega_i y_i \leq \Omega$. Since $f(x_i) > 0$ for all x_i, $\frac{\sum_{i \in N} f(x_i) u_i}{\sum_{i \in N} f(x_i)} \leq 1/\delta$ if and only if $\sum_{i \in N} (1 - \delta u_i) f(x_i) \geq 0$. Note that $\sum_{i \in N} (1 - \delta u_i) = \sum_{i=1}^{n} v_i (y_i + 1) - \sum_{i=1}^{n} v_i - V$. Thus, y is a certificate of Knapsack if and only if x is feasible for FDP and the defender's expected loss is at most $1/\delta$. □

Despite the negative results for the general case, we design an approximation algorithm for the classical score function in Eq. (1) based on mixed integer linear programming (MILP) enhanced with binary search. As shown in Sec 5, it can solve medium sized problems (up to 200 targets) efficiently. Given $f(x_i) = \exp(\sum_{k \in M} w_k x_{ik})$, scaling the score by a factor of e^{-W} does not affect the attack probability, where $W = |w|$ is the L^1 norm of $w = (w_1, \ldots, w_m)$. Thus, we treat the score function as $f(x_i) = \exp(\sum_{k \in M} w_k x_{ik} - W)$.

With slight abuse of notation, we denote the score of target i as f_i. Let $z_i = \sum_{k \in M} w_k x_{ik} - W \in [-2W, 0]$. We divide the interval $[-2W, 0]$ into $2W/\epsilon$ subintervals, each of length ϵ. On interval $[-l\epsilon, -(l-1)\epsilon]$ with $l = 0, 1, \ldots, 2W/\epsilon$, we approximate the function e^{z_i} with the line segment of slope γ_l connecting the points $(-l\epsilon, e^{-l\epsilon})$ and $(-(l-1)\epsilon, e^{-(l-1)\epsilon})$. We use this method to approximate f_i in the following mathematical program $\mathcal{MP}1$. We represent $z_i = -\sum_l z_{il}$, where each variable z_{il} indicates the quantity z_i takes up on the interval $[-l\epsilon, -(l-1)\epsilon]$. The constraints in Eq. (9)-(10) ensure that $z_{i(l+1)} > 0$ only if $z_{il} = \epsilon$. While $\mathcal{MP}1$ is not technically a MILP, we can linearize the objective and the constraint involving absolute value following a standard procedure [31]. The full MILP formulation can be found in the full arXiv version of the paper.[7]

$$(\mathcal{MP}1) \quad \min_{f,z,x,y} \quad \frac{\sum_i f_i u_i}{\sum_i f_i} \tag{6}$$

$$s.t. \quad f_i = e^{-2W} + \sum_l \gamma_l (\epsilon - z_{il}), \quad \forall i \in N \tag{7}$$

$$\sum_{k \in M} w_k x_{ik} - W = -\sum_l z_{il}, \quad \forall i \in N \tag{8}$$

$$\epsilon y_{il} \leq z_{il}, z_{i(l+1)} \leq \epsilon y_{il}, \quad \forall l, \forall i \in N \tag{9}$$

$$z_{il} \in [0, \epsilon], y_{il} \in \{0, 1\}, \quad \forall l, \forall i \in N \tag{10}$$

Constraints (3)-(5)

We can now establish the following bound.

Theorem 4. *Given $\epsilon < 1$, the MILP is a $2\epsilon^2$-approximation to the original problem.*

[7] The full version of the paper is available at https://arxiv.org/abs/1905.04833.

Proof. To analyze the approximation bound of this MILP, we first need to analyze the tightness of the linear approximation. Consider two points s_1, s_2 where $s_2 - s_1 = \epsilon$. The line segment is $t(s) = \frac{1}{\epsilon}(e^{s_2} - e^{s_1})s - \frac{1}{\epsilon}(e^{s_2} - e^{s_1})s_1 + e^{s_1}$. Let $\Delta(s)$ be the ratio between the line and e^s on the interval $[s_1, s_2]$. Note that $\Delta(s)$ is maximized at

$$s^* = 1 + s_1 - \frac{\epsilon}{e^\epsilon - 1}, \qquad \text{with} \quad \Delta(s^*) = \frac{\frac{e^\epsilon - 1}{\epsilon}}{\exp\{1 - \frac{\epsilon}{e^\epsilon - 1}\}}.$$

Now, let $v = \frac{e^\epsilon - 1}{\epsilon}$. It is known that $v \in [1, 1 + \epsilon]$ when $\epsilon < 1.7$. Note that $\delta(x^*) = v \exp\{\frac{1}{v} - 1\} \leq 1 + (v-1)^2/2$, which holds for all $v \geq 1$. Let $\hat{f}(\cdot)$ be the piecewise linear approximation. For any target i and observable feature configuration x_i, we have

$$\frac{\hat{f}(x_i)}{f(x_i)} \leq v \leq 1 + \frac{\epsilon^2}{2}.$$

Let x^* be the optimal observable features against the true score function f, and let x' be the optimal observable features to the above MILP. Let $U(\cdot)$ be the defender's expected loss, and $\hat{U}(\cdot)$ be the approximate defender's expected loss. For any observable feature configuration x, we have

$$\begin{aligned}
|\hat{U}(x) - U(x)| &= \left| \frac{\sum_i \hat{f}(x_i)u_i}{\sum_i \hat{f}(x_i)} - \frac{\sum_i f(x_i)u_i}{\sum_i f(x_i)} \right| \\
&= \left| \frac{\sum_i \hat{f}(x_i)u_i}{\sum_i \hat{f}(x_i)} - \frac{\sum_i \hat{f}(x_i)u_i}{\sum_i f(x_i)} + \frac{\sum_i \hat{f}(x_i)u_i}{\sum_i f(x_i)} - \frac{\sum_i f(x_i)u_i}{\sum_i f(x_i)} \right| \\
&\leq \frac{2}{\sum_i f(x_i)} \left| \sum_i f(x_i) - \sum_i \hat{f}(x_i) \right| = 2 \left(\frac{\sum_i \hat{f}(x_i)}{\sum_i f(x_i)} - 1 \right) \leq \epsilon^2
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}
U(x') - U(x^*) &= U(x') - \hat{U}(x') + \hat{U}(x') - U(x^*) \\
&\leq U(x') - \hat{U}(x') + \hat{U}(x^*) - U(x^*) \leq 2\epsilon^2 \quad \square
\end{aligned}$$

While $\mathcal{MP}1$ could be transformed into a MILP, the necessary linearization introduces many additional variables, increasing the size of the problem. To improve scalability, we perform binary search on the objective value δ. Specifically, the objective at each iteration of the binary search becomes

$$\min_{f,z,x,y} \quad \sum_i f_i u_i - \delta \sum_i f_i. \tag{11}$$

At each iteration, if the objective value of Eq. (11) is negative, we update the binary search upper bound, and update the lower bound if positive. We proceed to the next iteration until the gap between the bounds is smaller than tolerance ϵ_{bs} and then we output the solution x^{bs} when the upper bound was last updated. The complete procedure is

Algorithm 1: MILP-BS

Initialize $L = -1, U = 1, \delta = 0, \epsilon_{bs}$
while $U - L > \epsilon_{bs}$ **do**
  Solve the MILP $\mathcal{MP}1$ with objective in Eq. (11).
  if *objective value* < 0 **then**
    Let $U = \delta$
  else
    Let $L = \delta$
return U, the MILP solution when U was last updated.

given as Alg. 1. Since Eq. (11) is linear itself, we no longer need to perform linearization on it to obtain a MILP. This leads to significant speedup as we show later. We also preserve the approximation bound using triangle inequalities.

Theorem 5. *Given $\epsilon < 1$ and tolerance ϵ_{bs}, binary search gives a $(2\epsilon^2+\epsilon_{bs})$-approximation.*

Proof. Suppose binary search terminates with interval of length $U - L \leq \epsilon_{bs}$, and observable features x^{bs}. Both x^{bs} and the optimal observable features x' to the MILP lie in this interval. This means $U(x^{bs}, \tilde{f}) - U(x', \tilde{f}) \leq \epsilon_{bs}$. Recall that x^* is the optimal observable features against the true score function f. Therefore, we have

$$\begin{aligned}
&U(x^{bs}, f) - U(x^*, f) = U(x^{bs}, f) - U(x^{bs}, \tilde{f}) + U(x^{bs}, \tilde{f}) - U(x^*, f) \\
&\leq U(x^{bs}, f) - U(x^{bs}, \tilde{f}) + U(x', \tilde{f}) + \epsilon_{bs} - U(x^*, f) \\
&\leq U(x^{bs}, f) - U(x^{bs}, \tilde{f}) + U(x^*, \tilde{f}) + \epsilon_{bs} - U(x^*, f) \\
&\leq 2\epsilon^2 + \epsilon_{bs} \quad \square
\end{aligned}$$

Now, we connect the learning and planning results together. Suppose we learned an approximate score function $\hat{f}$ (Theorem 1), and we find an approximately optimal feature configuration (Theorem 4) assuming $\hat{f}$. The following result shows that we can still guarantee end-to-end approximate optimality.

Theorem 6. *Suppose for some $\epsilon \leq 1/4$, $\frac{1}{1+\epsilon} < \frac{\hat{f}(x_i)}{f(x_i)} < 1 + \epsilon$ for all x_i. Then, $|U(x, \hat{f}) - U(x, f)| \leq 4\epsilon$ for all x. Let $x^* = \arg\min_x U(x, f)$ and x'' be such that $U(x'', \hat{f}) \leq \min_x U(x, \hat{f}) + \eta$, then $U(x'', f) - U(x^*, f) \leq 8\epsilon + \eta$.*

Proof. Let $\hat{f}(x_i) = \exp(\sum_k \hat{w}_k x_{ik})$ and $f(x_i) = \exp(\sum_k w_k x_{ik})$. Since

$$\frac{1}{1+\epsilon} < \frac{\hat{f}(x_i)}{f(x_i)} < 1 + \epsilon,$$

we get

$$-\epsilon \leq -\ln(1+\epsilon) < \sum_k (\hat{w}_k - w_k) x_{ik} = \ln \frac{\hat{f}(x_i)}{f(x_i)} < \ln(1+\epsilon) \leq \epsilon.$$

That is, $|\sum_k(\hat{w}_k - w_k)x_{ik}| < \epsilon$. The proof of Theorem 3.7 in [8] now follows to prove the first part of Theorem 6 if we redefine their $u_i(p_i)$ as $\sum_{k\in M} w_k x_{ik}$ and $\hat{u}_i(p_i)$ as $\sum_{k\in M} \hat{w}_k x_{ik}$. For completeness, we adapt their proof below using our notations.

As defined in Section 3, $D^x(t) = \frac{f(x_t)}{\sum_i f(x_i)}$ and $\hat{D}^x(t) = \frac{\hat{f}(x_t)}{\sum_i \hat{f}(x_i)}$. We have

$$\left|\ln\frac{\hat{D}^x(t)}{D^x(t)}\right| = \left|\left(\sum_k(\hat{w}_k - w_k)x_{tk}\right) - \ln\frac{\sum_i \exp\{\sum_k \hat{w}_k x_{ik}\}}{\sum_i \exp\{\sum_k w_k x_{ik}\}}\right|$$

$$\leq \left|\sum_k(\hat{w}_k - w_k)x_{tk}\right| + \left|\ln\frac{\sum_i \exp\{\sum_k w_k x_{ik}\}\exp\{\sum_k(\hat{w}_k - w_k)x_{ik}\}}{\sum_i \exp\{\sum_k w_k x_{ik}\}}\right|$$

$$< \epsilon + \max_i \left|\ln\exp\{\sum_k(\hat{w}_k - w_k)x_{ik}\}\right| < 2\epsilon$$

Using a few inequalities we can bound $\left|\frac{\hat{D}^x(t)}{D^x(t)} - 1\right| \leq 4\epsilon$. This leads to, for all x,

$$|U(x,\hat{f}) - U(x,f)| = \left|\sum_{i\in N}(\hat{D}^x(i) - D^x(i))u_i\right| \leq \sum_{i\in N}\left|\hat{D}^x(i) - D^x(i)\right||u_i|$$

$$= \sum_{i\in N}\left|\frac{\hat{D}^x(i)}{D^x(i)} - 1\right||u_i|\,D^x(i) \leq 4\epsilon\sum_{i\in N}|u_i|\,D^x(i) \leq 4\epsilon\max_{i\in N}|u_i| \leq 4\epsilon$$

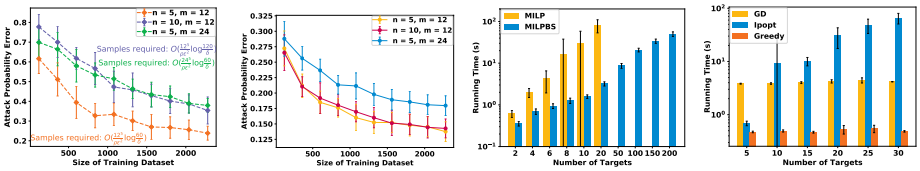
Let $x^* = \arg\min_x U(x,f)$ be the true optimal feature configuration, $x' = \arg\min_x U(x,\hat{f})$ be the optimal configuration using the learned score function $\hat{f}$, and x'' be an approximate optimal configuration against $\hat{f}$, i.e., $U(x'',\hat{f}) \leq U(x',\hat{f}) + \eta$. We have

$$U(x'',f) \leq U(x'',\hat{f}) + 4\epsilon \leq U(x',\hat{f}) + 4\epsilon + \eta \leq U(x^*,\hat{f}) + 4\epsilon + \eta \leq U(x^*,f) + 8\epsilon + \eta. \quad \square$$

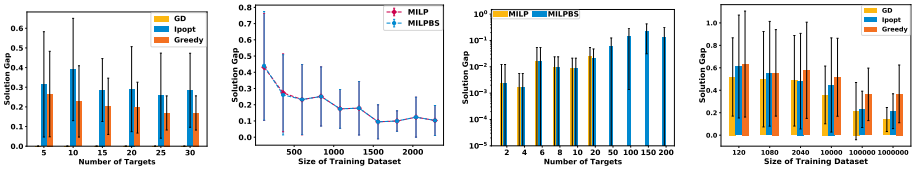
In addition, we propose two exact algorithms for special cases of FDP, which can be found in the arXiv version. When the deception cost is associated with discrete features only, we provide an exact MILP formulation. When there is no budget and feasibility constraints, we can find the optimal defender strategy in $O(n\log n + m)$ time using a greedy algorithm. Inspired by this greedy algorithm, we introduce a greedy heuristic for the general case. GREEDY (Alg.2 in the arXiv version) finds the feature vectors that maximize and minimize the score, respectively, using gradient descent-based algorithm. It then greedily applies these features to targets of extreme losses. We show its performance in the following section as well.

5 Experiments

We present the experimental results for our learning and planning algorithms separately, and then combine them to demonstrate the effectiveness of our learning and planning framework. All experiments are carried out on a 3.8GHz Intel Core i5 CPU with 32GB RAM. We use Ipopt as our non-convex solver and CPLEX 12.8 as the MILP solver. All results are averaged over 20 instances; error bars represent standard deviations. Details about hyper-parameters can be found in the arXiv version of the paper.



(a) Learning classical score function in Eq. (1)
(b) Learning NN-3 score function
(c) Planning with classical score function in Eq. (1), $m = 12$
(d) Planning with NN-3 score function, $m = 12$



(e) Planning with NN-3 score function, $m = 12$
(f) Learning + planning, classical score function, $n = 5, m = 12$
(g) Learning + Planning, classical score function, $m = 12$
(h) Learning + Planning, NN-3 score function, $n = 5, m = 12$

Fig. 1. Experimental results

5.1 Learning

Classical score function First, we assume the adversary uses the classical score function in Eq (1). The defender learns this score function using the closed-form estimation (CF) in Theorem 1. We study how the learning accuracy changes with the size of training sample d. We sample the parameters of the true score function f uniformly at random from $[-0.5, 0.5]$. We then generate m feature configurations uniformly at random. For each of them, we sample the attacked target d/m times according to f, obtaining a training set of d samples. We generate a test set $\tilde{D}$ of 5×10^5 configurations sampled uniformly at random. We measure the learning error as the mean total variation distance between the attack distribution from the learned $\hat{f}$ and that of the true model f:

$$\frac{1}{|\tilde{D}|}\sum_{j=1}^{|\tilde{D}|} d_{TV}\left(\left(\frac{f(x_i^j)}{\sum_{t\in N} f(x_t^j)}\right)_{i\in N}, \left(\frac{\hat{f}(x_i^j)}{\sum_{t\in N} \hat{f}(x_t^j)}\right)_{i\in N}\right).$$

Figure 1a shows that the learning error decreases as we increase the number of samples. Theorem 1 provides a sample complexity bound, which we annotate in Figure 1a as well. The experiment shows that we need much fewer samples to learn a relatively good score function, and smaller games exhibit smaller learning error.

3-layer NN represented (NN-3) score function We assume the adversary uses a 3-layer neural network score function, whose details are in the full version of the paper. We use the gradient descent-based (GD) learning algorithm RMSProp as described in Section 3, with learning rate 0.1. For each sample size d, we generate d feature configurations and

sample an attacked target for each of them in the training set. Fig. 1b shows GD can minimize the learning error to below 0.15. Note that the training data are different in Fig. 1a and 1b, thus the two figures are not directly comparable.

We also measured $|\hat{\theta} - \theta|$, the L_1 error in the score function parameter θ, which directly relates to the sample complexity bound in Theorem 1. We include the results in the full version of the paper.

5.2 Planning

We test our algorithms on finding the optimal feature configuration against a known attacker model. The FDP parameter distributions are included in the full version.

Classical score function Fig. 1c shows that the binary search version of the MILP based on $\mathcal{MP}1$ (MILPBS) runs faster than that without binary search on most instances. MILPBS scales up to problems with 200 targets, which is already at the scale of many real-world problems. MILP does not scale beyond problems with 20 targets. In the arXiv version, we show that MILPBS also scales better in terms of the number of features. We set the MILP's error bound at 0.005 and $\epsilon_{\text{bs}} = 1e - 4$; the difference in the two algorithms' results is negligible.

NN-3 score function When the features are continuous without feasibility constraints, planning becomes a non-convex optimization problem. We can apply the gradient-based optimizer or non-convex solver. Recall that $U(x)$ is the defender's expected loss using feature configuration x. We measure the solution gap of alg $\in \{$Ipopt, GD, GREEDY$\}$ as $\frac{U(x^{\text{alg}}) - U(x^{\text{GD}})}{U(x^{\text{GD}})}$, where x^{alg} is the solution from the corresponding algorithm.

Fig. 1d and 1e show the running time and solution gap fixing $m = 12$. The running time of GD and GREEDY does not change much across different problem sizes, yet Ipopt runs slower than the former two on most problem instances. GD also has smaller solution gap than Ipopt and GREEDY. In the full version we show the number of features affect these metrics in a similar way.

5.3 Combining Learning and Planning

We integrate the learning and planning algorithms to examine our full framework. The defender learns a score function $\hat{f}$ using algorithm L. Then, she uses planning algorithm P to find an optimal configuration $x^{\text{L,P}}$ assuming $\hat{f}$. We measure the solution gap as $\frac{U(x^{\text{L,P}}) - U(x^*)}{U(x^*)}$, where x^* is the optimal feature configuration against the true attacker model, computed using MILPBS or GD.

Classical score function We test learning algorithm CF and planning algorithms P $\in$ $\{$MILP, MILPBS$\}$. Fig. 1f shows how the solution gap changes with the size of the training dataset. With $n \leq 20$ targets, all algorithms yield solution gaps below 0.1 (Fig. 1g). The reader might note the overlapping error bars, which are expected since MILP and MILPBS should not differ much in solution quality. Indeed, the difference is negligible as the smallest p-value of the 6 paired t-tests (fixing the number of targets for which they are tested) is 0.16.

NN-3 score function We test learning algorithm GD and planning algorithms P $\in$ {GD, Ipopt, GREEDY}. Fig. 1h shows how the solution gap changes with the size of training dataset d. Paired t-tests suggest that GD has significantly smaller solution gap than GREEDY ($p < 0.03$) at each size of training dataset except 1080. Ipopt also has significantly smaller solution gap than GREEDY ($p < 0.01$) when on large datasets with $d \geq 10^5$ samples. On the largest dataset $d = 10^6$, GD also performs significantly better than Ipopt ($p = 0.04$).

Compared to the case with classical score functions, more data are required here to achieve a small solution gap. Since learning error is small for both cases (Fig. 1a,1b), this suggests planning is more sensitive to NN-3 score functions than classical score functions.

5.4 Case Study: Credit Bureau Network

The financial sector is a major victim of cyber attacks due to its large amount of valuable information and relatively low level of security measures. In this case study, we ground our FDP model in a credit bureau's network. We show how feature deception improves the network security when the attacker follows a domain-specific rule-based behavioral model.

We note that the purpose of this case study is not to show the scalability of our algorithm: all previous experiments fulfill that purpose. Instead, here we demonstrate why deception is useful, how our algorithm yields deception strategies reasonable in the real world, and how our algorithm capably handles an attacker which does not conform to our assumed score function.

As shown in Table 2, we consider a network of 10 nodes (i.e. targets) with 6 binary features: operating system (Windows/Linux) and the availability of SMTP, NetBIOS, HTTP, SQL, and Samba services. Each node has a type of server running on it, which determines the features available on that node. Some nodes would incur a high loss if attacked, like the database servers, because for a credit bureau the safety of users' credit information is of utmost importance. Others might incur a low loss, such as the mail servers and the web server. Nodes of the same type might lead to different losses. For example, some database servers might have access to more information than others. Each feature has different switching cost c_k. For the operating system, the cost is $c_k = 5$. For SQL, Samba, and HTTP services, the cost is 2. The cost is 1 for others. The defender has a budget of 10. There is no constraint on switching each individual feature, i.e. $C(\hat{x}_{ik}) = \{0, 1\}$. However, we impose that Windows + Samba and Linux + NetBIOS cannot be present on the same node, as it is technically impossible to do so.

We demonstrate the entire learning and planning pipeline. We use an attacker's behavior model common in the security analysis. The attacker cares about a subset $M' \subseteq M$ of the features, and we call each such feature $k \in M'$ a requirement. The attack is uniformly randomized among the targets that satisfy the most requirements. Although this decision rule does not fit our classical score function, we can approximate it by giving large weights w_k to the requirement features, and 0 to the rest.

First, we consider an APT-like attacker, who wants to exfiltrate data by exploiting the SMTP service. They have expertise in Linux systems and want to maintain a high degree of stealth. Thus, their decision rule is based on the three requirement features:

Node type	Node ID	Actual features $\hat{x}_i$	Loss u_i
Mail server	0, 1	Windows, SMTP, NetBIOS	0.1
Web server	2	Windows, HTTP	0.2
App server	3, 4	Windows, SQL, NetBIOS	0.3
Database server	5,6,7	Linux, SQL, SMTP, Samba	0.4
Database server	8,9	Linux, SQL, SMTP, Samba	0.8

Table 2. Feature configuration of a typical credit bureau computer network.

Attacker	Solution x_i	Attacked nodes	Loss
APT	Node 1: Windows $\rightarrow$ Linux Node 1: SQL off $\rightarrow$ on Node 1: NetBIOS on $\rightarrow$ off Node 8, 9: SMTP on $\rightarrow$ off	5,6,7,8,9 $\rightarrow$1 ,5, 6, 7	$0.56 \rightarrow$ 0.325
Botnet	Node 3: NetBIOS on $\rightarrow$ off Node 4: NetBIOS on $\rightarrow$ off	0,1,3,4 $\rightarrow$0,1	$0.2 \rightarrow$ 0.1

Table 3. Learning + planning results for 2 types of attackers.

Linux, SMTP, and SQL. Without deception, the attacker would randomize attack over nodes 5-9, because these nodes satisfy 3 requirements and other nodes satisfy at most 2. As shown in Table 3, the optimal solution for the learning and planning problem leads to an expected defender's loss of 0.325, which is a 42% decrease from the loss with no deception. With limited budget, the defender makes the least harmful target, node 1, very attractive and the most harmful targets, nodes 8 and 9, less attractive.

We also consider a botnet attacker, who wants to create a bot by exploiting the NetBIOS service. They have expertise in Windows and want to maintain a moderate degree of stealth. Thus, their decision rule is based on two requirement features: Windows and NetBIOS. The results in Table 3 shows that the defender should set the NetBIOS observed value to be off for nodes 3 and 4, attracting the attacker to the least harmful nodes. This reduces the defender's expected loss by 50% compared to not using deception.

6 Related Work

Deception Deception has been studied in many domains, and of immediate relevance is its use in cybersecurity [26]. Studies have suggested that deceptively responding to an attacker's scanning and probing could be a useful defensive measure [12,2]. Schlenker et al. [27] and Wang and Zeng [32] propose game-theoretic models where the defender manipulates the query response to a known attacker. Proposing a domain-independent model, we advance the state of the art by (1) providing a unified learning and planning framework with theoretical guarantee which can deal with unknown attackers, (2) extending the finite "type" space in both papers, where "type" is defined by the combination of feature values, to an infinite feature space that allows for both continuous and discrete features, and (3) incorporating a highly expressive bounded rationality model whereas both papers assume perfectly rational attackers.

For the more general case, Horak et al. [9] study a defender that engages an attacker in a sequential interaction. A complementary view where the attacker aims at deceiving the defender is provided in [19,6]. Different from them, we assume no knowledge of the set of possible attacker types. In [35,7,19,6] deception is defined as deceptively allocating defensive resources. We study feature deception where no effective tools can thwart an attack, which is arguably more realistic in high-stakes interactions. When such tools exist, feature deception is still valuable for strategic defense.

Learning in Stackelberg games Much work has been devoted to learning in Stackelberg games. Our work is most directly related to that of Haghtalab et al. [8]. They show that three defender strategies are sufficient to learn a SUQR-like adversary behavior model in Stackelberg security games. The only decision variable in their model, the coverage probability, may be viewed as a single feature in FDP. FDP allows for an arbitrary number of features, and this realistic extension makes their key technique inapplicable for analyzing the sample complexity. Our main learning result also removes the technical constraints on defender strategies present in their work. Sinha et al. [29] study learning adversary's preferences in a probably approximately correct (PAC) setting. However, their learning accuracy depends heavily on the quality of distribution from which they sample the defender's strategies. We provide a uniform guarantee in a distribution-free context. Other papers [3,17,15,21] study the online learning setting with rational attackers. As pointed out in [8], considering the more realistic bounded rationality scenario allows us to make use of historical data and use our algorithm more easily in practice.

Planning with boundedly rational attackers Yang et al. [34] propose a MILP-based solution in security games. Our planning algorithm goes beyond the coverage probability and determines the configuration of multiple features, and adopt a more expressive behavior model. The subsequent papers that incorporate learning with such bounded rationality models do not provide any theoretical guarantee [33,5]. A recent work develops a learning and planning pipeline in security games [22]. However, their algorithm requires the defender know a priori some parameters in the attacker's behavior model, and provides no global optimality guarantee.

7 Discussion

We conclude with a few remarks regarding the generality and limitations of our work. First, our model allows the attacker to have knowledge of deception if the knowledge is built into their behavior. For example, the attacker avoids attacking a target because it is "too good to be true". This can be captured by a score function that assigns a low score for such a target.

Second, our model can handle sophisticated attackers who can outstrip deception. A singleton feasible set $C(\hat{x}_{ik})$ implies the defender knows the attacker can find out the actual value of a feature. As an important next step, we will study the change of attacker's belief of deception over repeated interactions.

Third, typically, actual features on functional targets are environmental parameters beyond the defender's control, or at least have high cost of manipulation. Altering them

and defender's losses u_i does not align conceptually with deception. Thus, we treat them as fixed. For a target with no fixed actual values, e.g., a honeypot, the defender's cost is just the cost of configuring the feature, e.g., installing Windows. For consistency, we can set $\hat{x}_{ik}$ as the feature value with the lowest configuration cost, and η_{ik} is the additional cost for a different feature value.

Fourth, the attacker's preference might shift when there is a major change in security landscape, e.g. a new vulnerability disclosed. In such case, a proactive defender will recalibrate the system: recompute the attacker's model and reconfigure the features. Moreover, exactly because the defender has learned the preferences before the change using our algorithms, the defender now knows better what qualifies as a major change. Our algorithms are fast enough for a proactive defender to run regularly.

Fifth, when faced with a group of attackers, in FDP we learn an average behavioral model of the population. To handle multiple attacker types, one could refer to the literature on Bayesian Stackelberg games [20].

Finally, in FDP the defender uses only pure strategies. In many domains such as cybersecurity, frequent system reconfiguration is often too costly. Thus, the system appears static to the attacker. We leave to future work to explore mixed strategies in applications where they are appropriate.

Acknowledgments

This research was sponsored by the Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on.

References

1. Abbasi, Y., Kar, D., Sintov, N., Tambe, M., Ben-Asher, N., Morrison, D., Gonzalez, C.: Know your adversary: Insights for a better adversarial behavioral model. In: CogSci (2016)
2. Albanese, M., Battista, E., Jajodia, S.: Deceiving attackers by creating a virtual attack surface. In: Cyber Deception (2016)
3. Blum, A., Haghtalab, N., Procaccia, A.D.: Learning optimal commitment to overcome insecurity. In: NIPS (2014)
4. Chiang, C.Y.J., Gottlieb, Y.M., Sugrim, S.J., Chadha, R., Serban, C., Poylisher, A., Marvel, L.M., Santos, J.: Acyds: An adaptive cyber deception system. In: MILCOM (2016)
5. Fang, F., Stone, P., Tambe, M.: When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In: IJCAI (2015)
6. Gan, J., Xu, H., Guo, Q., Tran-Thanh, L., Rabinovich, Z., Wooldridge, M.: Imitative follower deception in stackelberg games. In: EC (2019)
7. Guo, Q., An, B., Bosanskỳ, B., Kiekintveld, C.: Comparing strategic secrecy and stackelberg commitment in security games. In: IJCAI. pp. 3691–3699 (2017)

8. Haghtalab, N., Fang, F., Nguyen, T.H., Sinha, A., Procaccia, A.D., Tambe, M.: Three strategies to success: Learning adversary models in security games. In: IJCAI (2016)
9. Horák, K., Zhu, Q., Bošanskỳ, B.: Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In: GameSec (2017)
10. Hurlburt, G.: "good enough" security: The best we'll ever have. Computer (2016)
11. Jafarian, J.H., Al-Shaer, E., Duan, Q.: Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the first workshop on Hot topics in software defined networks. ACM (2012)
12. Jajodia, S., Park, N., Pierazzi, F., Pugliese, A., Serra, E., Simari, G.I., Subrahmanian, V.: A probabilistic logic of cyber deception. IEEE Trans. Inf. Forensics Secur. **12**(11) (2017)
13. Jajodia, S., Subrahmanian, V., Swarup, V., Wang, C.: Cyber deception. Springer (2016)
14. Latimer, J.: Deception in War. John Murray (2001)
15. Letchford, J., Conitzer, V., Munagala, K.: Learning and approximating the optimal strategy to commit to. In: SAGT (2009)
16. Lyon, G.F.: Nmap network scanning: The official Nmap project guide to network discovery and security scanning. Insecure (2009)
17. Marecki, J., Tesauro, G., Segal, R.: Playing repeated stackelberg games with unknown opponents. In: AAMAS (2012)
18. Nakashima, E.: To thwart hackers, firms salting their servers with fake data
19. Nguyen, T.H., Wang, Y., Sinha, A., Wellman, M.P.: Deception in finitely repeated security games. In: AAAI (2019)
20. Paruchuri, P., Pearce, J.P., Marecki, J., Tambe, M., Ordonez, F., Kraus, S.: Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In: AAMAS (2008)
21. Peng, B., Shen, W., Tang, P., Zuo, S.: Learning optimal strategies to commit to. In: AAAI (2019)
22. Perrault, A., Wilder, B., Ewing, E., Mate, A., Dilkina, B., Tambe, M.: Decision-focused learning of adversary behavior in security games. arXiv preprint arXiv:1903.00958 (2019)
23. Potter, B., Day, G.: The effectiveness of anti-malware tools. Computer Fraud & Security (2009)
24. Provos, N., et al.: A virtual honeypot framework. In: USENIX Security Symposium (2004)
25. PwC: Operation Cloud Hopper Technical Annex
26. Rowe, N.C.: Deception in defense of computer systems from cyber attack. In: Cyber Warfare and Cyber Terrorism. IGI Global (2007)
27. Schlenker, A., Thakoor, O., Xu, H., Fang, F., Tambe, M., Tran-Thanh, L., Vayanos, P., Vorobeychik, Y.: Deceiving cyber adversaries: A game theoretic approach. In: AAMAS (2018)
28. Shamsi, Z., Nandwani, A., Leonard, D., Loguinov, D.: Hershel: single-packet os fingerprinting. In: ACM SIGMETRICS Performance Evaluation Review (2014)
29. Sinha, A., Kar, D., Tambe, M.: Learning adversary behavior in security games: A pac model perspective. In: AAMAS (2016)
30. Spitzner, L.: The honeynet project: Trapping the hackers. IEEE Security & Privacy (2003)
31. Stancu-Minasian, I.M.: Fractional programming: theory, methods and applications, vol. 409. Springer Science & Business Media (2012)
32. Wang, W., Zeng, B.: A two-stage deception game for network defense. In: GameSec (2018)
33. Yang, R., Ford, B., Tambe, M., Lemieux, A.: Adaptive resource allocation for wildlife protection against illegal poachers. In: AAMAS (2014)
34. Yang, R., Ordonez, F., Tambe, M.: Computing optimal strategy against quantal response in security games. In: AAMAS (2012)
35. Yin, Y., An, B., Vorobeychik, Y., Zhuang, J.: Optimal deceptive strategies in security games: A preliminary study. In: AAAI Symposium on Applied Computational Game Theory (2014)