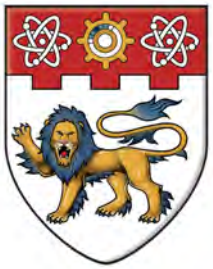


NANYANG TECHNOLOGICAL UNIVERSITY

ADVANCED ATTACK AND DEFENSE TECHNIQUES IN MACHINE LEARNING SYSTEMS

MENGCHEN ZHAO

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2019

ADVANCED ATTACK AND DEFENSE TECHNIQUES IN MACHINE LEARNING SYSTEMS

MENGCHEN ZHAO

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A thesis submitted to Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2019

Acknowledgements

Reaching the end of my PhD study, I am very grateful for those people who helped and accompanied with me. This four-year experience teaches me how to behave professionally and act like a grown-up, even though I really don't want to grow up. I have seen many brilliant and excellent people, who always inspire me to become better, and be humble.

At the first place, I would express my sincere gratitude to my supervisor Bo An. Unlike many teachers I have been met in past years, Bo teaches me not only knowledge, but also the methodology to do research and the principals to work with people. I really learned a lot from him during these years: the enthusiasm for research, the professional behaviors in work and the strong will to chase excellency. I am very lucky to be his student and I hope that I could let him be proud of me someday.

I would also thank my thesis advisory committee members: Xiaohui Bei and Dusit Niyato. Those discussions and meetings help me a lot in improving my work, and build a better vision for me in doing research.

During the past four years in NTU, I am grateful to have the privilege to collaborate with so many excellent researchers: Christopher Kiekintveld, Wei Gao, Teng Zhang, Yaodong Yu, Sulin Liu, Sinno Jialin Pan, Xiaobo Ma, Yanhai Xiong and Haipeng Chen. I thank you for your guidance and insights in those projects we worked on together as well as the patience of mentoring.

I am very fortune to spend four years of time with these cute people in our research group: Qingyu Guo, Yanhai Xiong, Haipeng Chen, Zhen Wang, Jiarui Gan, Xinrun Wang, Martin Strobel, Youzhi Zhang, Wanyuan Wang, Jiuchan Jiang, Xu He, Lei Feng, Aye Phyu. Thank you all for giving me so much happiness, laughs and help and I would never forget these precious memories.

Most importantly, I want to thank my parents for their endless support and love. They

always remember the deadlines of my work even though I just unintentionally mentioned them months ago. They care every details of my life, even more than themselves. Mom and dad, thank you and I love you!

Contents

List of Figures

List of Tables

Abstract

The security of machine learning systems has become a great concern in many real-world applications involving adversaries, including spam filtering, malware detection and e-commerce. There is an increasing trend of study on the security of machine learning systems but the current research is still far from satisfactory. Towards building secure machine learning systems, the first step is to study their vulnerability, which turns out to be very challenging due to the variety and complexity of machine learning systems. Combating adversaries in machine learning systems is even more challenging due to the strategic behavior of the adversaries.

This thesis studies both the adversarial threats and the defenses in real-world machine learning systems. Regarding the adversarial threats, we begin by studying label contamination attacks, which is an important type of data poisoning attacks. Then we generalize the conventional data poisoning attacks on single-task learning models to multi-task learning models. Regarding defending against real-world attacks, we first study the spear phishing attacks in email systems and propose a framework for optimizing the personalized email filtering thresholds to mitigate such attacks. Then, we study the fraud transactions in e-commerce systems and propose a deep reinforcement learning based impression allocation mechanism for combating fraudulent sellers. The specific contributions of this thesis are listed below.

First, regarding the label contamination attacks, we develop a Projected Gradient Ascent (PGA) algorithm to compute attacks on a family of empirical risk minimizations and show that an attack on one victim model can also be effective on other victim models. This makes it possible that the attacker designs an attack against a substitute model and transfers it to a black-box victim model. Based on the observation of the transferability, we develop a defense algorithm to identify the data points that are most likely to be attacked. Empiri-

cal studies show that PGA significantly outperforms existing baselines and linear learning models are better substitute models than nonlinear ones.

Second, in the study of data poisoning attacks on muti-task learning models, we formulate the problem of computing optimal poisoning attacks on Multi-Task Relationship Learning (MTRL) as a bilevel program that is adaptive to arbitrary choice of *target* tasks and *attacking* tasks. We propose an efficient algorithm called PATOM for computing optimal attack strategies. PATOM leverages the optimality conditions of the subproblem of MTRL to compute the implicit gradients of the upper level objective function. Experimental results on real-world datasets show that MTRL models are very sensitive to poisoning attacks and the attacker can significantly degrade the performance of target tasks, by either directly poisoning the target tasks or indirectly poisoning the related tasks exploiting the task relatedness. We also found that the tasks being attacked are always strongly correlated, which provides a clue for defending against such attacks.

Third, on defending against spear phishing email attacks, we consider two important extensions of the previous threat models. First, we consider the cases where multiple users provide access to the same information or credential. Second, we consider attackers who make sequential attack plans based on the outcome of previous attacks. Our analysis starts from scenarios where there is only one credential and then extends to more general scenarios with multiple credentials. For single-credential scenarios, we demonstrate that the optimal defense strategy can be found by solving a binary combinatorial optimization problem called PEDS. For multiple-credential scenarios, we formulate it as a bilevel optimization problem for finding the optimal defense strategy and then reduce it to a single level optimization problem called PEMS using complementary slackness conditions. Experimental results show that both PEDS and PEMS lead to significant higher defender utilities than two existing benchmarks in different parameter settings. Also, both PEDS and PEMS are more robust than the existing benchmarks considering uncertainties.

Fourth, on combating fraudulent sellers in e-commerce platforms, we focus on improving the platform's impression allocation mechanism to maximize its profit and reduce the sellers' fraudulent behaviors simultaneously. First, we learn a seller behavior model to predict the sellers' fraudulent behaviors from the real-world data provided by one of the largest e-commerce company in the world. Then, we formulate the platform's impression allocation problem as a continuous Markov Decision Process (MDP) with unbounded action space. In order to make the action executable in practice and facilitate learning, we propose a novel deep reinforcement learning algorithm DDPG-ANP that introduces an action norm penalty to the reward function. Experimental results show that our algorithm significantly outperforms existing baselines in terms of scalability and solution quality.

Chapter 1

Introduction

Thanks to the recent success of machine learning technologies, we have witnessed a rapid growth of machine learning applications, which make our daily lives more convenient and interesting. For example, the face recognition technologies allow us to unlock our phones by just taking a look at them; the enhanced email filtering technologies keep us away from numerous spam and phishing emails; the speech recognition technologies allow us to request a song by simply talking to an intelligent speaker; the autonomous driving technologies would set us free from driving in the near future. Unfortunately, the vulnerabilities of machine learning systems open a new door for the adversaries, who could potentially compromise a system by exploiting such vulnerabilities. The security of machine learning systems has been a critical concern for adversarial applications such as spam filtering, intrusion detection, malware detection and fraud detection in e-commerce [1, 2].

As Chinese strategist Sun Tzu said, "Know yourself and your enemy then you will never be defeated." Towards securing machine learning systems, the first step is to understand how the adversaries would launch attacks and what damages can the attacks cause. Therefore, it is very important to study the attack techniques. Although a random or a heuristic attack strategy could also cause damages to machine learning systems, it is more interesting and

necessary to study the optimal attacks, because the adversaries in real world could be very intelligent. However, it turns out that finding the optimal attack strategies are very challenging. First, identifying the attacker's strategy space is challenging because the attackers' capabilities vary in different application domains. Second, it is algorithmic challenging to develop efficient and effective attack strategies due to the problem scale and the complexity of the machine learning system.

Defending machine learning systems is even more challenging than attacking them. First, the defender's capabilities vary in different machine learning systems and we need to identify her strategy space. Second, the robustness of defense strategies must be considered because a machine learning system might face various, perhaps irrational attackers. Third, in many scenarios such as email filtering, the defenders usually need to sacrifice some economy for security. Therefore, the trade-off between economy and security becomes a tricky problem. Last, in complex and highly dynamic systems such as e-commerce platforms, it is almost impossible to exactly solve for the optimal defense strategies. Therefore, model-free methodologies need to be developed for finding good defense strategies.

This thesis dedicates to reveal vulnerabilities of machine learning algorithms and develop defense strategies for combating adversaries in real-world machine learning systems. First, we study data poisoning attacks, in particular label contamination attacks, against a class of machine learning models. Second, we extend data poisoning attacks on single-task learning models to multi-task learning models. Third, we study how to optimally set the personalized email filtering thresholds to defend against spear phishing attacks. Fourth, we investigate how to combat fraudulent sellers in e-commerce through adaptive buyer impression allocation. Detailed problem descriptions and our contributions are as follows.

1.1 Adversarial Threats in Machine Learning Systems

At the first place, we focus on exploring vulnerabilities of machine learning systems in algorithmic level. Adversarial threats to machine learning algorithms can be classified as two categories: exploratory attacks and causative attacks [3]. The exploratory attacks exploit the vulnerabilities of trained models (e.g., classifiers) but do not affect the training phase. For example, hackers can obfuscate malware code in order to bypass the malware detection. The causative attacks (also known as poisoning attacks) target at the training phase, where the adversaries aim to make the learned model in favor of them by manipulating the training data. For example, the adversaries in recommender systems can manipulate the recommendations by providing deliberately calculated ratings [4]. Poisoning attacks are usually more dangerous than exploratory attacks because the adversaries could potentially control the whole system. We first study label contamination attack, which is an important type of poisoning attacks. We also propose a methodology for attacking black-box learning models. Then, we extend the data poisoning attacks from single-task learning models to multi-task learning models.

1.1.1 Label Contamination Attacks

Label contamination attacks usually happen when labels of training data are collected from external sources. For example, one can use crowdsourcing platforms (e.g., Amazon Mechanical Turk) to collect labels from human workers; Netflix relies on users' ratings to improve their recommendation systems; collaborative spam filtering updates the email classifier periodically based on end-users' feedback, where malicious users can mislabel emails in their inboxes to feed false data to the updating process.

We study the label contamination attack against a broad family of binary classification models. We focus on answering three questions that have not been addressed by existing

work. First, consider a highly motivated attacker with full knowledge of the victim learning model, how to compute the label contamination attack against the victim model? Second, if the victim learning model is a black-box, how does the attacker design effective attacks against it? Third, how to defend against the label contamination attacks?

Previous work on label contamination attacks has three limitations [5, 6]. First, they restrict the attacker's goal to decrease the accuracy of a victim learning model, whereas in reality the attackers may have arbitrary objectives. Second, they focus on computing attacks against Support Vector Machines (SVMs) and their algorithms cannot generalize to other victim learning models. Third, they assume that the attacker has full knowledge of the victim learning model, which might be unrealistic in reality. Regarding the defense against poisoning attacks, there are generally two lines of research: robust learning focuses on improving the robustness of learning algorithms under contaminated data [5], and data sanitization focuses on removing suspicious data from training set [7, 8]. Most robust learning and data sanitization techniques require a set of clean data, which is used to develop metrics for identifying future contaminated data. However, such techniques become useless when the set of clean data is hard to obtain, or is contaminated by the attacker.

We make four key contributions. First, we extend the existing work on label contamination attack to allow a broad family of victim models and arbitrary attacker objectives. We formulate the optimal attack problem as a mixed-integer bilevel program. Second, we exploit the Representer Theorem [9] and propose a Projected Gradient Ascent (PGA) algorithm to approximately solve the bilevel program. Third, we propose a substitute-based attack method for attacking black-box learning models, which leverages the *transferability* of label contamination attacks. To our knowledge, we are the first to study the transferability of poisoning attacks. Finally, we empirically analyze the transferabilities with respect to five representative substitute models and show that linear models are significantly better substitutes than nonlinear ones.

1.1.2 Poisoning Attacks on Multi-Task Learning

Traditional research on data poisoning attacks, including label contamination attacks and data injection attacks, focus on single-task learning models. In this work, we formally analyze optimal poisoning attacks on multi-task learning (MTL) models, where multiple tasks are learned jointly to achieve better performance than single-task learning [10]. Specifically, we focus on multi-task relationship learning (MTRL) models, a popular subclass of MTL models where task relationships are quantized and are learned directly from training data [11, 12]. Many MTL-based machine systems collect training data from individual users to provide personalized services, including collaborative spam filtering and personalized recommendations, which makes them vulnerable to poisoning attacks launched by cyber criminals. For example, in an MTL-based recommender system, attackers can control a considerable number of user accounts either by hacking existing user accounts or creating fictitious user accounts.

Previous works on poisoning attacks focus on single-task learning (STL) models, including support vector machines [13], autoregressive models [14] and factorization-based collaborative filterings [4]. However, none of them study poisoning attacks on MTL models. Computing optimal poisoning attacks on MTL models can be much more challenging than on STL models, because MTL tasks are related with each other and an attack on one task might potentially influence other tasks. This also opens a door for the attacker to attack some accessible tasks and indirectly influence the unaccessible target tasks, which cannot be addressed by existing methods on poisoning STL models.

The major contributions of our work are threefold. First, we formulate the optimal poisoning attack problem on MTRL as a bilevel program that is adaptive to any choice of target tasks and attacking tasks. Second, we develop a stochastic gradient ascent based algorithm called PATOM for solving the optimal attack problem, where the gradients are computed

based on the optimality conditions of the convex subproblem of MTRL. Third, we demonstrate experimentally that MTRL is very sensitive to data poisoning attacks. The attacker can significantly degrade the performance of target tasks, by either directly poisoning the target tasks or indirectly poisoning the related tasks. Moreover, we study the change of task relationships under attacks and found that the attacking tasks usually have strong local correlations, which suggests that a group of strongly correlated tasks could be dangerous to the learner.

1.2 Combating Adversaries in Machine Learning Systems

Combating adversaries in machine learning systems highly depends on the application scenarios, because in different systems the adversaries' attack methods, defenders' defense actions and the system structures can be highly different. We study how to combat adversaries in two different systems. First, we investigate how to defend email filtering systems against spear phishers, who are much more harmful than ordinary spammers. Then, we focus on the fraudulent sellers in e-commerce systems, who generate fake transactions in order to escalate their reputations.

1.2.1 Combating Sequential Spear Phishers

Email is not a secure communications channel, and attackers have exploited this via spam emails for many years. However, in recent years cyber attacks using email have become increasingly targeted and much more damaging to organizations [15]. These targeted email attacks are commonly known as spear phishing. They target individuals or small groups of people, but use personal information and social engineering to craft very believable messages with the goal of inducing the recipient to open an attachment, or visit an unsafe website by

clicking a link. Executing a spear phishing attack is much more costly than sending a broad spam message, but it is also much more likely to succeed and the potential damage is much greater. For example, in 2011 the RSA company was breached by a spear phishing attack [16]. This attack resulted in privileged access to secure systems, and stolen information related to the company's SecurID two-factor authentication products.

Email filtering systems are one of the primary defenses against both spam and spear phishing attacks. These systems typically use black and white lists as well as machine learning methods to score the likelihood that an email is malicious before sending it to a user [17]. Setting the threshold for how safe a message must be to be delivered is a key strategic decision for the network administrator [18]. If the threshold is too high, malicious emails will easily pass the filtering system, but if the threshold is too low normal emails will be filtered. Recent work has proposed a game-theoretic model that can improve the effectiveness of filtering if thresholds are personalized according to individuals' values and susceptibilities [19]. They assume that the attacker's strategy is simply selecting a subset of users to attack that maximizes an additive expected reward, ignoring the cost of attacks and the outcome of previous attacks.

However, in many incidents such as Operation Aurora [20], attackers launches sophisticated attacks toward few targets over months. In such cases, attackers have plenty of time and attack resources and they can plan long term sequential attack strategies to achieve difficult objectives [21]. In this work we extend the literature (and particularly the personalized filtering model of [19]) to consider more sophisticated attackers who can make sequential decisions about which users to send spear phishing emails to. Specifically, we consider more complex (also realistic) objective functions for both the attacker and defender, including modeling attack costs, and situations where it is only necessary to compromise one user from a set of users that has access to important data or credentials (i.e., the user values are substitutable).

Our contributions are fourfold. First, we consider the case where there is a single important credential that the attacker seeks to gain and model the attacker's decision making as a Markov Decision Process (MDP). We formulate a bilevel optimization problem for the defender and show that the attacker's problem (i.e., lower level problem) can be solved by a linear program. Solving the linear program is computational consuming since the number of variables and constraints grow exponentially with the number of users. Our second contribution is to find a simplified representation of the defender's utility and thus reduce the defender's bilevel program into a single level binary combinatorial optimization program (which we call PEDS) by exploiting the structure of the attacker's MDP. Our third contribution is to extend the single-credential case to a a more general case where there could be multiple sensitive credentials. For the multiple-credential case, the defender's utility cannot be represented in the same way as the single-credential case. We consider the dual program of the linear program that solves the attacker's MDP and represent the defender's loss from spear phishing attacks by a linear combination of dual variables. We then propose a single level formulation (which we call PEMS) for the defender, which is reduced from the proposed bilevel problem using complementary slackness conditions. Our fourth contribution is to evaluate PEDS and PEMS by comparing our solutions with two existing benchmarks and show that our solutions lead to significant higher defender utilities in different parameter settings and are also robust considering uncertainties.

1.2.2 Combating Fraudulent Sellers in E-Commerce

One of the major functions of the e-commerce platforms is to guide buyer impressions to sellers, where *a buyer impression means one buyer click on a product*. Buyer impressions are usually allocated through a ranking system that displays the sellers's products in some order based on their quality scores. In order to increase the total number of transactions, the platform tends to allocate more buyer impressions to popular products, where the popularity

usually reflects as the conversion rate, i.e., the probability that a buyer buys the product if he clicks on it. From the seller's perspective, they usually spend much effort on getting more buyer impressions and orders. A legal approach to obtain more buyer impressions is advertising. However, due to the high cost of advertising, many sellers choose illegal ways to make their products look popular to obtain more buyer impressions [22]. The most common approach to increase the products' popularity is through faking transactions, where sellers control a number of buyer accounts and use them to buy their own products and provide positive feedback [23, 24]. Such fraudulent behaviors severely decrease the effectiveness of impression allocation and jeopardize the business environment.

Currently, e-commerce platforms mainly rely on fraud detection techniques to combat fraudulent behaviors [25–27]. However, given the fact that there is no perfect fraud detection system, it is necessary to explore alternative approaches for combating fraud in e-commerce. In this work, we take a mechanism design approach to address this problem. Existing approaches for impression allocation mechanism focus on maximizing the profit of the platform, ignoring the influence on sellers' fraudulent behaviors. However, a mechanism that maximizes the platform's profit might also induce more fraudulent behaviors, which have long-term negative effect on the platform. Our objective is to improve the platform's impression allocation mechanism to maintain the platform's profit and reducing the fraudulent behaviors at the same time.

A recent line of works introduces deep reinforcement learning to e-commerce mechanism design [28–30]. However, their approaches are impractical for real-world applications. First, they model the platform's action as an $n-$dimensional vector, where each entry of the vector represents the number of impressions to be allocated to a seller. However, there are millions of sellers in the real world and their approach cannot scale up due to the high-dimensional action space. Second, the outputted actions are not executable in practice since the number of impressions that a seller actually receives depends on a complex parameterized ranking

system. Third, they directly apply the Deep Deterministic Policy Gradient (DDPG) [31] with a softmax output layer in the actor network, which makes the allocation of buyer impressions more smooth. However, in practice, the distribution of buyer impressions is very sharp because the products in the first few pages account for the most buyer clicks.

In this work, we focus on improving the platform's impression allocation mechanism considering both the platform's profit and fraudulent behaviors. Our contributions are fourfold. First, we learn a seller behavior model to predict the sellers' fraudulent behaviors using real-world data, which is one of the largest e-commerce platforms in the world. Second, we formulate the platform's decision making problem as an MDP, where the platform's action is to determine the parameters of the ranking system so that the dimensionality of the action space does not grow with the number of sellers. Third, as DDPG performs poorly in our problem, we propose a novel algorithm Deep Deterministic Policy Gradient with Action Norm Penalty (DDPG-ANP), where the norm of the agent's action is included in the reward function to facilitate learning in an unbounded action space. Fourth, we evaluate DDPG-ANP with DDPG and several baselines in terms of scalability and solution quality. Experimental results show that DDPG-ANP outperforms all baselines.

1.3 Thesis Organization

The structure of this thesis is organized as follows: **Chapter 2** reviews the related works to provide context of this thesis. **Chapter 3** considers label contamination attacks against black-box supervised learning models. **Chapter 4** investigates data poisoning attacks on multi-task relationship learning models. **Chapter 5** studies spear phishing attacks and optimally setting the email filtering thresholds to defend such attacks. **Chapter 6** focuses on fraudulent sellers in e-commerce and explores combating such sellers through optimal impression allocation. **Chapter 7** summarizes the thesis and presents possible directions for

future work.

Chapter 2

Related Work

In this chapter, we review existing research that is relevant to this thesis. First, we review existing works on adversarial machine learning, including data poisoning attacks, test-time attacks and their transferabilities. Since we study the data poisoning attacks on multi-task learning, we will also introduce the related works in multi-task learning. Moreover, we will review existing studies on spear phishing attacks, which is a hot research topic in the field of cyber security. Finally, we will revisit the related methods for combating fraudulent sellers in e-commerce.

2.1 Adversarial Machine Learning

The security of machine learning has attract many research attentions over the last decade. [3] conclude and provide a taxonomy of different types of attacks on machine learning techniques and systems. They also discuss a variety of possible defenses against adversarial attacks. [32] investigate both causative attacks (a.k.a. poisoning attacks or training-time attacks) and exploratory attacks (a.k.a. test-time attacks) with case studies. In particular, they give methodologies for modeling the attacker's capabilities and behaviors, and explore

the limits of an adversary's knowledge about the learning process. A recent book summarizes the advances of adversarial machine learning techniques, including the attacks on deep learning and approaches for improving robustness of deep neural networks [33]. We will introduce training-time attacks and test-time attacks respectively.

2.1.1 Training-time Attacks

Training-time attacks against machine learning algorithms has become an emerging research in the field of adversarial machine learning. [34] study a simple poisoning attack on online centroid anomaly detection, where attacker shifts the normal ball by injecting malicious data so that the ball accepts an anomaly point. Poisoning attacks on machine learning algorithms are pineered by [13], which studies the poisoning attacks on Support Vector Machines (SVMs), where the attacker is allowed to progressively inject malicious points to the training data in order to maximize the classification error. In recent years, poisoning attack is generalized to many popular machine learning techniques, including regression models [35], feature selection models [36], autoregressive models [14], latent Dirichlet allocation [37] and factorization-based collaborative filtering [4]. An algorithmic framework for identifying the optimal training set attacks is provided in [38]. Poisoning attacks is intrinsically a bilevel optimization problem, where the lower level problem minimizes the learner's training error and the upper level problem maximizes the attacker's utility along with some constraints on the attacker. The main challenge is to develop algorithms for efficiently solving the bilevel problem given that the size of the problem is usually big. Note that all of the aforementioned works assume that the attacker has full knowledge of the learning model. It is still unclear how to launch poisoning attacks against black-box learning models.

While data poisoning attacks against single-task learning have been studied extensively, few consider poisoning attacks against multi-task learning (MTL) algorithms. In order to find

out how to carry out poisoning attacks on MTL, we need to introduce some basics of MTL. MTL is class of machine learning algorithms that focus on learning multiple tasks simultaneously. In general, MTL can be categorized into four classes: feature learning approaches, low-rank approaches, task clustering approaches, and task relationship approaches. Feature learning approaches aim to learn a common shared feature space among multiple tasks to boost the learning performance of each task [39]. Low-rank approaches assume that the model parameters of different tasks share a low-rank structure, and discovery of such a low-rank structure could help learning a more precise model for each task [40]. Task clustering approaches assume that different tasks form several task-clusters, each of which consists of similar tasks [41]. Task relationship learning aims to quantify and learn task relationship automatically from data, such that knowledge can be transferred among related tasks [11]. However, as we discussed, the vulnerability of MTL has never been studied. In this work, we fill the gap by investigating the vulnerability of task relationship learning approaches, which have proven to be effective in MTL.

2.1.2 Test-Time Attacks

In classical machine learning, the underlying distribution of test data is assumed to be stationary. However, in many adversarial applications such as spam filtering and malware detection, adversaries can actively change the distribution of test data by crafting test samples, which are also referred to adversarial samples [42]. Such attacks are called exploratory attacks [3] or evasion attacks [42]. There is an extensive study of exploratory attacks on traditional machine learning models, including classification [1, 43], feature selection models [44] and kernel machines [45]. Recently, exploratory attacks on deep learning models have attracted more and more interests [46, 47].

There has been existing work that studies the transferability of exploratory attack where

the attacker perturbs the legitimate inputs to induce the trained classifier to misclassify them. The transferability of such an attack means that the inputs perturbed to induce one classifier can also induce other classifiers to produce misclassifications. The transferability of evasion attack among deep neural networks (DNNs) is demonstrated by [48]. Then, an extensive study explored the transferability of evasion attacks among five classifiers, including SVM, logistic regression, decision tree, k-nearest neighbors and DNNs [49]. Moreover, [50] proposed an ensemble-based adversarial example crafting method for attacking black-box learning models. Similarly, such transferability can be observed in poisoning attacks. Unfortunately, none of the existing work studies the transferability of poisoning attacks.

2.2 Combating Spear Phishing Attacks

Spear phishing has been an important type of threats to companies, institutes and organizations [51]. Some behavioral science studies focus on improving humans' awareness of phishing attacks [52, 53]. For example, [54] conducted a large-scale experiment that tracked workers' reactions to a series of carefully crafted spear phishing emails and a variety of immediate training and awareness activities. Others dedicate to improve the email filter's ability to identify spear phishing emails by considering more domain-specific features [17, 55]. One can refer to [56] for more detailed approaches for fighting against spear phishing attacks. A recent work tries to improve the performance of the email filter by using personalized email filtering thresholds [19], which is then extended to the multi-defender scenarios [57]. Moreover, the method of setting personalized thresholds has been extended to general intrusion detection systems [58]. However, they neglect the fact that most spear phishing attacks are rather sequential attacks instead of not one-shot attacks. Therefore, the sequential nature of the attackers has to be considered in designing defense strategies.

2.3 Combating Fraud in E-Commerce

In e-commerce, fraud transactions mean that the sellers making fake transactions either with themselves or their conspiracies, such to improve their reputation and gain more profit in the future. As sellers dedicate to make fraud transactions look like normal ones, a detection system is usually employed to identify fraud transactions. As part of fraud transactions in e-commerce, credit card fraud detection has been extensively studied [59, 60]. E-commerce companies usually employ systematic solutions to defend against fraud transactions. For example, Alibaba's anti-fraud system TFS incorporate graph-based detection module and time series based setection module to achieve real-time fraud detection [61]. For another example, JD's fraud detection system CLUE captures detailed information on users' click actions using neural-network based embedding, and models sequences of such clicks using a recurrent neural network.

Besides fraud detection, reinforcement mechanism design is another approach that has been proved effective in reducing fraudulent behaviors. Reinforcement mechanism design is a reinforcement learning framework that automatically optimizes mechanisms, without making too many unrealistic assumptions [28]. This framework has been applied to dynamic pricing in sponsored search auctions [62] and impression allocation in e-commerce [29, 30]. A key challenge in applying reinforcement mechanism design in e-commerce is the scalability issue since there are potentially millions of sellers on real-world e-commerce platforms. Instead of computing an impression allocation strategy for each seller [29, 30], we directly optimize the parameters of the ranking system to avoid the high dimensional action space and significantly reduce the training time.

Chapter 3

Label contamination attacks

This chapter makes extensive study on label contamination attacks (LCAs). LCA is an important type of data poisoning attack where an attacker manipulates the labels of training data to make the learned model beneficial to him. Existing work on LCA assumes that the attacker has full knowledge of the victim learning model, whereas the victim model is usually a black-box to the attacker. In this work, we develop a Projected Gradient Ascent (PGA) algorithm to compute LCAs on a family of empirical risk minimizations and show that an attack on one victim model can also be effective on other victim models. This makes it possible that the attacker designs an attack against a substitute model and transfers it to a black-box victim model. Empirical studies show that PGA significantly outperforms existing baselines and linear learning models are better substitute models than nonlinear ones.

3.1 LCAs on Binary Classification Models

In this section, we first introduce the label contamination attack against linear classifiers and formulate the optimal attack problem as a bilevel optimization problem. Then, we generalize our framework to solve the optimal attack against nonlinear kernel machines. We begin

by introducing the linear binary classification problem. Given a set of training data $D = \{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbb{R}^k, y_i \in \{-1, +1\}\}_{i=1}^n$, a linear classifier can be solved from the following optimization problem.

$$\min_{f \in \mathcal{H}} \quad C \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \frac{1}{2}||f||^2 \tag{3.1}$$

where $f(\mathbf{x}_i) = \mathbf{w}^\intercal \mathbf{x}_i + b$ is the decision function, $||f||^2 = ||\mathbf{w}||^2$ is square of the ℓ_2 norm of $\mathbf{w}$, $\mathcal{H}$ is the hypothesis space, L is the loss function and C is the regularization parameter. For a testing instance $\mathbf{x}_i$, its predicted label is $sgn(f(\mathbf{x}_i))$. Without loss of generality, we denote $\mathbf{x}_i$ as $(1, \mathbf{x}_i)$ and denote $\mathbf{w}$ as $(b, \mathbf{w})$ so that $f(\mathbf{x}_i)$ can be equivalently represented by $\mathbf{w}^\intercal \mathbf{x}_i$, where $\mathbf{w} \in \mathbb{R}^{k+1}$.

Attacker's goal: Most existing work on poisoning attacks assumes that the attacker's goal is to decrease the classifier's accuracy [6, 13]. A recent work allows the attacker to have an arbitrary objective model (a classifier) and the attacker's goal is to make the learner's learned model close to the objective model [38]. However, they restrict the attacker's objective model to be a linear classifier. We extend their setting to allow the attacker to have an arbitrary objective model, which is represented by a function $f^* : \mathbf{x} \to \{-1, +1\}$. We define two kinds of attacks based on the attacker's incentives in real world.

- Integrity attack. The attacker has some test instances $\{\mathbf{x}_i\}_{i=n+1}^m$ and wants the labels predicted by the victim model to be similar to that predicted by f^*. For example, a spammer may only want certain spam to be classified as regular ones. Note that $\{\mathbf{x}_i\}_{i=n+1}^m$ can be a mixture of instances that the attacker has preference on and those he is neutral about.

- Availability attack. The attacker wants to decrease the accuracy of the victim model. For example, an attacker may want to disturb a recommender system by decreasing

the accuracy of its built-in classification models.

Attacker's capability: In data poisoning attacks, an attacker who takes full control of training data can create an arbitrary victim model. However, in reality, the attacker usually faces some constraints. In this work, we assume that the attacker can flip at most B labels of the training set D. We denote by $D' = \{(\mathbf{x}_i, y'_i)\}_{i=1}^n$ the contaminated training set. We introduce a binary vector $\mathbf{z}$ and denote $y'_i = y_i(1 - 2z_i)$ so that $z_i = 1$ means that the label of sample i is flipped and $z_i = 0$ otherwise.

3.1.1 Attacking Linear Classifiers

In this work, we consider three linear classifiers: SVM, Logistic Regression (LR) and Least-squares SVM (LS-SVM), but note that our methods allow general loss functions as long as they are differentiable. The three classifiers can be obtained by replacing the loss function L in Eq.(1) with the following loss functions.

- Hinge loss (SVM): $L^1(y_i, f(\mathbf{x}_i)) = \max\{0, 1-y_i f(\mathbf{x}_i)\}$
- Logistic loss (LR): $L^2(y_i, f(\mathbf{x}_i)) = \log(1 + \exp(-y_i f(\mathbf{x}_i)))$
- Squared hinge loss (LS-SVM): $L^3(y_i, f(\mathbf{x}_i))=(1-y_i f(\mathbf{x}_i))^2$

For attacking linear classifiers, the attacker first reduces his objective model f^* to a weight vector $\mathbf{w}^* \in \mathbb{R}^{k+1}$. In other words, $\mathbf{w}^*$ can be viewed as a linear classifier that is the closest to f^*. Specifically, in the integrity attack, $\mathbf{w}^*$ can be learned from $D^a_{in}=\{(\mathbf{x}_i, y_i)|y_i=f^*(\mathbf{x}_i)\}_{i=n+1}^m$. In the availability attack, $\mathbf{w}^*$ can be learned from $D^a_{av}=\{(\mathbf{x}_i, -y_i)|(\mathbf{x}_i, y_i)\in D\}_{i=1}^n$. In both integrity attack and availability attack, the attacker wants the learner's learned weight vector $\mathbf{w}$ as close to $\mathbf{w}^*$ as possible. Since $\mathbf{w}$ and $\mathbf{w}^*$ can be viewed as two hyper-lines in a $k+1$-dimensional space, intuitively, the attacker's goal can be viewed as rotating $\mathbf{w}$ to $\mathbf{w}^*$. We assume that the attacker's goal is maximizing the cosine

of the angle between $\mathbf{w}$ and $\mathbf{w}^*$ and define the attacker's utility function as:

$$U(\mathbf{w}, \mathbf{w}^*) = \frac{\mathbf{w}^\intercal \mathbf{w}^*}{||\mathbf{w}||||\mathbf{w}^*||}.$$

We formulate the optimal attack problem as the following bilevel program.

$$\max_{\mathbf{z}} \quad \frac{\mathbf{w}^\intercal \mathbf{w}^*}{||\mathbf{w}||||\mathbf{w}^*||} \tag{3.2}$$

$$\text{s.t.} \quad f \in \arg\min_{g \in \mathcal{H}} \quad C \sum_{i=1}^{n} L(y_i', g(\mathbf{x}_i)) + \frac{1}{2}||g||^2 \tag{3.3}$$

$$\sum_{i=1}^{n} z_i \leq B \tag{3.4}$$

$$y_i' = y_i(1 - 2z_i), \forall i \in [n] \tag{3.5}$$

$$z_i \in \{0, 1\}, \forall i \in [n] \tag{3.6}$$

One can obtain the optimal attack problem on specific linear classifiers by replacing the loss function L in Eq.(3.1) with the associated loss functions. Eqs.(3.2) - (3.6) is a mixed-integer bilevel program, which is generally hard to solve. We will introduce the PGA algorithm to approximately solve this problem in Section 3.2.

3.1.2 Attacking Kernel SVMs

A kernel machine applies a feature mapping $\phi : \mathbb{R}^k \to \mathbb{R}^r$ on training data so that the data could be more separable in higher dimensional space (usually $r > k$). A kernel SVM can be viewed as a linear SVM in the transformed feature space. Since r can be arbitrarily large, instead of solving the primal problem Eq.(3.1), one usually solves its dual problem:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^\intercal Q \boldsymbol{\alpha} - \sum_{i=1}^{n} \alpha_i \tag{3.7}$$

$$0 \leq \alpha_i \leq C, \forall i \in [n] \tag{3.8}$$

where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^\intercal \phi(\mathbf{x}_j)$. In practice, $\phi(\mathbf{x}_i)^\intercal \phi(\mathbf{x}_j)$ is usually replaced by a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ to facilitate computation. We classify the kernel functions into two classes: one with finite feature mapping (e.g., polynomial kernels) and the other with infinite feature mapping (e.g., radial basis function kernels). We will introduce how to attack these kernel SVMs separately.

For kernel SVMs with finite feature mapping, the attacker first reduces his objective model f^* to a weight vector $\mathbf{w}^* \in \mathbb{R}^{r+1}$. Similar to the linear classification case, in the integrity attack, $\mathbf{w}^*$ can be learned from $D_{in}^a = \{(\phi(\mathbf{x}_i), y_i) | y_i = f^*(\mathbf{x}_i)\}_{i=n+1}^m$. In the availability attack, $\mathbf{w}^*$ can be learned from $D_{av}^a = \{(\phi(\mathbf{x}_i), -y_i) | (\mathbf{x}_i, y_i) \in D\}_{i=1}^n$. For kernel SVMs with infinite feature mapping, we use the technique of random Fourier features [63, 64] to construct an approximate finite feature mapping. The random Fourier features are constructed by first sampling random vectors $\omega_1, ..., \omega_q$ from $p(\omega)$, where $p(\omega)$ is the Fourier transform of kernel function K. Then, $\mathbf{x}_i$ is transformed to $\phi(\mathbf{x}_i)$ with new features

$$\phi(\mathbf{x}_i) = (sin(\omega_1^\intercal \mathbf{x}_i), cos(\omega_1^\intercal \mathbf{x}_i), ..., sin(\omega_q^\intercal \mathbf{x}_i), cos(\omega_q^\intercal \mathbf{x}_i)).$$

One can refer to [64] for detailed procedures. The random Fourier features ensures $\phi(\mathbf{x}_i)^\intercal \phi(\mathbf{x}_j) \approx K(\mathbf{x}_i, \mathbf{x}_j)$. Note that the dimension of ϕ is $2q$, where q is the number of random vectors drawn from $p(\omega)$. The attacker can construct his objective model $\mathbf{w}^* \in \mathbb{R}^{2q+1}$ similarly to the finite feature mapping case using the feature mapping ϕ.

In order to obtain the optimal attack problem on kernel SVMs, we need to replace the lower level problem Eqs.(3.3) - (3.6) with Eqs.(3.7) - (3.8) and add constraint Eq.(3.9) to the upper level problem, which is derived from the Representer Theorem [65].

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i' \phi(\mathbf{x}_i) \tag{3.9}$$

Algorithm 1: Projected Gradient Ascent (PGA)

Input: Original training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, attacker's objective model $\mathbf{w}^*$, budget B, step size η, iteration limit t^{max};
Choose a random $\mathbf{z}^0 \in [0,1]^n$;
$\mathbf{y}'^0 \leftarrow$ Flip($\mathbf{z}^0$);
Train a classifier using training data $\{(\mathbf{x}_i, \mathbf{y}'^0)\}$;
Initialize dual variables $\boldsymbol{\alpha}^0$ and primal variables $\mathbf{w}^0$;
$t \leftarrow 1$;
while *Not converge and* $t < t^{max}$ **do**
  $\mathbf{z}^t \leftarrow \text{Proj}(\mathbf{z}^{t-1} + \eta \nabla_{\mathbf{z}^{t-1}} U)$;
  $\mathbf{y}'^t \leftarrow$ Flip($\mathbf{z}^t$);
  Retrain the classifier using training data $\{(\mathbf{x}_i, \mathbf{y}'^t)\}$;
  Update $\boldsymbol{\alpha}^t, \mathbf{w}^t$;
  $t \leftarrow t + 1$;
end
Output: Contaminated labels $\mathbf{y}'^t$.

Algorithm 2: Flip strategy

Input: $\mathbf{z}$, original labels $\mathbf{y}$, budget B;
$\Gamma \leftarrow$ Indices of $Sort([z_1, z_2, ..., z_n], \text{`}descent\text{'})$;
$j \leftarrow 1$, $y'_i \leftarrow y_i, \forall i \in [n]$;
while $\sum_{i=1}^n z_{\Gamma(j)} \leq B$ **do**
  $y'_{\Gamma(j)} \leftarrow -y_{\Gamma(j)}$;
  $j \leftarrow j + 1$;
end
Output: Flipped labels $\mathbf{y}'$.

3.2 Computing Attacking Strategies

Inspired by [4, 36, 38], we develop the PGA algorithm 1 for computing approximate solutions of Eqs.(3.2) - (3.6) and show that PGA can also compute attack strategies on kernel SVMs. We first relax binary variables z_i to interval $[0, 1]$ and solve the relaxed problem. PGA works by gradually updating $\mathbf{z}^t$ along its approximate gradients until converge or the iteration limit is reached. Since $\mathbf{z}^t$ is a real number vector and retraining the classifier requires $\mathbf{y}'^t$ to be a binary vector, we construct a *flip strategy* to project $\mathbf{z}^t$ to $\mathbf{y}'^t$. The flip strategy is shown in Algorithm 2. At each iteration, the projector Proj($\mathbf{z}$) first projects $\mathbf{z}$ to

an ℓ_∞ norm ball by truncating each z_i into range $[0, 1]$. Then the projected point is further projected to an ℓ_1 norm ball with diameter B, which ensures that $\sum_{i=1}^{n} z_i \leq B$.

Steps 4 and 10 in PGA involve training process of the victim model. If the victim model is SVM, in step 4 and 10 we solve the dual SVM problem Eqs.(3.7) - (3.8). If the victim model is logistic regression, we solve the following dual logistic regression problem:

$$\max_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^\intercal Q\boldsymbol{\alpha} + \sum_{i:\alpha_i>0} \alpha_i \log \alpha_i + \sum_{i:\alpha_i<C} (C-\alpha_i)\log(C-\alpha_i) \tag{3.10}$$

$$0 \leq \alpha_i \leq C, \forall i \in [n] \tag{3.11}$$

where $Q_{ij} = y'_i y'_j \mathbf{x}_i{}^\intercal \mathbf{x}_j$. If the victim model is least-squares SVM, we solve the dual least-squares SVM problem:

$$(Q + C^{-1} I_n)\boldsymbol{\alpha} = \mathbf{1}_n \tag{3.12}$$

where I_n is the $n \times n$ identical matrix and $\mathbf{1}_n$ is the n-dimensional vector of element 1. If the victim learning model is kernel SVM, we solve Eqs.(3.7) - (3.8) with $Q_{ij} = y'_i y'_j \phi(\mathbf{x}_i)^\intercal \phi(\mathbf{x}_j)$. In step 5 (step 11) of PGA, $\boldsymbol{\alpha}^0$ ($\boldsymbol{\alpha}^t$) is the solution of the problem solved in step 4 (step 10) and $\mathbf{w}^0$ ($\mathbf{w}^t$) is computed using Eq.(9).

In order to compute the gradient $\nabla_{\mathbf{z}} U$ in step 8 (refer to Section 3.1.1 for the definition of U), we first apply chain rule to arrive at:

$$\nabla_{\mathbf{z}} U = \nabla_{\mathbf{w}} U \cdot \nabla_{\mathbf{y}'} \mathbf{w} \cdot \nabla_{\mathbf{z}} \mathbf{y}' \tag{3.13}$$

The first and the third gradient can be easily computed as:

$$\frac{\partial U}{\partial w_j} = \frac{||\mathbf{w}||^2 w_j^* - \mathbf{w}^\intercal \mathbf{w}^* w_j}{||\mathbf{w}||^3 ||\mathbf{w}^*||} \tag{3.14}$$

$$\frac{\partial y_i'}{\partial z_j} = -\mathbb{1}(i=j) 2 y_i \tag{3.15}$$

where $\mathbb{1}(\cdot)$ is the indicator function. The second gradient $\nabla_{\mathbf{y}'}\mathbf{w}$ is hard to compute since it involves an optimization procedure. We leverage Eq.(3.9) to approximately compute the second gradient. If the victim learning model is linear, Eq.(3.9) is modified as Eq.(3.16).

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i' \mathbf{x}_i. \tag{3.16}$$

Taking the derivatives of both sides we have:

$$\frac{\partial w_j}{\partial y_i'} = \alpha_i x_{ij} \tag{3.17}$$

If the victim learning model is a kernel SVM, we can take derivatives of both sides of Eq.(3.9) and obtain:

$$\frac{\partial w_j}{\partial y_i'} = \alpha_i \phi(x_{ij}) \tag{3.18}$$

3.3 Attacking Black-Box Victim Models Using Substitutes

In previous sections we introduced how to compute attacks against a broad family of learning models. However, the attacker may not have full knowledge of the victim learning model in many real-world scenarios. Observing that an attack targets on one learning model can also

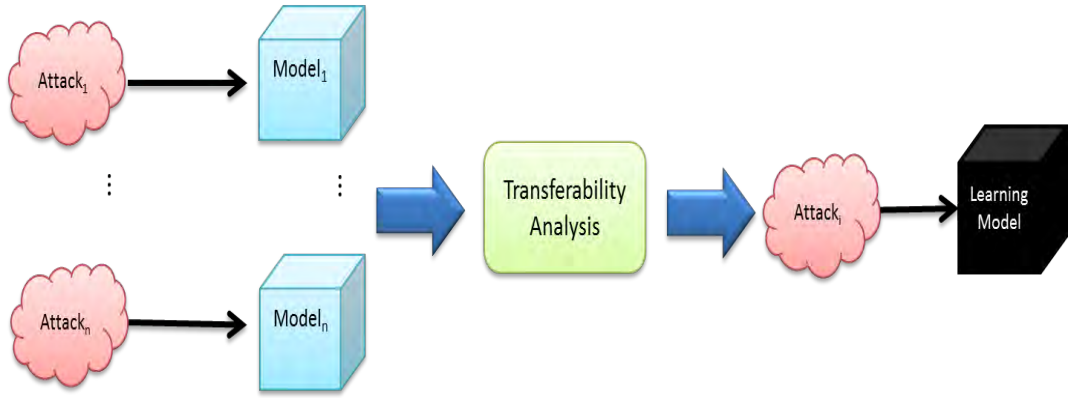


Figure 3.1: Decision boundaries (solid lines) of learned models under attacks with different attacker budgets.

be effective on another learning model even if the two models have different architectures, the attacker can design attack against a substitute model and then perform this attack on the victim learning model. Figure 3.1 shows the general paradigm of black-box attacks. A good substitute model is such that the attack against it is also effective on a general family of learning models.

The effectiveness of substitute-based attacks on a victim model can be evaluated by the victim model's accuracy on a test set D_{test}. In the integrity attack, D_{test} can be $D_{in}^a = \{(\mathbf{x}_i, y_i)|y_i{=}f^*(\mathbf{x}_i)\}_{i=n+1}^m$ and in the availability attack D_{test} can be $D_{av}^a = \{(\mathbf{x}_i, -y_i)|(\mathbf{x}_i, y_i){\in}D\}_{i=1}^n$. As discussed in Section 3.1, the attacker aims to increase the classification accuracy of the victim model on D_{test} because the attacker's objective model $\mathbf{w}^*$ is learned from D_{in}^a and D_{av}^a, with respect to integrity attack and availability attack. We denote by $M = \{M_1, M_2, ..., M_{|M|}\}$ the set of learning models and by τ_i the attack against model M_i. We denote by $M_j^{\tau_i}$ the victim model M_j learned under attack τ_i. Then the effectiveness of the attack against substitute M_i on victim model M_j can be evaluated by $accuracy(M_j^{\tau_i}, D_{test})$. We will evaluate the effectiveness of five substitute models on eight victim models in Section 3.4.3.

3.4 Experimental Evaluation

In this section, we evaluate the proposed attack and defense algorithms and analyze the transferability of the attacks. We compute attacks against three linear learning models: SVM, logistic regression (LR), least-squares SVM (LS-SVM) and two nonlinear models: SVM with polynomial kernel (POLY) and radial basis function kernel (RBF). We will use five public data sets: Australian (690 points, 14 features), W8a (10000 points, 300 features), Spambase (4601 points, 57 features) [66], Wine (130 points, 14 features) and Skin (5000 points, 3 features) [1]. All training processes are implemented with LIBSVM [67] and LINLINEAR [68]. All attacks computed by PGA are the best among 50 runs.

3.4.1 Integrity Attacks Visualization

We visualize the integrity attacks against SVM, LR, LS-SVM, POLY, RBF computed by PGA. We set the regularization parameter C=1 for all five models. We set the parameters d=2 for polynomial kernel and γ=0.1 for RBF kernel. The training set is a 2-D artificial data set containing 100 points. We ignore the process of generating the attacker's objective model and set it as an arbitrary one. Figure 3.2 shows how the attacks under different attacker budgets can affect the decision boundaries of victim models. The dashed black lines represent the attacker's objective model. The attacker wants the points on the left side of this line to be classified as "red" and the points on the right side to be classified as "blue". The bigger red (blue) points are originally blue (red) and are flipped by the attacker. We can see that the victim learning models can be converted to models that are very close to the attacker's objective model under only 20 flips. In addition, the attacked points with respect to different victim models are highly similar, which indicates that the attacks have transferability.

[1] Except Spambase, all data sets can be downloaded from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

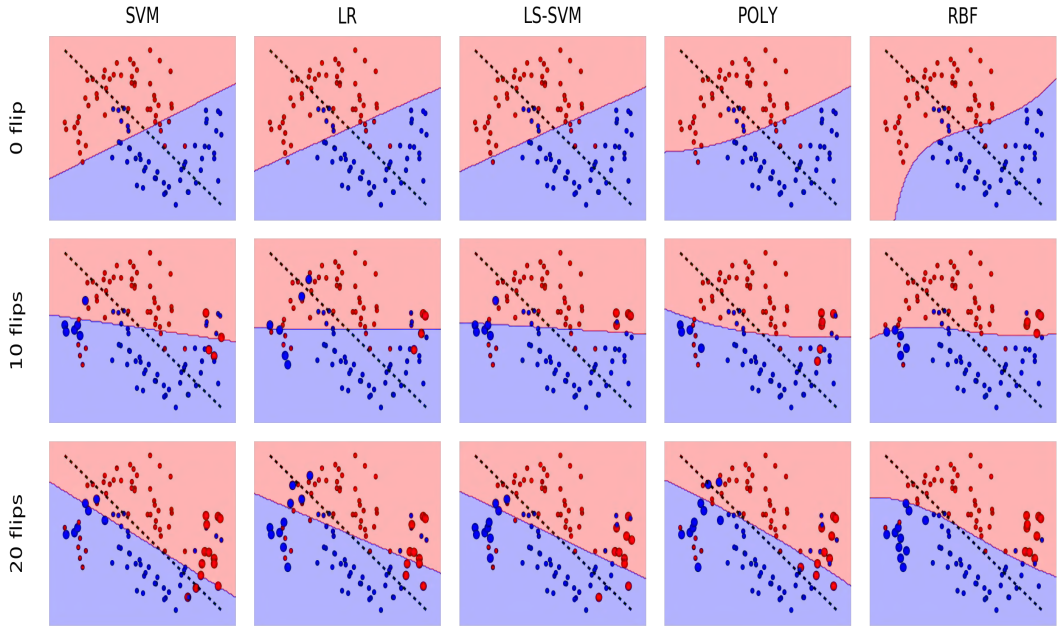


Figure 3.2: Decision boundaries (solid lines) of learned models under attacks with different attacker budgets.

3.4.2 Solution Quality Comparison

We compute availability attacks against SVM using PGA and compare our solution with two baselines. The first baseline is a random flip strategy, where the attacker randomly flips the labels of training data under his budget. For each data set and budget, we compute the random attack for 50 times and report the best out of them. The second baseline, Adversarial Label Flip Attack on SVMs (ALFA) [6], is an existing algorithm that can compute attacks that decrease the accuracy of SVMs. ALFA works by iteratively solving a quadratic and a linear problem until convergence. Figure 3.3 shows that the attacks computed by PGA significantly outperform both baselines. X-axis is the percentage of flipped points and y-axis is the accuracy of victim model on training set. On the W8a data set, the attacks computed by PGA decrease the victim model's accuracy from 90% to 30% with only 30% flips. We also find that PGA scales significantly better than ALFA. Because in each iteration of ALFA it solves two optimization problems and their sizes grow with the number of data points, while

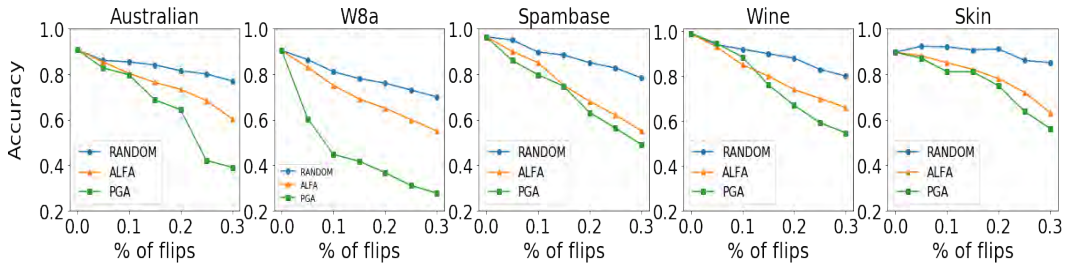


Figure 3.3: Accuracy of victim model under different attacker budgets.

in each iteration of PGA it trains a linear classifier, which can be efficiently implemented with LIBLINEAR.

3.4.3 Transferability Analysis

We compute availability attacks against the aforementioned five substitute models using PGA and test the accuracy of eight victim models under the attacks. The victim models include the five substitute models and decision tree (DT), k-nearest neighbors (KNN) and Naive Bayes (NB). The DT, KNN and NB models are trained using MATLAB R2016b Statics and Machine Learning Toolbox and all parameters are set by default. We set the attacker's budget as 30% of the training points.

Table 3.1 shows the influence of the five attacks on the eight victim models. First, we can see from the diagonal values that if the substitute model and the victim model are of the same type, the attack can significantly degrade the accuracy of the victim model. Second, the performance of an attack designed for a linear model on another linear victim model are comparable with the attack designed for the victim model, which means that the attack designed for a linear model has a good transferability when victim models are also linear. Third, the attack designed for a linear model has a good transferability when the victim model is nonlinear. However, the attack designed for nonlinear models has a bad transferability when the victim models are linear. For example, on the Skin dataset, the attack designed

for RBF can degrade the accuracy of an RBF model to 0.38. However, an SVM victim model under this attack can still achieve 0.94 accuracy, which means that the attack barely has influence on the SVM model. Fourth, on the Australian and the Skin dataset, the attacks designed for the five substitute models have similar transferability when the victim models are DT, KNN and NB. However on the Spambase dataset, the attacks designed for linear models have significantly better transferability than those designed for nonlinear models. In conclusion, attacks against linear models generally have a good transferability than that against nonlinear models.

3.5 Chapter Summary

This work studies label contamination attacks against classification models. We first focused on the problem of optimal label contamination attack against a family of empirical risk minimization models. We formulated each optimal attack problem as a mixed integer bilevel program and developed the PGA algorithm to compute the near-optimal attacks. Then, we considered a more realistic scenario where the victim model are a black-box to the attacker. In such a scenario, we proposed a substitute-based attacking strategy for the attacker. In the experimental part, we studied the transferability of the label contamination attacks and demonstrated that the substitute-based attacks can be very effective against black-box learning models when appropriate substitute model is chosen. We also discussed about possible defenses to mitigate data poisoning attacks.

	SVM	LR	LS-SVM	POLY	RBF	DT	KNN	NB
SVM	0.40	0.55	0.42	0.63	0.49	0.68	0.70	0.39
LR	0.55	0.53	0.48	0.59	0.49	0.69	0.70	0.33
LS-SVM	0.53	0.54	0.25	0.63	0.64	0.66	0.70	0.33
POLY	0.67	0.68	0.55	0.53	0.52	0.62	0.70	0.43
RBF	0.82	0.78	0.69	0.67	0.55	0.64	0.70	0.48

(a) Australian dataset.

	SVM	LR	LS-SVM	POLY	RBF	DT	KNN	NB
SVM	0.45	0.47	0.48	0.62	0.55	0.68	0.70	0.35
LR	0.54	0.48	0.48	0.63	0.67	0.69	0.70	0.33
LS-SVM	0.53	0.50	0.50	0.63	0.66	0.68	0.69	0.36
POLY	0.74	0.73	0.74	0.76	0.74	0.70	0.70	0.61
RBF	0.83	0.81	0.82	0.84	0.54	0.71	0.71	0.78

(b) Spambase dataset.

	SVM	LR	LS-SVM	POLY	RBF	DT	KNN	NB
SVM	0.56	0.59	0.58	0.59	0.58	0.71	0.70	0.56
LR	0.57	0.59	0.57	0.61	0.78	0.70	0.70	0.56
LS-SVM	0.46	0.52	0.51	0.50	0.46	0.67	0.69	0.46
POLY	0.90	0.60	0.69	0.52	0.77	0.69	0.75	0.45
RBF	0.94	0.90	0.88	0.91	0.38	0.76	0.69	0.58

(c) Skin dataset.

Table 3.1: Accuracy of victim models under substitute-based attacks.

Chapter 4

Data poisoning attacks on multi-task relationship learning

This chapter studies data poisoning attacks, in particular data injection attacks on multi-task learning (MTL) models. MTL is a machine learning paradigm that improves the performance of each task by exploiting useful information contained in multiple related tasks. However, the relatedness of tasks can be exploited by attackers to launch data poisoning attacks, which has been demonstrated to be a big threat to single-task learning. In this work, we provide the first study on the vulnerability of MTL. Specifically, we focus on multi-task relationship learning (MTRL) models, a popular subclass of MTL models where task relationships are quantized and are learned directly from training data. We formulate the problem of computing optimal poisoning attacks on MTRL as a bilevel program that is adaptive to arbitrary choice of *target* tasks and *attacking* tasks. We propose an efficient algorithm called PATOM for computing optimal attack strategies. PATOM leverages the optimality conditions of the subproblem of MTRL to compute the implicit gradients of the upper level objective function. Experimental results on real-world datasets show that MTRL models are very sensitive to poisoning attacks and the attacker can significantly degrade the performance of target tasks,

by either directly poisoning the target tasks or indirectly poisoning the related tasks exploiting the task relatedness. We also found that the tasks being attacked are always strongly correlated, which provides a clue for defending against such attacks.

4.1 Multi-Task Relationship Learning

We denote by $T = \{T_i\}_{i=1}^m$ the set of learning tasks. For each task T_i, we are given a set of training data $D_i = \{(\mathbf{x}_j^i, y_j^i)|\mathbf{x}_j^i \in \mathbb{R}^d, j = 1, ..., n_i\}$. The label $y_j^i \in \mathbb{R}$ if the task is a regression task and $y_j^i \in \{-1, +1\}$ if the task is a binary classification task. Note that a multi-class classification problem can be easily decomposed to a set of binary classification problems using the one-vs-the-rest strategy [68]. The goal of MTL is to jointly learn a prediction function $f_i(\mathbf{x})$ for each task. In this work, we consider linear prediction functions where $f_i(\mathbf{x}) = (\mathbf{w}^i)^\top \mathbf{x} + b_i$, but note that it is easy to extend to non-linear cases using kernel methods. For the ease of representation, we denote $(\mathbf{x}, 1)$ by $\mathbf{x}$ and denote $(\mathbf{w}, b)$ by $\mathbf{w}$ so that $f_i(\mathbf{x}) = (\mathbf{w}^i)^\top \mathbf{x}$.

We consider a general multi-task relationship learning (MTRL) formulation [11] as follows, which includes many existing popular MTL methods as its special cases [69–72].

$$\min_{\mathbf{W},\mathbf{\Omega}} \quad \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} l((\mathbf{w}^i)^\top \mathbf{x}_j^i, y_j^i) + \frac{\lambda_1}{2} \text{tr}(\mathbf{W}\mathbf{W}^\top) + \frac{\lambda_2}{2} \text{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top), \tag{4.1}$$

$$\text{s.t.} \quad \mathbf{\Omega} \succeq 0, \text{tr}(\mathbf{\Omega}) = 1, \tag{4.2}$$

where $l(\cdot)$ is an arbitrary convex loss function, $\mathbf{W}$ is a matrix whose i-th column $\mathbf{w}^i$ is the weight vector of task T_i, $\mathbf{\Omega} \in \mathbb{R}^{m \times m}$ is the covariance matrix that describes positive,

negative and unrelated task relationships. The first term in the objective function measures the empirical loss of all tasks with the term $1/n_i$ to balance the different sample sizes of tasks. The second term in the objective function is to penalize the complexity of $\mathbf{W}$, and the last term serves as the task-relationship regularization term. The first constraint ensures that the covariance matrix $\mathbf{\Omega}$ is positive semi-definite, and the second constraint controls its complexity.

4.2 Data Poisoning Attacks on MTRL

In this section, we introduce the problem settings for the data poisoning attack on MTRL. We define three kinds of attacks based on real-world scenarios and propose a bilevel formulation for computing optimal attacks.

We assume that the attacker aims to degrade the performance of a set of *target* tasks $T_{tar} \subset T$ by injecting data to a set of *attacking* tasks $T_{att} \subset T$. We denote by $\widehat{D}_i = \{(\widehat{\mathbf{x}}_j^i, \widehat{y}_j^i) | \widehat{\mathbf{x}}_j^i \in \mathbb{R}^d, j = 1, ..., \widehat{n}_i\}$ the set of malicious data injected to task i. Specially, $\widehat{D}_i = \varnothing$, i.e., $\widehat{n}_i = 0$, if $T_i \notin T_{att}$. We define and study the following three kinds of attacks based on real-world scenarios.

- **Direct attack**: $T_{tar} = T_{att}$. Attacker can directly inject data to all the target tasks. For example, in product review sentiment analysis, each task is a sentiment classification task that classifies a review as negative or positive. On e-commerce platforms such as Amazon, attackers can directly attack the target tasks by providing crafted reviews to the target products.

- **Indirect attack**: $T_{tar} \cap T_{att} = \varnothing$. Attacker cannot inject data to any of the target tasks. However, he can inject data to other tasks and indirectly influence the target tasks. For example, personalized recommendations treat each user as a task and use

users' feedback to train personalized recommendation models. In such scenarios, attackers usually cannot access the training data of target tasks. However, attackers can launch indirect attacks by faking some malicious user accounts, which will be treated as attacking tasks, and providing crafted feedback to the systems.

- **Hybrid attack**: A mixture of direct attack and indirect attack where the attacker can inject data to both target tasks and attacking tasks.

We denote by $\mathcal{L}(D, \mathbf{w}) = \sum_{k=1}^{|D|} l(\mathbf{w}^\top \mathbf{x}_k, y_k)$ the empirical loss incurred by weight vector $\mathbf{w}$ on data set D, and define the attacker's utility function as the empirical loss on training data of the target tasks:

$$\mathcal{U} = \sum_{\{i|T_i \in T_{tar}\}} \mathcal{L}(D_i, \mathbf{w}^i).$$

Following the Kerckhoffs' principle [73] and existing works on poisoning attacks [4, 13], we assume that the attacker has full knowledge of the victim MTRL model. In reality, attackers can either obtain the knowledge of victim models by exploiting insider threats [74] or probing the machine learning systems by sending queries from the outside [75]. We then formulate the optimal attack problem as the following bilevel optimization problem. Problem (4.3) is the upper level problem, in which the objective function is the attacker's utility $\mathcal{U}$. The variables of the upper level problem are the injected data points $\widehat{D}_i$, which are usually constrained in real-world scenarios. For example, the injected data should have similar scale with the clean data. Problem (4.4) is the lower level problem, which is an MTRL problem with training set consists of both clean and injected data points. The lower level problem can be regarded as the constraint of the upper level problem. In other words, the variables $\mathbf{W}$ used for computing the upper level objective $\mathcal{U}$ is kept to be the optimal solution of the lower level problem.

$$\max_{\{\widehat{D}_i|T_i\in T_{att}\}} \sum_{\{i|T_i\in T_{tar}\}} \mathcal{L}(D_i, \mathbf{w}^i), \tag{4.3}$$

$$\text{s.t.} \quad \text{Constraints on } \{\widehat{D}_i|T_i \in T_{att}\},$$

$$\min_{\mathbf{W},\mathbf{\Omega}} \quad \sum_{i'=1}^{m} \frac{1}{n_{i'}+\widehat{n}_{i'}} \mathcal{L}(D_{i'} \cup \widehat{D}_{i'}, \mathbf{w}^{i'}) + \frac{\lambda_1}{2}\text{tr}(\mathbf{W}\mathbf{W}^\top) + \frac{\lambda_2}{2}\text{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top), \tag{4.4}$$

$$\text{s.t.} \quad \mathbf{\Omega} \succeq 0, \text{tr}(\mathbf{\Omega}) = 1. \tag{4.5}$$

4.3 Computing Optimal Attack Strategies

In this section, we propose an algorithm called PATOM for computing optimal attack strategies. PATOM is a projected stochastic gradient ascent based algorithm that efficiently maximizes the injected data in the direction of increasing the empirical loss of target tasks. Since there is no close-form relation between the empirical loss and the injected data, we compute the gradients exploiting the optimality conditions of the subproblem of MTRL.

4.3.1 General Optimization Framework

Bilevel problems are usually hard to solve due to their non-linearity, non-differentiability and non-convexity. In our bilevel formulation, although the upper level problem (4.3) is relatively simple, the lower level problem (4.4) is highly non-linear and non-convex. Inspired by [4, 23, 36, 38], we use a projected gradient ascent method to solve our proposed bilevel problem. The idea is to iteratively update the injected data in the direction of maximizing the attacker's utility function $\mathcal{U}$. In order to reduce the complexity of the optimal attack problem,

we fix the labels of injected data $\widehat{y}_j^i$ and optimize over the features of injected data $\widehat{\mathbf{x}}_j^i$. The update rule is written as follows,

$$(\widehat{\mathbf{x}}_j^i)^t \leftarrow \text{Proj}_{\mathbb{X}}((\widehat{\mathbf{x}}_j^i)^{t-1} + \eta \nabla_{(\widehat{\mathbf{x}}_j^i)^{t-1}} \mathcal{U}), \forall i, j, \tag{4.6}$$

where η is the step size, t denotes the t-th iteration, and $\mathbb{X}$ represents the feasible region of the injected data, which is specified by the first constraint in the upper level problem (4.3). We consider $\mathbb{X}$ as an ℓ_2-norm ball with diameter r. Therefore, $\text{Proj}_{\mathbb{X}}$ can be represented by:

$$\text{Proj}_{\mathbb{X}}(\mathbf{x}) = \begin{cases} \mathbf{x}, & \text{if } ||\mathbf{x}||_2 \leq r, \\ \frac{\mathbf{x}r}{||\mathbf{x}||_2}, & \text{if } ||\mathbf{x}||_2 > r. \end{cases}$$

In order to compute the gradients $\nabla_{(\widehat{\mathbf{x}}_j^i)^{t-1}} \mathcal{U}$, we first apply the chain rule to arrive at

$$\nabla_{(\widehat{\mathbf{x}}_j^i)} \mathcal{U} = \nabla_W \mathcal{U} \cdot \nabla_{(\widehat{\mathbf{x}}_j^i)} \mathbf{W}. \tag{4.7}$$

However, note that $\mathcal{U}$ is the sum of losses incurred by every point in the target tasks, the first term on the right side could be computationally expensive if the number of data points in target tasks is large. Therefore, we instead propose a projected stochastic gradient ascent based algorithm, called PATOM, to improve the scalability of our approach.

The details of PATOM is shown in Algorithm 3. We first randomly initialize the injected data $\widehat{D}_i$ within the ℓ_2-norm ball with diameter r. Using the injected data, we solve the MTRL problem (the lower level problem (4.4)), and obtain the initial values of the weight matrix $\mathbf{W}_0$ and the covariance matrix $\mathbf{\Omega}_0$. In each iteration, we perform a projected stochastic gradient ascent procedure steps (7-10) on all the injected data. Specifically, for each data point $(\mathbf{x}_q^p, y_q^p)$ sampled from D_{batch}, we compute the gradients of its associate loss $l((\mathbf{w}_t^p)^\top \mathbf{x}_q^p, y_q^p)$ with respect to each injected data $\widehat{\mathbf{x}}_j^i$. Therefore, by replacing $\mathcal{U}$ in (4.6) with $l((\mathbf{w}_t^p)^\top \mathbf{x}_q^p, y_q^p)$ we have the stochastic version of the update rule as shown in (4.8). Then, with the updated

injected data $\widehat{D}_i = \widehat{D}_i^t$, we solve the lower level problem (4.4) again to obtain a new weight matrix $\mathbf{W}_t$ and a new covariance matrix $\mathbf{\Omega}_t$, which will be used in the next iteration.

$$(\widehat{\mathbf{x}}_j^i)^t \leftarrow \text{Proj}_{\mathbb{X}}((\widehat{\mathbf{x}}_j^i)^{t-1} + \eta \nabla_{(\widehat{\mathbf{x}}_j^i)^{t-1}} l((\mathbf{w}_{t-1}^p)^\top \mathbf{x}_q^p, y_q^p)), \forall i, j. \tag{4.8}$$

Algorithm 3: computing Poisoning ATtacks On Multi-task relationship learning (PATOM)

Input: T_{tar}, T_{att}, step size η, attacker budget $\widehat{n}_i$.
Randomly initialize $\widehat{D}_i^0 = \{((\widehat{\mathbf{x}}_j^i)^0, (\widehat{y}_j^i)^0) | j = 1, ..., \widehat{n}_i\}, \forall i \in T_{att}$.
$\widehat{D}_i = \widehat{D}_i^0, , \forall i \in T_{att}$.
Solve lower level problem (**??**) to obtain $\mathbf{W}_0$ and $\mathbf{\Omega}_0$.
$t \leftarrow 1$.
while $t < t^{max}$ **do**
    Sample a batch D_{batch} from $\cup_{i \in T_{tar}} D_i$.
    for $(\mathbf{x}_q^p, y_q^p) \in D_{batch}$ **do**
        for $i \in T_{att}, j = 1...\widehat{n}_i$ **do**
            Update $(\widehat{\mathbf{x}}_j^i)^t$ according to (4.8).
        end
    end
    $\widehat{D}_i = \widehat{D}_i^t, \forall i \in T_{att}$.
    Solve (**??**) to obtain $\mathbf{W}_t$ and $\mathbf{\Omega}_t$.
    $t \leftarrow t + 1$.
end

4.3.2 Gradients Computation

In order to compute the gradients $\nabla_{(\widehat{\mathbf{x}}_j^i)} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p)$ in (4.8), we still apply the chain rule and obtain:

$$\nabla_{\widehat{\mathbf{x}}_j^i} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) = \nabla_{\mathbf{w}^p} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) \cdot \nabla_{\widehat{\mathbf{x}}_j^i} \mathbf{w}^p. \tag{4.9}$$

We can see that the first term on the right side depends only on the loss function $l(\cdot)$ and is relatively easy to compute. However, the second term on the right side depends on the optimality conditions of lower level problem (4.4). In the rest of this section, we show how to compute the gradients with respect to two commonly used loss functions. For regression tasks, we adopt least-square loss: $l_1(\mathbf{w}^\top \mathbf{x}, y) = (y - \mathbf{w}^\top \mathbf{x})^2$. For classification tasks, we adopt squared hinge loss: $l_2(\mathbf{w}^\top \mathbf{x}, y) = (1 - y\mathbf{w}^\top \mathbf{x})^2$.

We first fix $\mathbf{\Omega}$ to eliminate the constraints of the lower level problem (4.4), and obtain the following sub-problem:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \sum_{i=1}^{m} \frac{1}{n_i + \widehat{n}_i} \mathcal{L}(D_i \cup \widehat{D}_i, \mathbf{w}^i) + \frac{\lambda_1}{2} \mathrm{tr}(\mathbf{W}\mathbf{W}^\top) \\ & + \frac{\lambda_2}{2} \mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top). \end{aligned} \tag{4.10}$$

As shown in [11], MTRL problems can be solved by an alternating approach with $\mathbf{\Omega}$ and $\mathbf{W}$ alternatingly fixed in each iteration. Also note that in bilevel optimization, the optimality of the lower level problem can be considered as a constraint to the upper level problem. Therefore, at convergence, we can treat $\mathbf{\Omega}$ in Problem (4.10) as a constant-value matrix when computing the gradients. We then substitute the least-square loss function $l_1(\cdot)$ into Problem (4.10) and reformulate it as the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \sum_{i=1}^{m} \frac{1}{n_i + \widehat{n}_i} \left(\sum_{j=1}^{n_i} (\varepsilon_j^i)^2 + \sum_{j'=1}^{\widehat{n}_i} (\hat{\varepsilon}_j^i)^2 \right) + \frac{\lambda_1}{2} \mathrm{tr}(\mathbf{W}\mathbf{W}^\top) \\ & + \frac{\lambda_2}{2} \mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^\top), \\ \text{s.t.} \quad & \varepsilon_j^i = y_j^i - (\mathbf{w}^i)^\top \mathbf{x}_j^i, \quad \forall i, j, \\ & \hat{\varepsilon}_{j'}^i = \widehat{y}_{j'}^i - (\mathbf{w}^i)^\top \widehat{\mathbf{x}}_{j'}^i, \quad \forall i, j'. \end{aligned} \tag{4.11}$$

The Lagrangian of the problem (4.11) is:

$$\begin{aligned}\mathcal{G} =& \sum_{i=1}^{m} \frac{1}{n_i+\widehat{n}_i} \left(\sum_{j=1}^{n_i} (\varepsilon_j^i)^2 + \sum_{j=1}^{\widehat{n}_i} (\hat{\varepsilon}_j^i)^2 \right) + \frac{\lambda_1}{2} \mathrm{tr}(\mathbf{W}\mathbf{W}^\top) \\ &+ \frac{\lambda_2}{2} \mathrm{tr}(\mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^\top) \\ &+ \sum_{i=1}^{m} \left(\sum_{j=1}^{n_i} \alpha_j^i \left(y_j^i - (\mathbf{w}^i)^\top \mathbf{x}_j^i - \varepsilon_j^i \right) \right. \\ &\qquad \left. + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i (\widehat{y}_{j'}^i - (\mathbf{w}^i)^\top \widehat{\mathbf{x}}_{j'}^i - \hat{\varepsilon}_{j'}^i) \right). \end{aligned} \tag{4.12}$$

The gradient of $\mathcal{G}$ with respect to $\mathbf{W}$ is:

$$\begin{aligned}\frac{\partial G}{\partial \mathbf{W}} =& \mathbf{W}(\lambda_1 \mathbf{I}_m + \lambda_2 \boldsymbol{\Omega}^{-1}) \\ &- \sum_{i=1}^{m} \left(\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i \mathbf{e}_i^\top + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \mathbf{e}_i^\top \right). \end{aligned} \tag{4.13}$$

where $\mathbf{I}_m$ is $m \times m$ identity matrix, and $\mathbf{e}_i$ is the i-th column of $\mathbf{I}_m$. By setting $\frac{\partial G}{\partial \mathbf{W}} = \mathbf{0}$, we obtain:

$$\mathbf{W} = \sum_{i=1}^{m} \left(\left(\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \right) \mathbf{e}_i^\top \boldsymbol{\Omega} (\lambda_1 \boldsymbol{\Omega} + \lambda_2 \mathbf{I}_n)^{-1} \right), \tag{4.14}$$

which implies that each task's weight vector $\mathbf{w}^i$ can be represented as a linear combination of training data from all tasks. For simplicity in presentation, we denote by $\boldsymbol{\Phi} = \boldsymbol{\Omega}(\lambda_1 \boldsymbol{\Omega} + \lambda_2 \mathbf{I}_n)^{-1}$, and reexpress (4.14) as the following form:

$$\mathbf{w}^p = \sum_{i=1}^{m} \Phi_{i,p} \left(\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \right), p = 1...m. \tag{4.15}$$

Similarly, we substitute the squared hinge loss into the loss function $\mathcal{L}(\cdot)$ in Problem (4.10),

and obtain:

$$\mathbf{w}^p = \sum_{i=1}^{m} \Phi_{i,p} \left(\sum_{j=1}^{n_i} \alpha_j^i y_j^i \mathbf{x}_j^i + \sum_{j'=1}^{\widehat{n}_i} \widehat{\alpha}_{j'}^i \widehat{y}_{j'}^i \widehat{\mathbf{x}}_{j'}^i \right), p = 1...m. \tag{4.16}$$

Given (4.15) and (4.16), we can compute the gradient in (4.8). In case of the least-square loss, we have:

$$\begin{aligned} \nabla_{\widehat{\mathbf{x}}_j^i} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) &= 2((\mathbf{w}^p)^\top \mathbf{x}_q^p - y_q^p) \mathbf{x}_q^p \frac{\partial \mathbf{w}^p}{\partial \widehat{\mathbf{x}}_j^i} \\ &= 2((\mathbf{w}^p)^\top \mathbf{x}_q^p - y_q^p) \mathbf{x}_q^p \widehat{\alpha}_j^i \Phi_{i,p}. \end{aligned} \tag{4.17}$$

In case of the squared hinge loss, we have:

$$\begin{aligned} &\nabla_{\widehat{\mathbf{x}}_j^i} l((\mathbf{w}^p)^\top \mathbf{x}_q^p, y_q^p) \\ &= 2(y_q^p (\mathbf{w}^p)^\top \mathbf{x}_q^p - 1) y_q^p \mathbf{x}_q^p \frac{\partial \mathbf{w}^p}{\partial \widehat{\mathbf{x}}_j^i} \\ &= 2(y_q^p (\mathbf{w}^p)^\top \mathbf{x}_q^p - 1) y_q^p \mathbf{x}_q^p \widehat{y}_j^i \widehat{\alpha}_j^i \Phi_{i,p}. \end{aligned} \tag{4.18}$$

4.4 Experimental Results

In this section, we first evaluate PATOM in terms of convergence and solution quality. Experimental results show that PATOM converges to local optima in less than 10 iterations and the attack strategies computed by PATOM significantly outperform baselines. Second, we study the task relationships under the data poisoning attacks and found that task relationships are very sensitive to the attacks. We also found that the tasks under attacking form strong correlations.

4.4.1 Datasets

We use three real-world datasets to validate our proposed methods. The Landmine and the MNIST datasets are used for classification tasks and Sarcos dataset is used for regression tasks. For each dataset, all data points are divided by the maximum ℓ_2 norm among them, so that all data points are within a ℓ_2-norm ball with diameter 1. We consider this ball as the feasible region of the injected data in order to ensure that the injected data and the clean data are at the same scale. We use the area under the ROC curve (AUC) to evaluate the learning performance for classification tasks, and the normalized mean squared error (NMSE) for regression tasks. The higher AUC corresponds to the better performance for classification and the lower NMSE corresponds to the better performance for regression. The detailed description of the datasets are given below.

- **Sarcos**[1] relates to an inverse dynamics problem for a 7 degrees-of-freedom SARCOS anthropomorphic robot arm. The input is a 21-dimensional space that includes 7 joint positions, 7 joint velocities and 7 joint accelerations. Each input instance is associated with 7 joint torques. Following previous work [11], each task is to learn a mapping from the 21-dimensional input space to one of the 7 torques. The dataset contains 44,484 training examples and 4,449 test examples.

- **Landmine**[2] consists of 29 tasks collected from various landmine fields. A data point in each task is represented by a 9-dimensional feature vector, and associated with a corresponding binary label ("1" for landmine and "-1" for cluster). The feature vectors are extracted from radar images, concatenating four momentbased features, three correlation-based features, one energy ratio feature and one spatial variance feature. The tasks entail different numbers of data points, varying from 89 to 138 examples.

[1] `http://www.gaussianprocess.org/gpml/data/`.
[2] `http://people.ee.duke.edu/~lcarin/LandmineData.zip`.

- **MNIST**[3] is a hand-written digit dataset with 10 classes. We use the one-vs-the-rest strategy to decompose the multi-class classification problem to 10 binary classification problems, and treat each binary classification problem as a task. To form the training set for each task, we randomly draw 300 data points of the designated digits and assign label "+1" and draw an equal number of instances from other classes randomly and assign label "-1". The dataset contains 60,000 training examples and 10,000 test examples. We use principal component analysis (PCA) to reduce the feature space to a 128-dimensional space.

4.4.2 Evaluating Convergence of PATOM

Our first set of experiments study the convergence of PATOM on Sarcos dataset and Landmine dataset, with respect to regression tasks and classification tasks. On Sarcos dataset, we randomly draw 300 training examples and 600 test examples from the associated training and test set. For direct attacks, we select 3 tasks on Sarcos dataset and 15 tasks on Landmine dataset as the respective target tasks, and set the attacking tasks the same as the target tasks. For indirect attacks, we use the same target tasks as in the direct attack, and treat the rest of tasks as the attacking tasks. For hybrid attacks, we randomly select the same number of attacking tasks as in the indirect attack experiments from all tasks. We set the step size $\eta = 100$ and the lower level problem parameters $\lambda_1 = \lambda_2 = 0.1$. The batch size is set to be three times larger than the clean data. The number of injected data points in each task is set to be 20% of the clean data.

Figure 4.1 shows the results of the convergence experiments on the two datasets, where x-axis represents the number of iterations in PATOM, and y-axis represents the NMSE averaged over target tasks of Sarcos dataset and the AUC averaged over target tasks on Landmine dataset, respectively. We can see that for all the three kinds of attacks on the two datasets,

[3] `http://yann.lecun.com/exdb/mnist/`.

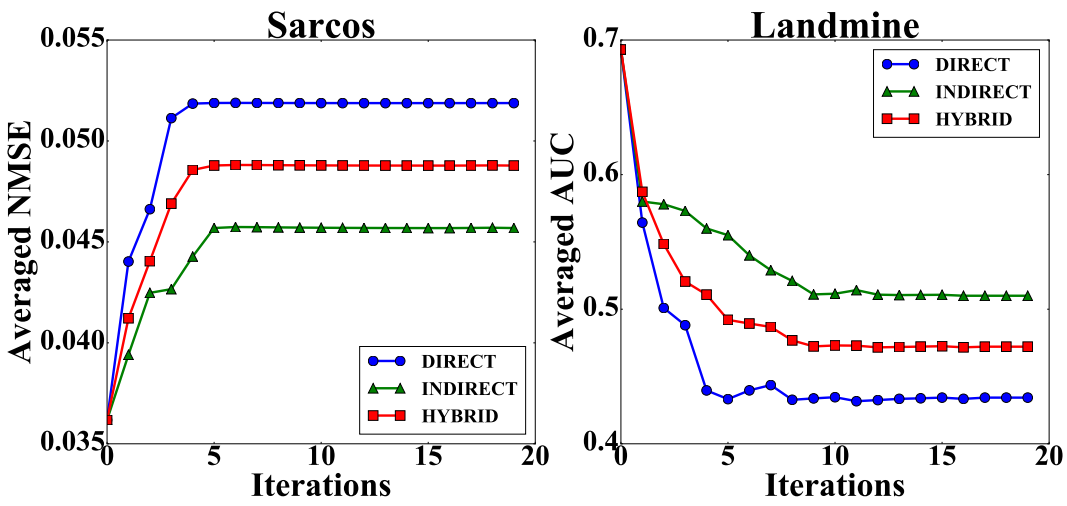


Figure 4.1: Convergence of PATOM.

PATOM converges to local optima in less than 10 iterations, where at iteration 0 the injected data points are randomly initialized. Since existing optimization techniques cannot guarantee global optimal solutions for nonconvex programs, all of the solutions we find are approximate. However, we can get an estimation of the global optimal solution by selecting multiple start points and comparing the local optima. In our experiments, we observe very similar local optima values when choosing multiple start points. Based on this observation, we run PATOM with one start point in our remaining experiments.

4.4.3 Evaluating Solution Qualities

Our second set of experiments evaluates the performance of MTRL under direct attacks and indirect attacks with respect to different datasets. On each dataset, we select 4 different pairs of target task set and attacking task set. Each pair (T_{tar}, T_{att}) is chosen by randomly selecting half of tasks to form T_{tar} and the rest of tasks to form T_{att}. We have $|T_{tar}| = 4$ and $|T_{att}| = 3$ on Sarcos dataset, $|T_{tar}| = 15$ and $|T_{att}| = 14$ on Landmine dataset, and $|T_{tar}| = 5$ and $|T_{att}| = 5$ on MNIST dataset. For a pair (T_{tar}, T_{att}) of each dataset, we compare the averaged NMSE or averaged AUC over the target tasks under four kinds of attacks: direct

attacks, indirect attacks, random direct attacks and random indirect attacks. The last two kinds of attacks are treated as baselines, where the injected data points are randomly chosen. Figure 4.2 shows the results of quality comparison among the four kinds of attacks. Each figure corresponds to a choice of pair (T_{tar}, T_{att}) of the associated dataset. The bold line (dashed line) with circle marker represents direct attacks (random direct attacks); the bold line (dashed line) with square marker represents indirect attacks (random indirect attacks). The budget represents the ratio of the number of injected data points to the number of clean data points. Some interesting findings include:

- Direct attacks are more effective than indirect attacks and random attacks given the same budget. From Figure 4.2, we can see that direct attacks significantly degrade the learning performance on all datasets. For example, on Sarcos dataset, direct attacks with 30% malicious data injected leads to about 50% higher averaged NMSE. However, note that in some scenarios, attackers may have larger budget for launching indirect attacks. Take the recommender system for example, attackers can provide arbitrary number of training data through the malicious accounts created by themselves. In such cases, indirect attacks are also big threats to the learning system.

- Both direct attacks and indirect attacks computed by PATOM significantly outperform random attacks, respectively, which demonstrates that the real-world attackers can do much better than just launching random attacks.

- Different choices of pairs (T_{tar}, T_{att}) influence the attacks' performance. For example, we can see from the second figure of the first row of Figure 4.2, the indirect attacks lead to a higher loss than random direct attacks. However, in the third figure of the first row, the random direct attacks lead to a higher loss than indirect attacks.

- Indirect attacks almost have no effect on MNIST dataset. This is because we can easily learn good classifiers on MNIST dataset using only hundreds of training examples.

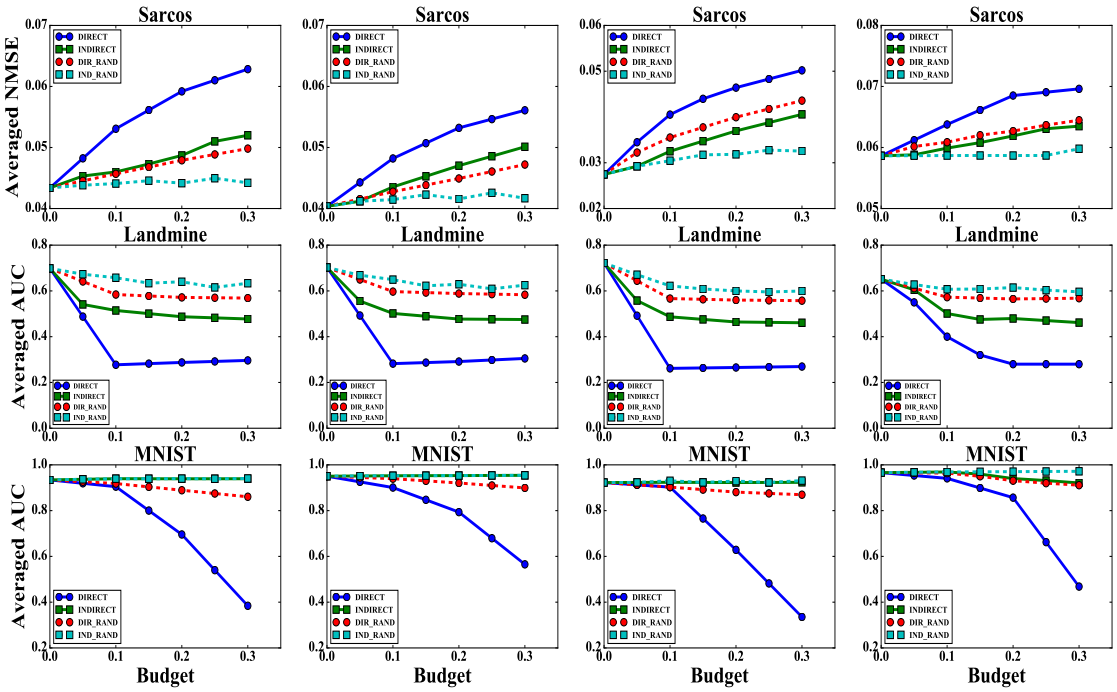
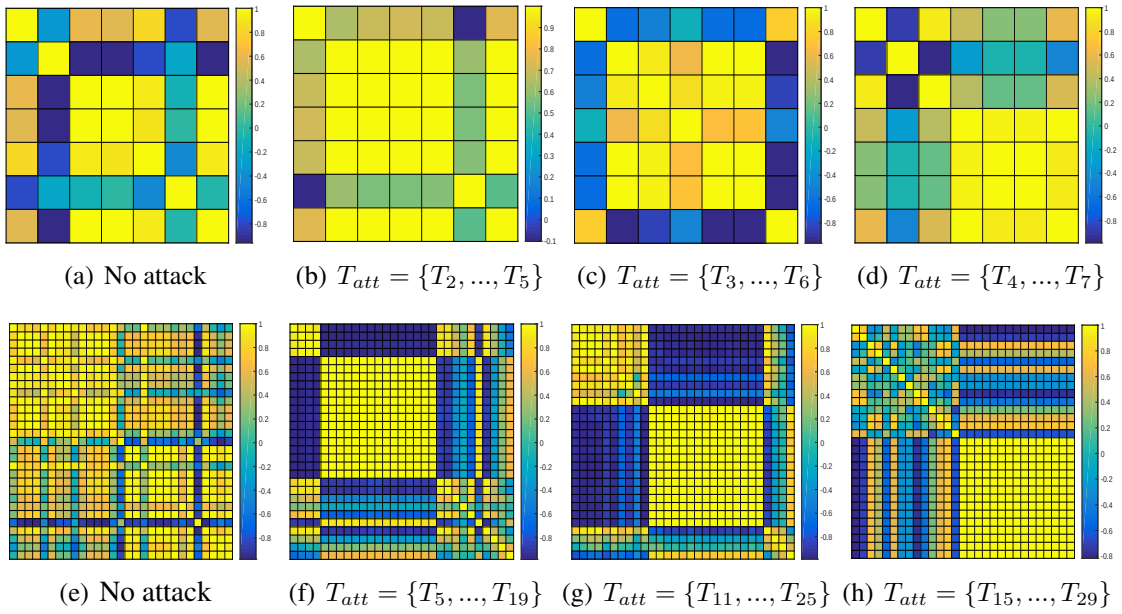


Figure 4.2: Solution quality comparison.

Therefore, each task does not need much help from other tasks and the task correlations are relatively low. Consequently, it is hard for the attacker to launch effective indirect attacks by exploiting task relationships.

4.4.4 Evaluating Task Relationships

Our third set of experiments study the task relationships under different attacks. We fix the target task set as $T_{tar} = \{T_1, T_2, T_3\}$ on Sarcos dataset and $T_{tar} = \{T_1, ..., T_{15}\}$ on Landmine dataset. Then, for each dataset, we select three different attacking task sets and compute three hybrid attacks. We set the amount of injected data to be 30% of the clean data with respect to each task. We convert the learned covariance matrices to correlation matrices and visualize them in Figure 4.3. Since the Sarcos dataset has 7 tasks and the Landmine dataset has 29 tasks, the learned correlation matrix is a 7×7 symmetric matrix on Sarcos dataset and 29×29 symmetric matrix on Landmine dataset.



(a) No attack (b) $T_{att} = \{T_2, ..., T_5\}$ (c) $T_{att} = \{T_3, ..., T_6\}$ (d) $T_{att} = \{T_4, ..., T_7\}$

(e) No attack (f) $T_{att} = \{T_5, ..., T_{19}\}$ (g) $T_{att} = \{T_{11}, ..., T_{25}\}$ (h) $T_{att} = \{T_{15}, ..., T_{29}\}$

Figure 4.3: Visualization of task correlations under attacks.

Figure 4.3 shows that on both datasets the ground-truth task correlations are significantly subverted by the data poisoning attacks. (a) - (d) are the results on Sarcos dataset and (e) - (f) are the results on Landmine dataset. The color of each grid represents the value of the correlation matrix, ranging from -1 (blue) to $+1$ (yellow). The first figure of each row is the ground-truth task correlations learned with clean data. The target task set is set to be $T_{tar} = \{T_1, T_2, T_3\}$ on Sarcos dataset and $T_{tar} = \{T_1, ..., T_{15}\}$ for Landmine dataset and remains the same under different attacks. For example, in Figure 3(a), the ground-truth correlation between tasks 2 and 3 is -0.99, which suggest that the two tasks are highly negatively correlated. However, in Figure 3(b), the correlation between tasks 2 and 3 becomes 0.99, meaning that the two tasks are highly positively correlated. Similar results can be found on Landmine dataset. Moreover, from Figures 3(b) - 3(d) and 3(f) - 3(h), we observe that the attacking tasks are usually highly positive correlated, in contrast with other tasks. This suggests that the machine learner needs to be aware of a group of tasks that form strong local correlations.

4.5 Chapter Summary

This work studies the data poisoning attacks on MTRL models. To the best of our knowledge, we are the first to study the vulnerability of MTL. We categorize the data poisoning attacks into direct attacks, indirect attacks and hybrid attacks based on the real-world scenarios. We propose a bilevel formulation that includes the three kinds of attacks to analyze the optimal attack problems. We propose PATOM, a stochastic gradient ascent based approach that leverages the optimality conditions of MTRL to compute the gradients. We evaluate PATOM in terms of convergence and solution quality on real-world datasets. Experimental results show that PATOM converges to local optima in less than 10 iterations and the attack strategies computed by PATOM significantly outperform baselines. We also study the task correlations under data poisoning attacks.

Chapter 5

Combating Spear Phishing Attacks

This chapter studies the spear phishing attacks and their countermeasures. Highly targeted spear phishing attacks are increasingly common, and have been implicated in many major security breaches. Figure 5.1 shows the general procedure of spear phishing attacks. Email filtering systems are the first line of defense against such attacks. These filters are typically configured with uniform thresholds for deciding whether or not to allow a message to be delivered to a user. However, users have very significant differences in both their susceptibility to phishing attacks as well as their access to critical information and credentials that can cause damage. Recent work has considered setting personalized thresholds for individual users based on a Stackelberg game model. We consider two important extensions of the previous model. First, in our model user values can be substitutable, modeling cases where multiple users provide access to the same information or credential. Second, we consider attackers who make sequential attack plans based on the outcome of previous attacks. Our analysis starts from scenarios where there is only one credential and then extends to more general scenarios with multiple credentials. For single-credential scenarios, we demonstrate that the optimal defense strategy can be found by solving a binary combinatorial optimization problem called PEDS. For multiple-credential scenarios, we formulate it as a bilevel

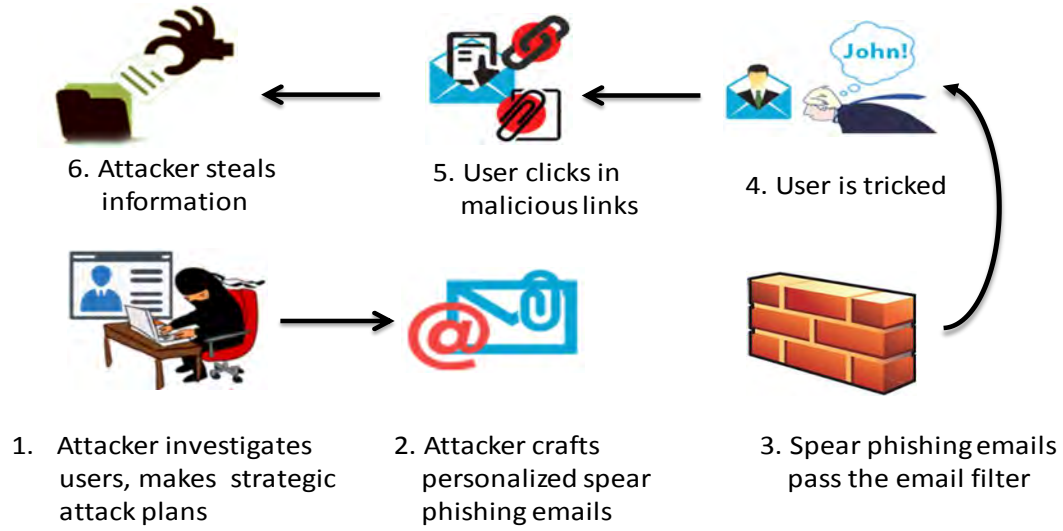


Figure 5.1: Spear phishing attacks.

optimization problem for finding the optimal defense strategy and then reduce it to a single level optimization problem called PEMS using complementary slackness conditions. Experimental results show that both PEDS and PEMS lead to significant higher defender utilities than two existing benchmarks in different parameter settings. Also, both PEDS and PEMS are more robust than the existing benchmarks considering uncertainties.

5.1 Sequential Attacks with A Single Credential

We consider a spear phishing game between an attacker and a defender. The defender (e.g., an organization) has a *credential*[1] that can be accessed by a set of users $U = \{1, 2, ..., |U|\}$. For now we consider only a single credential, and later generalize the model to multiple credentials. The attacker, wanting to gain access to the credential, sends spear phishing emails to the users based on an attack plan taking into account the *susceptibility*, *confidentiality level* and *attack cost* of the users. We denote by a_u the susceptibility of user u, meaning that u will be compromised with probability a_u after a spear phishing email is delivered to her.

[1]We use the generic term "credential" here to mean any critical data or access privilege that the attacker is seeking to gain.

There are many methods to measure a_u, e.g., by sending probe emails to the users [76–78]. We denote by k_u the confidentiality level of user u, meaning that user u can access the credential with probability k_u when she is compromised. The attacker sustains some costs when launching attacks, such as crafting phishing emails, investigating users and writing malware. We denote by c_u the cost of attacking user u.

When receiving emails, the filter first scores them according to their likelihood of being malicious emails, and then delivers only those with scores lower than a given threshold [17, 79]. It is possible that malicious emails are misclassified as normal ones. We call such misclassifications *false negatives*. On the other side, some normal emails might be misclassified as malicious. We call such misclassifications *false positives*. In binary classification, a threshold determines a pair (x_u, y_u) where $x_u, y_u \in [0, 1]$ are the false negative rate and the false positive rate, respectively. Moreover, the relationship between x_u and y_u can be characterized as a function $\Phi : [0, 1] \to [0, 1]$, $y_u = \Phi(x_u)$, which is a Receiver Operating Characteristic (ROC) curve, with y-axis replaced by false positive rate [80]. In practice, Φ is represented by a set of data points and can be approximated by a piecewise linear function ϕ [81]. By adjusting the thresholds, the organization can determine a pair (x_u, y_u) for each user. We will use the false negative rate vector $\mathbf{x}$ to represent the defender's strategy. But note that using $\mathbf{y}$ as the defender's strategy is equivalent to as Φ and ϕ are bijections. Intuitively, the defender actually controls the probability that malicious emails will pass the filter (x_u) and the probability that normal emails will be filtered (y_u).

Figure 5.2 shows the attack flow. The attacker sends a spear phishing email to a targeted user. The email will pass the filter with probability x_u and otherwise be discarded. We assume that the attacker is able to observe whether the email is delivered and opened by the user using email tracking techniques[2]. When receiving the email, the user will be tricked with probability a_u and otherwise be *alerted*. We assume that if the user is tricked, she will

[2]For example, Yesware provides services allowing their clients to view the detailed status of outgoing emails, including whether the emails are opened and the time the receivers spend on each email[82].

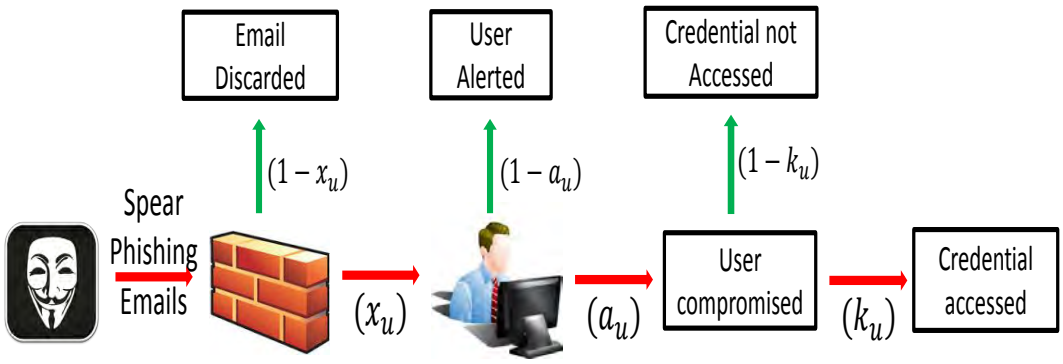


Figure 5.2: Spear Phishing Attack Flow.

be *compromised*, and if the user is alerted, she will be aware of being targeted and not be tricked by subsequential phishing emails. If the user is compromised, the attacker can access the credential with probability k_u.

5.1.1 Stackelberg Spear Phishing Game

We model the interaction between the defender and the attacker as a Stackelberg game. The defender moves first by choosing a false negative probability vector $\mathbf{x}$. After observing $\mathbf{x}$[3], the attacker launches an optimal attack. We denote by $\pi_{\mathbf{x}}$ the attacker's optimal policy that maximizes his expected utility given the defender's strategy $\mathbf{x}$.

We denote by $P_a(\mathbf{x}, \pi_{\mathbf{x}})$ the attacker's expected utility and by $P_d(\mathbf{x}, \pi_{\mathbf{x}})$ the defender's expected utility given strategy profile $(\mathbf{x}, \pi_{\mathbf{x}})$. We denote by L the value of the credential. The attacker suffers a cost c_u each time he attacks user u and he gains L if he accesses the credential. The defender's loss is threefold. (1) The defender loses L if the credential is accessed by the attacker. (2) The defender loses FP_u for per normal email sent to user u filtered. (3) Besides spear phishing attacks, the defender also faces mass attacks (e.g., spam and regular phishing emails), which are usually less harmful than spear phishing attacks. We

[3]We make the worst-case assumption that the attacker knows $\mathbf{x}$ since spear phishers collect security information about the organization before attacking [83].

assume that the probability that a mass attack email passes the filter is x_u[4] and the defender loses N_u for per mass attack email delivered to user u. Note that the defender sustains the second and the third parts of loss constantly as normal emails and mass attack emails are sent to users constantly. However, spear phishing attacks usually happen in a relatively short period. To make the three kinds of losses comparable, we assume that the defender's expected utility is measured in a time period $\mathcal{T}$. We denote by $FP_u^{\mathcal{T}}$ the expected loss of misclassifying normal emails sent to user u and by $N_u^{\mathcal{T}}$ the expected loss of delivering mass attack emails sent to user u during $\mathcal{T}$, which can be computed by

$$FP_u^{\mathcal{T}} = FP_u \times E[\text{number of normal emails sent to } u \text{ during } \mathcal{T}]$$
$$N_u^{\mathcal{T}} = N_u \times E[\text{number of mass attack emails sent to } u \text{ during } \mathcal{T}]$$

The defender's loss from filtering normal emails and delivering mass attack emails can be simply represented as the summation of the loss from every individual user, $\sum_{u\in U} x_u N_u^{\mathcal{T}}$ and $\sum_{u\in U} \phi(x_u) FP_u^{\mathcal{T}}$ respectively. However, the defender's loss from spear phishing attacks is not cumulative. We denote by $\rho^{\mathcal{T}}$ the probability that the spear phishing attacks occur in time period $\mathcal{T}$ and by $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ the probability that the attacker will access the credential given the strategy profile $(\mathbf{x}, \pi_{\mathbf{x}})$. Then the defender's expected utility can be represented as

$$P_d(\mathbf{x},\pi_{\mathbf{x}}) = -\rho^{\mathcal{T}}\theta(\mathbf{x},\pi_{\mathbf{x}})L - \sum_{u\in U} x_u N_u^{\mathcal{T}} - \sum_{u\in U} \phi(x_u) FP_u^{\mathcal{T}}$$

We consider the widely used strong Stackelberg equilibrium (SSE) as our solution concept [84–87].

[4]This assumption means that the classification accuracies for spear phishing emails and mass attack emails are the same. Note that our approach can be easily extended to the case where these accuracies are different, by introducing a function that captures the relationship between these accuracies.

Definition 1. *If a strategy profile* $(\mathbf{x}^*, \pi_{\mathbf{x}^*})$ *such that* $P_d(\mathbf{x}^*, \pi_{\mathbf{x}^*}) \geq P_d(\mathbf{x}, \pi_{\mathbf{x}})$ *holds for any possible* $\mathbf{x}$*, under the assumption that the attacker plays a best response and breaks ties among multiple optimal policies in favor of the defender, then* $(\mathbf{x}^*, \pi_{\mathbf{x}^*})$ *is an SSE strategy profile.*

5.2 Optimal Attack with A Single Credential

In this section, we model the attacker's decision making as a Markov Decision Process (MDP) and show that the MDP can be solved by a linear program.

5.2.1 Attacker's MDP

The attacker's MDP can be represented as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \pi)$. $\mathcal{S} = \{s | s \subseteq U\} \cup \{s^n, s^y\}$ is the state space that consists of *non-terminal states* and two *terminal states* s^n, s^y. A non-terminal state corresponds to a subset of the user set U that represents the users who have not been alerted or compromised. The initial state is $s_0 = U$. The terminal state s^n represents the situation where the attacker stops attacking without accessing the credential, while s^y represents the situation where the attacker stops attacking with the credential accessed. $\mathcal{A} = \{a | a = u \in U \text{ or } a = stop\}$ is the attacker's action space where $a = u$ means that the attacker chooses to attack user u, and $a = stop$ means that the attacker stops attacking. We denote by $\mathcal{A}^s = \{a | a = u \in s \text{ or } a = stop\}$ the attacker's action space at non-terminal state s, since the attacker only attacks users that have not been alerted or compromised. Transition function $T(s, a, s')$ represents the probability that s transitions to s' by executing action a. Reward function $R(s, a, s')$ represents the attacker's reward when s transitions to s' by executing action a. $\pi : \mathcal{S} \to \mathcal{A}$ is a deterministic function that projects each non-terminal state to an action.

Now we define T and R. We assume that the terminal states always transition to themselves with probability 1 and with reward 0. For any non-terminal state s, if the attacker stops attacking, s transitions to s^n with reward 0. If the attacker chooses to attack user $u \in \mathcal{A}^s$, there are four possible transitions: (1) If the malicious email fails to pass the filter, s transitions to itself. The transition probability is $1 - x_u$ and the reward is $-c_u$. (2) If the email is delivered and user u is alerted, s transitions to $s^{-u} = s \setminus \{u\}$. The transition probability is $x_u(1 - a_u)$ and the reward is $-c_u$. (3) If the email passes the filter and u is compromised, however u does not have access to the credential, then s transitions to $s^{-u} = s \setminus \{u\}$. The transition probability is $x_u a_u(1 - k_u)$ and the reward is $-c_u$. Note that in both transitions (2) and (3), s transitions to s^{-u} with the same reward $-c_u$. Therefore they can be merged into one transition with probability $x_u(1 - a_u) + x_u a_u(1 - k_u) = x_u(1 - a_u k_u)$ and with reward $-c_u$. (4) The email passes the filter, user u is compromised and she can access the credential, s transitions to s^y. The transition probability is $x_u a_u k_u$. The transition function T and the reward function R can be summarized as:

	$T(s, a, s')$	$R(s, a, s')$
$a = stop, s' = s^n$	1	0
$a = u \in \mathcal{A}^s, s' = s$	$1 - x_u$	$-c_u$
$a = u \in \mathcal{A}^s, s' = s^{-u}$	$x_u(1 - a_u k_u)$	$-c_u$
$a = u \in \mathcal{A}^s, s' = s^y$	$x_u a_u k_u$	$L - c_u$

5.2.2 Solving the MDP

In this section, we review how the MDP can be solved by a linear program [88, 89]. The value function $V^\pi : \mathcal{S} \to \mathbb{R}$ represents the attacker's expected utility when his current state is s and he follows a policy π afterwards. Moreover, we denote by V^* the value function when the attacker follows the optimal policy $\pi_{\mathbf{x}}$. Then the attacker's expected utility can be

written as

$$P_a(\mathbf{x}, \pi_{\mathbf{x}}) = V^*(s_0).$$

The attacker's MDP can be solved by the following linear program [89].

$$\min_{\mathbf{V}_a^*} \quad \sum_{s \in \mathcal{S} \setminus \mathcal{S}^T} \mu(s) V_a^*(s) \tag{5.1}$$

$$\text{s.t.} \quad V_a^*(s) \geq \sum_{s' \in \mathcal{S}} T(s, a, s') \left[R(s, a, s') + V_a^*(s') \right]$$

$$\forall a \in \mathcal{A}^s, \forall s \in \mathcal{S} \setminus \mathcal{S}^T \tag{5.2}$$

$$V_a^*(s) = 0, \quad \forall s \in \mathcal{S}^T \tag{5.3}$$

where $S^T = \{s^n, s^y\}$ denotes the set of terminal states and $\mu(s)$ is the probability that the MDP starts from state s. Since we have an initial state s_0, $\mu(s) = 1$ if $s = s_0$ and 0 otherwise. The optimal policy $\pi_{\mathbf{x}}$ can be obtained:

$$\pi_{\mathbf{x}}(s) = \arg\max_{a \in \mathcal{A}^s} Q(s, a), \forall s \in \mathcal{S} \setminus \mathcal{S}^T,$$

where $Q(s, a) = \sum_{s' \in \mathcal{S}} T(s, a, s') \left[R(s, a, s') + V_a^*(s') \right]$.

5.3 Optimal Defense with A Single Credential

The defender seeks a false negative probability vector $\mathbf{x}$ that maximizes her expected utility given that the attacker plays the optimal policy $\pi_{\mathbf{x}}$. The defender's optimization problem is given by the following bilevel optimization problem.

$$\max_{\mathbf{x}} \quad P_d(\mathbf{x}, \pi_{\mathbf{x}}) \tag{5.4}$$

$$\text{s.t.} \quad x_u \in [0,1],\ \forall u \in U \tag{5.5}$$

$$\pi_{\mathbf{x}} \in \arg\max_{\pi} V^{\pi}(s_0) \tag{5.6}$$

The objective function 5.4 represents the defender's expected utility. Constraint 5.5 indicates that the false negative rate can only be chosen from $[0, 1]$. Constraint 5.6, i.e., the lower level problem, assures that the attacker always responds optimally. The hardness of solving this bilevel problem is twofold. First, $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ in $P_d(\mathbf{x}, \pi_{\mathbf{x}})$ does not have an explicit representation with respect to variables $\mathbf{x}$. Second, the lower level problem is hard to be characterized by a set of constraints. We will first show how to represent $\theta(\mathbf{x}, \pi_{\mathbf{x}})$, and then show that this bilevel problem is equivalent to a single level problem called PEDS.

5.3.1 Representing $\theta(\mathbf{x}, \pi_{\mathbf{x}})$

In fact, $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ is the probability that the attacker ends in the terminal state s^y given that he follows the optimal policy $\pi_{\mathbf{x}}$. Before we show how to represent $\theta(\mathbf{x}, \pi_{\mathbf{x}})$, we introduce two concepts: *reachable states* and *potential attack set*. Once a policy is determined, the MDP is reduced to a Markov chain where only some states (called *reachable states*) can be reached from the initial state if we consider the Markov chain as a graph. For example, if s_0=$\{u_1, u_2\}$ and $\pi(s_0)$=u_1, then state s=$\{u_1\}$ cannot be reached from s_0 with a nonnegative probability. We denote by $\Delta(\pi)$ the set of reachable states given the policy π. A policy π projects each reachable state $s \in \Delta(\pi)$ to an action $a \in \mathcal{A}^s$. We denote by $\Gamma(\pi)$ the potential attack set, which is the set of users that are projected from the reachable states under the policy π, i.e., $\Gamma(\pi) = \{\pi(s) | s \in \Delta(\pi)\}$. Lemma 1 states that if the immediate expected gain

of attacking user u (i.e., $x_u a_u k_u L$) is greater than the attack cost c_u, then the user is in the potential attack set. [5]

Lemma 1. $u \in \Gamma(\pi_{\mathbf{x}})$ *if and only if* $x_u a_u k_u L > c_u$.

Lemma 2 shows that $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ can be easily computed given the potential attack $\Gamma(\pi_{\mathbf{x}})$.

Lemma 2.

$$\theta(\mathbf{x}, \pi_{\mathbf{x}}) = \begin{cases} 1 - \prod_{u \in \Gamma(\pi_{\mathbf{x}})}(1 - a_u k_u), & \textit{if } \Gamma(\pi_{\mathbf{x}}) \neq \emptyset \\ 0, & \textit{if } \Gamma(\pi_{\mathbf{x}}) = \emptyset \end{cases}.$$

Combining Lemmas we can show that even though the attacker may have multiple optimal policies, they have the same potential attack set. Therefore, $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ does not change for different optimal attack policies.

Theorem 1. *The defender's expected utility remains the same no matter how the attacker breaks ties, i.e., choosing any optimal policy.*

5.3.2 PEDS: Reduced Single Level Problem

Now we show how to solve Eqs.(5)-(7) based on the lemmas. We define a function Λ_u for each user u:

$$\Lambda_u(x) = x N_u^{\mathcal{T}} + \phi(x) F P_u^{\mathcal{T}}, \ x \in [0, 1].$$

$\Lambda_u(x)$ represents the total loss from mass attacks and false positives of user u if she is assigned a false negative probability x. Λ_u is a piecewise linear function since it is the sum of a linear function and a piecewise linear function. Therefore, we can easily find a set $\arg\min_x \Lambda_u$ for each user.

[5] All proofs of Lemmas and Theorems are in Chapter 8.

We rewrite the defender's utility as

$$P_d(\mathbf{x}, \pi_{\mathbf{x}}) = -\rho^{\mathcal{T}}\theta(\mathbf{x}, \pi_{\mathbf{x}})L - \sum_{u \in U} \Lambda_u(x_u).$$

Lemma 2 indicate that the value of $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ depends on the potential attack set $\Gamma(\pi_{\mathbf{x}})$. We define a set $\mathcal{U} = \{u | \frac{c_u}{La_u k_u} \in [0, 1], u \in U\}$. If $u \in U \setminus \mathcal{U}$, the optimal false negative rate x_u^* can be any arbitrary point of $\arg\min_x \Lambda_u$ since $u \in \Gamma(\pi_{\mathbf{x}})$ holds for any $x_u \in [0, 1]$. If $u \in \mathcal{U}$, it holds that $u \notin \Gamma(\pi_{\mathbf{x}})$ when $x_u \in [0, \frac{c_u}{La_u k_u}]$ and $u \in \Gamma(\pi_{\mathbf{x}})$ when $x_u \in (\frac{c_u}{La_u k_u}, 1]$. Given $u \in \mathcal{U}$, we denote by x_u^1 the optimal false negative rate if $u \notin \Gamma(\pi_{\mathbf{x}})$ and by x_u^2 the optimal false negative rate if $u \in \Gamma(\pi_{\mathbf{x}})$.

Theorem 2. *x_u^1 is an arbitrary point in $\arg\min_{x \in [0, \frac{c_u}{La_u k_u}]} \Lambda_u$ and x_u^2 is an arbitrary point in $\arg\min_{x \in (\frac{c_u}{La_u k_u}, 1]} \Lambda_u$.*

Then Problem 5.4 are equivalent to the following binary combinatorial optimization problem, which we call PEDS (Personalized thrEsholds in Defending Sequential spear phishing attacks):

$$\max_{\alpha} \quad -\rho^{\mathcal{T}}(1 - \prod_{u \in U} \beta_u)L - \sum_{u \in U} \Lambda_u(x_u) \tag{5.7}$$

$$\text{s.t.} \quad x_u = x_u^0, \ \forall u \in U \setminus \mathcal{U} \tag{5.8}$$

$$\beta_u = 1, \ \forall u \in U \setminus \mathcal{U} \tag{5.9}$$

$$x_u = x_u^1 + (x_u^2 - x_u^1)\alpha_u, \ \forall u \in \mathcal{U} \tag{5.10}$$

$$\beta_u = 1 - a_u k_u \alpha_u, \ \forall u \in \mathcal{U} \tag{5.11}$$

$$\alpha_u \in \{0, 1\}, \ \forall u \in \mathcal{U} \tag{5.12}$$

where x_u^0 can be an arbitrary point from $\arg\min_x \Lambda_u$. α_u is the indicator of whether user u is in the potential attack set $\Gamma(\pi_{\mathbf{x}})$. $\alpha_u = 0$ indicates that u is in $\Gamma(\pi_{\mathbf{x}})$ and 1 otherwise. Since

PEDS's decision variables are binary, we can find the optimal solutions by using CPLEX CP Optimizer.

5.4 Multiple-Credential Model

An organization may need to protect many different credentials or pieces of sensitive information. We now consider the multiple-credential case, where the attacker's decision making can still be modeled as an MDP. Note that Problem 5.4 is still the defender's optimization problem except that π and V represent the policy and the value function of the new MDP. In this section, we first introduce the attacker's MDP with multiple credentials. Then we give the dual formulation of Problem 5.1 and show that using complementary slackness conditions, Problem 5.6 (i.e., the lower level optimization problem) can be replaced by a set of constraints, which guarantee that the attacker plays the best response. Consequently, bilevel problem Problem 5.4 is reduced to a single level problem which can be directly solved.

5.4.1 Optimal Attack with Multiple Credentials

We denote by $H = \{1, 2, ..., |H|\}$ the set of credentials, by L_h the value of the credential h and by m_u^h the probability that user u can access credential h. With multiple credentials, the attacker's MDP can be represented as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \pi)$. $\mathcal{S} = \mathcal{P}(U) \otimes \mathcal{P}(H)$ is the state space, where $\mathcal{P}(U)$ $(\mathcal{P}(H))$ is the power set of U (H). A state $s \in \mathcal{S}$ can be represented as $s = s(U) \otimes s(H)$, where $s(U) \subseteq U$ represents the set of users that have not be alerted or compromised and $s(H) \subseteq H$ represents the set of credentials that have not been accessed by the attacker. s is a terminal state if either $s(U) = \emptyset$ or $s(H) = \emptyset$, i.e., all the users have been alerted or compromised, or all credentials have been accessed. We use $\mathcal{S}^T$ to represent the set of terminal states. At each non-terminal state, the attacker's action space is

$\mathcal{A}^s = \{a | a = u \in s(U) \text{ or } a = stop\}$, in the sense that the attacker does not attack users that have been alerted or compromised. $\pi : \mathcal{S} \to \mathcal{A}$ represents a policy of the attacker.

$T(s, a, s')$ represents the probability and $R(s, a, s')$ represents the attack reward that s transitions to s' by executing action a. We assume that terminal states transition to themselves with probability 1 and reward 0. We define the transitions and rewards as follows. (1) If $s = s(U) \otimes s(H)$ is a non-terminal state and the attacker chooses to stop attacking at s, s transitions to the terminal state $s' = \emptyset \otimes s(H)$ with probability 1 and reward 0. (2) If $s = s(U) \otimes s(H)$ is a non-terminal state and the attacker chooses to attack a user $u \in s(U)$ at s, there are 3 kinds of transitions. (2.1) The malicious email is filtered, in which case s transitions to itself. The transition probability is $1 - x_u$ and the reward is $-c_u$. (2.2) The malicious email is delivered and user u is alerted, in which case s transitions to $s' = s(U) \setminus \{u\} \otimes s(H)$. The transition probability is $x_u(1 - a_u)$ and the reward is $-c_u$. (2.3) The malicious email is delivered and user u is compromised, after which the attacker will access each credential $h \in H$ with probability m_u^h. We have

$$T(s, a = u, s') = \begin{cases} 0, \text{ if } s'(U) \neq s(U) \setminus \{u\} \text{ or } s'(H) \not\subseteq s(H), \\ x_u a_u \prod_{h \in s(H) \setminus s'(H)} m_u^h \prod_{h \in s'(H)} (1 - m_u^h), \text{ otherwise.} \end{cases}$$

The associated rewards

$$R(s, a{=}u, s') = \begin{cases} 0, \text{ if } s'(U) \neq s(U) \setminus \{u\} \text{ or } s'(H) \not\subseteq s(H), \\ \sum_{h \in s(H) \setminus s'(H)} L_h, \text{ otherwise.} \end{cases}$$

Note that the new MDP can still be solved by linear program 5.1.

5.4.2 Defender's Loss from Spear Phishing Attacks

When there are multiple credentials, the probability of losing the credentials cannot be computed in the same way as in the single-credential case. We introduce another way to represent the defender's expected utility. Consider the dual of linear program 5.1:

$$\max_{\mathbf{W}} \sum_{s\in\mathcal{S}\setminus\mathcal{S}^T}\sum_{a\in\mathcal{A}^s}\sum_{s'\in\mathcal{S}} T(s,a,s')R(s,a,s')W(s,a) \tag{5.13}$$

$$\text{s.t.} \sum_{a'\in\mathcal{A}^{s'}} W(s',a') = \mu(s') + \sum_{s\in\mathcal{S}\setminus\mathcal{S}^T}\sum_{a\in\mathcal{A}^s} W(s,a)T(s,a,s')$$

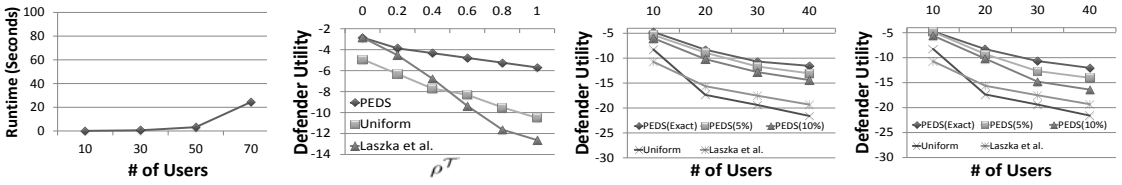
$$\forall s' \in \mathcal{S}\setminus\mathcal{S}^T \tag{5.14}$$

$$W(s,a) \in \mathbb{R}^+, \forall a \in \mathcal{A}^s, \forall s \in \mathcal{S}\setminus\mathcal{S}^T \tag{5.15}$$

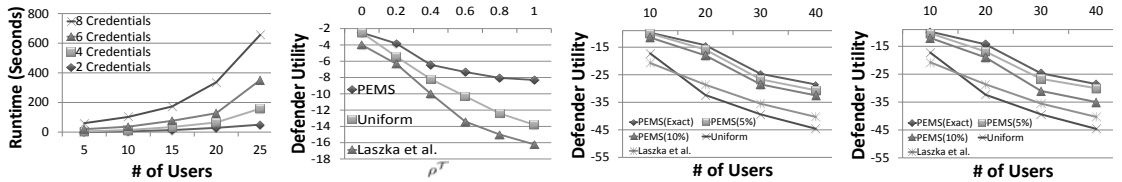
The dual variable $\mathbf{W}$ is called the *occupation measure* [90]. $W(s,a)$ can be interpreted as the expected total number of times that the system is in state s and action a is executed. $\sum_{a\in\mathcal{A}^s} W(s,a)$ is the expected total number of visits to state s. We define a reward function $R_d(s,a,s')$ for the defender.

$$R_d(s,a,s') = \begin{cases} -(R(s,a,s') + c_u), & \text{if } a = u \in \mathcal{A}^s, \\ 0, & \text{if } a = stop. \end{cases}$$

Recall that $R(s,a,s')$ is the attacker's reward when he executes action a and the state transitions from s to s'. In fact, $R(s,a,s')$ consists of the gain of accessing some credentials (positive) and the cost of attack (negative). The defender's loss can thus be represented as $-(R(s,a,s') + c_u)$ if $a = u \in \mathcal{A}^s$, and 0 if $a = stop$. Therefore the defender's expected loss from spear phishing attacks can be represented as $\sum_{s\in\mathcal{S}\setminus\mathcal{S}^T, a\in\mathcal{A}^s} W(s,a)\sum_{s'\in\mathcal{S}} T(s,a,s')R_d(s,a,s')$.



(a) PEDS: Runtime (b) PEDS: Solution quality (c) PEDS:Robustness w.r.t. c_u (d) PEDS:Robustness w.r.t. a_u



(e) PEMS: Runtime (f) PEMS: Solution quality (g) PEMS:Robustness w.r.t. c_u (h) PEMS:Robustness w.r.t. a_u

Figure 5.3: Performance of PEDS and PEMS.

5.4.3 Single Level Formulation

It follows that feasible solutions $\mathbf{V}_a^*$ and $\mathbf{W}$ are optimal for the original LP and its dual problem if the following complementary slackness conditions are satisfied:

$$\{V_a^*(s) - \sum_{s' \in \mathcal{S}} T(s,a,s')[R(s,a,s') + V_a^*(s')]\}W(s,a) = 0,$$
$$\forall a \in \mathcal{A}^s, \forall s \in \mathcal{S} \setminus \mathcal{S}^T. \tag{5.16}$$

Then the bilevel problem 5.4 can be converted to the following single level problem, which we call PEMS (Personalized thrEsholds in protecting Multiple credentialS):

$$\max_{\mathbf{x}} \quad \rho^{\mathcal{T}} \sum_{s\in\mathcal{S}\setminus\mathcal{S}^T, a\in\mathcal{A}^s} W(s,a) \sum_{s'\in\mathcal{S}} T(s,a,s') R_d(s,a,s') - \sum_{u\in U} \Lambda_u(x_u) \tag{5.17}$$

s.t. Eqs.(3), (4), (15) and (17)

$$W(s,a) \in \mathbb{R}^+, \forall a \in \mathcal{A}^s, \forall s \in \mathcal{S} \setminus \mathcal{S}^T \tag{5.18}$$

$$x_u \in [0,1], \ \forall u \in U \tag{5.19}$$

PEMS is a nonlinear problem and we can solve it by KNITRO, a popular solver that can handle many complicated non-linear optimization problems.

5.5 Experimental Evaluation

We evaluate PEDS and PEMS in terms of runtime, solution quality and robustness. All values of parameters are uniformly randomly generated from an interval unless otherwise specified. Specifically, values of credentials are generated from [10,15]. Attack costs c_u are generated from [0,2]. Users' susceptibilities a_u and their accesses to credentials k_u (m_u^h in multiple-credential case) are generated from [0,0.5]. Losses from mass attacks $N_u^{\mathcal{T}}$ and losses from false positives $FP^{\mathcal{T}}$ are generated from [0,1] and [0,5], respectively. PEDS is solved by CPLEX CP Optimizer (version 12.6) and PEMS is solved by KNITRO (version 9.0). All computations were performed on a 64-bit PC with 16 GB RAM and a quad-core Intel E5-1650 3.20GHz processor. We use a 10-section piecewise linear function ϕ to approximate the original false negative-false positive function Φ, which is drawn from prior work [19].

We compare the solutions computed by PEDS and PEMS with two existing benchmarks.

- **Uniform**: All users have a uniform false negative rate $x^* \in [0, 1]$ that maximizes the defender's expected utility. We discretize the interval [0,1] into 1000 equal-distance points and search among these points to find the optimal value x^*. In addition, we will use x^* as the starting point when solving PEMS.

- **Laszka et al.** [19]: An existing approach for personalized threshold setting assumes that the defender's expected loss from spear phishing attacks is the sum of users' individual expected losses. Following our notations, user u's individual loss is set to the immediate expected loss $x_u a_u k_u L$ in the single-credential case and $x_u a_u \sum_{h \in H} m_u^h L_h$ in the multiple-credential case.

Scalability Analysis

We first evaluate the scalability of PEDS and PEMS. We assume that each credential can only be accessed by 30% of total users with nonzero probability considering that sensitive information is usually accessed by a small portion of total users. Figure 5.3(a) shows that PEDS can solve games with 70 users in 23s. Figure 5.3(e) shows that both the number of users and the number of credentials have significant influence on the runtime of PEMS. PEMS runs slower than PEDS since nonlinear problems are usually more computationally consuming. However, we argue that both PEDS and PEMS are applicable in real-world cases due to two reasons. First, spear phishing attacks, unlike mass attacks, usually jeopardize a small group of people. For example, in the attack towards the US Nuclear Regulatory Commission, only 16 employees are targeted [91]. Second, in our model the defender does not need to update her strategy adaptively so that the runtime requirement is not very high.

Solution Quality Comparisons

We compare our approaches with two benchmarks for different values of $\rho^{\mathcal{T}}$, which measures the probability that spear phishing attacks happen in $\mathcal{T}$. Note from Figure 5.3(b) and Figure 5.3(f), when $\rho^{\mathcal{T}} = 0$, meaning that there is no spear phishing attacks, our approaches lead to the same defender utilities as Laszka et al.. In this case the defender's optimal strategy is simply setting $x_u = \arg\max_x \Lambda_u(x)$ for each user u, considering only mass attacks and false positives. With $\rho^{\mathcal{T}}$ growing, Laszka et al. performs significantly worse than our approaches.

Our approaches outperform the optimal uniform strategy. This is because that the optimal uniform strategy is computed under the constraints that all users' thresholds are equal. We compare our approaches with the optimal uniform strategy to show how much improvement "personalization" can bring. Our approaches also outperform Laszka et al.. The reason is, when computing defender strategy of Laszka et al., the attacker is assume to launch a non-sequential attack. It's not surprising that this strategy performs poorly when against a sequential decision making attacker. Moreover, note from Figure 5.3(b) that Laszka et al. performs even worse than the optimal uniform strategy when $\rho^{\mathcal{T}} > 0.5$. This indicates that estimation about the attacker's behaviour may be even more important than "personalization".

Robustness Analysis

Defender's estimation of the attack cost and user susceptibility may not be perfect. We consider a noise on c_u and a_u. In this section of experiments, estimations of c_u are drawn uniformly from two intervals $c_u \cdot [1-5\%, 1+5\%]$ and $c_u \cdot [1-10\%, 1+10\%]$. Estimations of a_u are drawn from $a_u \cdot [1-5\%, 1+5\%]$ and $a_u \cdot [1-10\%, 1+10\%]$. We use these estimations to compute the defender strategy and then use this strategy to compute the defender's utility in

the accurate parameter setting. Figure 5.3(c) (Figure 5.3(g)) shows that PEDS (PEMS) outperforms both benchmarks even with a 10% error range on attack cost c_u in single-credential (multiple-credential) case. Similarly, Figure 5.3(d) (Figure 5.3(h)) shows that PEDS (PEMS) outperforms both benchmarks w.r.t. the susceptibility measurement a_u in single-credential (multiple-credential) case.

5.6 Chapter Summary

This work studies the problem of setting personalized email filtering thresholds against sequential spear phishing attacks. We first consider a simple single-credential case and then extend it to a more general multiple-credential case. Our approach features the following novelties. (1) An MDP framework is proposed to model the sequential decision making attacker. (2) An efficient binary combinatorial optimization formulation PEDS is proposed for computing solutions for the single-credential case. (3) With multiple credentials, the defender's loss from spear phishing attacks is represented by a linear combination of dual variables. (4) A single level formulation PEMS, which is reduced from the defender's bilevel problem using complementary slackness conditions, is proposed for computing solutions for the multiple-credential case.

Chapter 6

Combating Fraudulent Sellers in E-Commerce

This chapter studies fraud transactions in e-commerce and countermeasures to combat fraudulent sellers. Conducting fraud transactions has become popular among e-commerce sellers to make their products favorable to the platform and buyers, which decreases the utilization efficiency of buyer impressions [1] and jeopardizes the business environment. Fraud detection techniques are necessary but not enough for the platform since it is impossible to recognize all the fraud transactions. In this work, we focus on improving the platform's impression allocation mechanism to maximize its profit and reduce the sellers' fraudulent behaviors simultaneously. First, we learn a seller behavior model to predict the sellers' fraudulent behaviors from the real-world data provided by Alibaba. Then, we formulate the platform's impression allocation problem as a continuous Markov Decision Process (MDP) with unbounded action space. In order to make the action executable in practice and facilitate learning, we propose a novel deep reinforcement learning algorithm DDPG-ANP that introduces an action norm penalty to the reward function. Experimental results show that our algorithm significantly

[1] In our work, a buyer impression is defined as a buyer click on an item.

outperforms existing baselines in terms of scalability and solution quality.

6.1 Impression Allocation with Fraudulent Sellers

We begin by introducing some basics of the impression allocation mechanism. When a buyer searches a keyword on an e-commerce website, the platform retrieves a set of related items and displays them in some order. We assume that there are n products to be placed into n slots and each slot i is associated with a click-through rate ctr_i, which means the probability that the buyer will click on slot i. Usually, the click-through rates of slots decrease from the top to the bottom in the list of search results. The platform computes a score for each product and places the products with higher scores to the slots with higher click-through rates. We denote by σ : product $\rightarrow \mathbb{R}$ the score function and refer it to the impression allocation mechanism as it determines the ranking of products and the number of impressions each product gets.

We consider the impression allocation problem as a sequential decision making process as the platform can update its strategy regularly. We assume that the platform updates its strategy every 3 days and consider 3 days as one time step. At time t, each seller i determines the price $price_i^t$ of his product[2]. We denote by b_i^t the number of buyer impressions that seller i receives at time t. We denote by cvr_i^t the conversion rate and by r_i^t the number of real transactions of product i at time t. We denote by f_i^t the number of detected fake transactions of product i at time t. Note that the real number of fake transactions might be different from f_i^t due to detection error. We use a feature vector $\mathbf{v}_i^t = (b_i^t, cvr_i^t, price_i^t, r_i^t, f_i^t)$ to represent seller i's record at time t. The platform determines a score function $\sigma^{t+1} : \mathbb{R}^5 \rightarrow \mathbb{R}$ that maps the each seller's record at time t to a real value, which is used to determine the sellers' ranking at time $t+1$. Table 6.1 summarizes the notations.

[2]For simplicity, we assume that each product is sold only by one seller and each seller sells only one product.

Notation	Description
ctr_i^t	click-through rate of slot i at time t
b_i^t	number of impressions of product i at time t
cvr_i^t	conversion rate of product i at time t
$price_i^t$	price of product i at time t
r_i^t	number of real transactions of i at time t
f_i^t	number of detected fake transactions of product i at time t

Table 6.1: Key seller features.

Gross Merchandise Volume (GMV) is a common metric for evaluating the scale of transactions, which is defined by the product of the total number of transactions and the average price of each transaction. We assume that each transaction contains only one product since we observe in our real-world data that over 95% of the transactions contain only one item. Existing works define the platform's utility as the positive GMV (GMV generated by real transactions), which fails to capture the goal of reducing fraudulent behaviors [30]. We define the platform's utility as the weighted sum of positive GMV and negative GMV (GMV generated by fake transactions) as follows.

$$U^T = \sum_{t=1}^{T} \sum_{i=1}^{n} (r_i^t - \lambda f_i^t) \cdot price_i^t$$

where $\lambda \geq 0$ represents the weight of negative GMV. We will formulate the platform's decision making problem as an MDP. As the transitions of the MDP cannot be explicitly defined, we exploit deep reinforcement learning to learn the platform's optimal policy. Since it is impractical to get the seller's responses to the platform's impression allocation mechanism in real time, we learn a seller behavior model using real-world data which is used to predict the sellers' fraudulent behaviors.

6.2 Learning Seller Behavior Model

In this section, we show how to learn a seller behavior model to predict the number of transactions that the seller intend to fake. Since the sellers usually decide whether to cheat based on the number of impressions received in the last few days, we assume that the expected number of fake transactions at time $t+1$ depends only on the feature vector until time t[3]. In this section, we describe how to estimate $\mathbb{E}[f_i^{t+1}|\mathbf{v}_i^t]$ from our real-world dataset. The learned seller behavior model will be used to simulate the environment for deep reinforcement learning in Section 6.3.

Dataset and Preprocessing. Our dataset is provided by one of the largest e-commerce company in the world. The dataset contains over one million records of products in the category of "women clothes" collected in one months. Each record is a feature vector that describes the statistics of a product in 3 consecutive days, including the product ID, the number of total exposures, the number of total buyer clicks, the average conversion rate, the average price, the average number of real and fake transactions, the total GMV, refund rate, and the buyer's feedback on the product's quality, logistics and the seller's service quality. We filter the products whose number of total transactions (including both real and fake transactions) in 3 days is zero since these inactive products are not important to the platform. We also filter the products whose average transaction price is lower than 1 dollar since these products are usually not real women clothes but are the seller's marketing tools. We uniformly randomly sample $100,000$ products from the remaining products. Then, we select 5 features of each product i and obtain the feature vector $\mathbf{v}_i^t = (b_i^t, cvr_i^t, price_i^t, r_i^t, f_i^t)$. In addition, we add a new feature $\frac{f_i^t}{r_i^t+f_i^t}$ to $\mathbf{v}_i^t$, which can be interpreted as the stage of a fraud product. On one hand, a fraud product often has a low number of real transactions at its early stages so that the ratio $\frac{f_i^t}{r_i^t+f_i^t}$ is relatively high. On the other hand, when the number

[3]Note that the real number of fake transactions is unknown to the platform. However, it is usually positively correlated with the number of detected fake transactions.

Model	LR	RR	LASSO	EN	DNN
nMSE	0.19	0.19	0.28	0.26	0.17
Runtime(s)	0.014	0.015	0.013	0.014	2.265

Table 6.2: Performance of regression models.

of real transactions goes high, the seller has less incentive to get more impression by faking transactions.

We treat the problem of predicting a seller's fraud behavior as a regression task, where each training point is a product's feature vector $\mathbf{v}_i^t = (b_i^t, cvr_i^t, price_i^t, r_i^t, f_i^t, \frac{f_i^t}{r_i^t+f_i^t})$ and the label is f_i^{t+1}, which represents the ground truth number of fake transactions in the following 3 consecutive days. However, we observe in our dataset that the numbers of fake transactions of over 90% sellers are zeros, which suggests that the data is imbalanced. Figure 6.1 shows the distribution of the number of fake transactions of $10,000$ products that are randomly sampled from our dataset. In order to reduce the imbalance of the data, we divide all products into two classes, where the negative class contains products that have zero fake transactions and the positive class contains products that have non-zero fake transactions. We randomly sample $50,000$ products from each class to form the training data and sample $20,000$ products from each class to form the test data. We normalize the training and test data by scaling each feature to a unit norm and test the performance of four popular regression models: linear regression (LR), Ridge regression (RR), LASSO, elastic net (EN) and deep neural network (DNN). The DNN has two hidden layers with 100 neurons with a relu activation function performed on the outputs of the hidden layers. All other parameters of the four regression models are set by default in Scikit-learn [92]. We evaluate the performance of the regression models by the normalized Mean Square Error (nMSE) on the test set. Table 6.2 shows the performance of the four regression models. We can see that DNN has the lowest nMSE, while LR and RR are slightly worse than DNN but significantly outperform LASSO and EN.

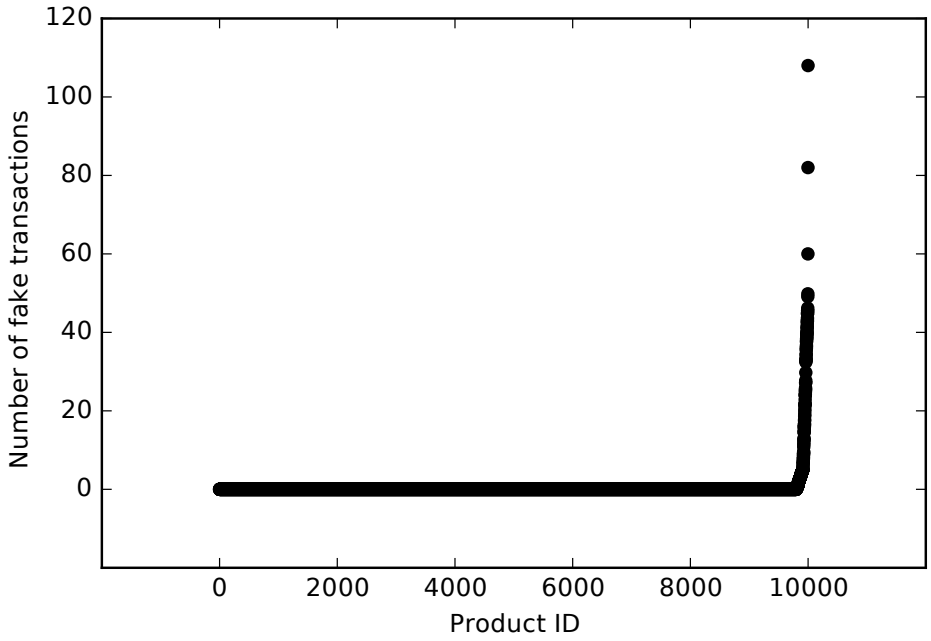


Figure 6.1: Distribution of the number of fake transactions.

We choose LR to simulate the environment for our experiment in Section 6.4 because it has comparable performance with DNN but with significantly lower computational cost. We denote by $\Delta : \mathbf{v}_i^t \rightarrow f_i^{t+1}$ the seller's behavior model, which can be represented by a weight vector $\mathbf{u}^*$ and bias α^* that satisfy

$$(\mathbf{u}^*, \alpha^*) = \arg\min_{\mathbf{u},\alpha} \sum_i (\mathbf{u}^\top \mathbf{v}_i^t + \alpha - f_i^{t+1})^2.$$

6.3 Optimizing via Deep Reinforcement Learning

In this section, we formulate the platform's decision making as a continuous state and continuous action MDP. We sample seller data from our real-world dataset to form the initial state and simulate the environment using the seller behavior model learned in Section 6.2. Then, we propose a novel algorithm based on the DDPG framework to solve the MDP.

6.3.1 MDP Formulation

The platform's optimization problem can be formulated as an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. A state $\mathbf{s}^t = (\mathbf{v}_1^t, ..., \mathbf{v}_n^t) \in \mathcal{S}$ represents all sellers' feature vectors at time t. An action $a^t \in \mathcal{A}$ represents the score function σ^{t+1} the platform uses at time $t+1$. We assume that the score function is linear with respect to the sellers' feature vectors. In other words, the score of seller i at time $t+1$ can be computed by $\sigma^{t+1}(\mathbf{v}_i^t) = {\mathbf{v}_i^t}^\top \mathbf{w}^{t+1} + \beta^{t+1}$, where $\mathbf{w}^{t+1}$ is the weight vector and β^{t+1} is the bias at time $t+1$. Then, the platform's action at time t can be represented as $\mathbf{a}^t = (\mathbf{w}^{t+1}, \beta^{t+1})$. The transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ are determined by the environment.

Specifically, given all sellers' feature vectors $(\mathbf{v}_1^t, ..., \mathbf{v}_n^t)$, the score function σ^{t+1}, the platform places the sellers in the n slots in the descending order of their scores. We assume that the click-through rates descend with the order of the slots, which can be predicted based on the historical data [93]. If there are a total of m^t buyer exposures at time t and the product i is placed at the j-th slot, the impressions that product i gets can be calculated as

$$b_i^t = m^t \cdot ctr_j^t.$$

There are many existing approaches for conversion rate prediction [94, 95]. For simplicity, we assume that the conversion rate $cvr_i^{t+1} = \max\{0, \mathcal{N}(cvr_i^t, 0.1)\}$, where $\mathcal{N}(cvr_i^t, 0.1)$ is a Gaussian distribution whose mean value is the conversion rate at time t and the variance is 0.1. Since the price of a product usually remains stable except promotion activities, we simulate the price at time $t+1$ as $price_i^{t+1} = \max\{0, \mathcal{N}(price_i^t, 0.1)\}$. Given $\mathbf{v}_i^t$, the number of fake transactions at time $t+1$ can be predicted using the learned seller behavior model, i.e., $f_i^{t+1} = \Delta(\mathbf{v}_i^t)$. One can refer to Section 6.2 for details of learning seller behavior model. The number of real transactions of product i at time $t+1$ can be calculated as $r_i^{t+1} = b_i^{t+1} \cdot cvr_i^{t+1} - f_i^{t+1}$. As discussed in Section 6.1, the reward function of the platform

is defined as the weighted sum of both positive GMV and negative GMV:

$$\mathcal{R}(\mathbf{s}^t, \mathbf{a}^t) = \frac{1}{n}\sum_{i=1}^{n}(r_i^{t+1} - \lambda f_i^{t+1}) \cdot price_i^{t+1}, \lambda \geq 0.$$

We denote by $\pi : \mathcal{S} \rightarrow \mathcal{A}$ the policy function of the platform. The platform seeks an optimal policy π^* that maximizes its accumulated reward in a period of time $\{1, ..., T\}$:

$$\pi^* \in \arg\max_{\pi} \sum_{t=1}^{T} \mathcal{R}(\mathbf{s}^t, \pi(\mathbf{s}^t).$$

6.3.2 Solving the MDP

The MDP $\mathcal{M}$ cannot be solved exactly since state space $\mathcal{S}$ and the action space $\mathcal{A}$ are continuous and the transition function $\mathcal{T}$ does not have an explicit form. We consider our problem as a continuous control problem and resort to deep reinforcement learning to optimize the platform's policy function. Deep deterministic policy gradient (DDPG) is a reinforcement learning algorithm that has been successfully applied to many continuous control problems [31]. DDPG is a policy gradient algorithm that uses a stochastic behavior policy for action exploration and estimates a deterministic policy. DDPG is also an actor-critic algorithm where the actor and the critic are represented by deep neural networks. The input of the actor network is the current state, and the output is a real value representing an action chosen from a continuous action space. The input of the critic network is the current state and the action given by the actor network and the output is the estimated Q-value of the state-action pair. The update rule of the actor network is given by the deterministic policy gradient theorem [96] and the critic network is updated based on the temporal-difference error computed using a target network.

Algorithm 4: Deep Deterministic Policy Gradient with Action Norm Penalty (DDPG-ANP)

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor network $\gamma(s|\theta^\gamma)$ with weights θ^Q and θ^γ.
Initialize target networks $Q'(s, a|\theta^{Q'})$ and $\gamma'(s|\theta^{\gamma'})$ with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\gamma'} \leftarrow \theta^\gamma$.
Initialize the replay buffer.
for *episode*=$1, ..., M$ **do**
    Initialize the state at $t = 1$: $\mathbf{s}^1 = (\mathbf{v}_1^1, ..., \mathbf{v}_n^1)$.
    for *t=1,...,T* **do**
        Determines an action $\mathbf{a}^t = \gamma(\mathbf{s}^t|\theta^\gamma) + \Phi^t$, where Φ^t is a Gaussian noise for exploration.
        Execute $\mathbf{a}^t$ and receive a reward computed by the re-engineered reward function $\widetilde{\mathcal{R}}(\mathbf{s}^t, \mathbf{a}^t)$ and a new state $\mathbf{s}^{t+1}$.
        Store n experiences $(s_i^t, a_i^t, r_i^t, s_i^{t+1})_{i=1,...,n}$ in replay buffer.
        Sample a minibatch of N experiences $(s^j, a^j, r^j, s^{j+1})_{j=1,...,N}$ from replay buffer.
        Set $y^j = r^j + \eta Q'(s^{j+1}, \gamma'(s^{j+1}|\theta^{\gamma'})|\theta^{Q'})$.
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_{j=1}^{N}(y^j - Q(s^j, a^j|\theta^Q))^2$
        Update actor using the sampled policy gradient:
        $\nabla_{\theta^\gamma} J \approx \frac{1}{N}\sum_{j=1}^{N} \nabla_a Q(s, a|\theta^Q)|_{s=s^j, a=\gamma(s^j)} \nabla_{\theta^\gamma}\gamma(s|\theta^\gamma)|_{s^j}$
        Update the target networks:
        $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
        $\theta^{\gamma'} \leftarrow \tau\theta^\gamma + (1-\tau)\theta^{\gamma'}$
    end
end

Unfortunately, DDPG performs very poor in solving our problem. Although the action space is continuous, the space of ranking results of the products is discrete. Given an arbitrary ranking result, there is an associated unbounded subspace of the action space. For example, if the weights and bias $(\mathbf{w}, b)$ realize one ranking result, then for any $\rho > 0$, $(\rho\mathbf{w}, \rho b)$ realize the same ranking result because the order of the scores of products remains the same. When the agent explores in an unbounded subspace, it receives the same reward since the reward $\mathcal{R}(\mathbf{s}^t, \mathbf{a}^t)$ is uniquely determined by the ranking result if we consider the click-through rate, the conversion rate and the price as constants. As a result, the agent does not get any informative reward signals when exploring in the subspace. In our experiments,

we found that the weights $\mathbf{w}, \beta$ usually go to infinity during the learning process, which suggests that the policy is trapped into some local optimal ranking results.

One approach to address this issue of unbounded action space is to impose an upper bound on the platform's action space. Although we can project the original action to the bounded action space and execute the projected action, there would be a bias on the policy gradients as is discussed in [97]. We address this issue by reward shaping, which is a method for engineering a reward function in order to provide more frequent feedback on appropriate behaviors. Specifically, we add an action norm penalty to the reward function:

$$\widetilde{\mathcal{R}}(\mathbf{s}^t, \mathbf{a}^t) = \frac{1}{n} \sum_{i=1}^{n} (r_i^{t+1} - \lambda f_i^{t+1}) \cdot price_i^{t+1} - \delta ||\mathbf{a}^t||_2,$$

where $||\mathbf{a}^t||_2$ is the ℓ_2 norm on the action $\mathbf{a}^t$ and $\delta \geq 0$ is its weight in the reward function. There are two advantages: on one hand, it avoids getting an unbounded action since the agent gets a low reward if the norm of action is large; on the other hand, it provides informative feedback to the agent when exploring in the subspace of the action space that is associated with one ranking result. We refer to our algorithm the Deep Deterministic Policy Gradient with Action Norm Penalty (DDPG-ANP), which is shown in Algorithm 4.

6.4 Experimental Results

We build a simulator using the seller behavior model described in Section 6.2 and the environmental dynamics described in Section 6.3.1. Then we evaluate our algorithm DDPG-ANP with several basedlines. First, we evaluate the scalability of DDPG-ANP compared with an existing work that directly apply DDPG to optimize the platform's impression allocation. Then, we evaluate the solution quality of DDPG-ANP compared with DDPG and a greedy impression allocation strategy. All computations were performed on a 64-bit PC with

8 GB RAM and a quad-core 3.20 GHz CPU. Experimental results show that our approach outperforms both baselines in terms of scalability and solution quality.

6.4.1 Scalability Evaluation

There is an existing work that formulates the platform's impression allocation problem as an MDP and directly uses DDPG to solve the MDP [30]. They represent the platform's action as an n-dimensional vector where i-th element represents the number of buyer impressions seller i gets. When the number of sellers is large, the platform has a high-dimensional action space and the scalability of the algorithm becomes a great challenge. In order to evaluate the scalability of DDPG with respect to the dimensionality of the action space, we reformulate the action space $\mathcal{A}$ of the MDP $\mathcal{M}$ defined in Section 6.3. Specifically, an action $\mathbf{a}^t = (b_1^{t+1}, ..., b_n^{t+1}) \in \mathcal{A}$ at time t is represented by the impression allocation of all sellers at time $t+1$. The state transitions and the reward function are accordingly modified based on the reformulated action space.

In the implementation of DDPG, both the actor network and the critic network are four-layer fully connected neural networks, where each of the two hidden layers consists of 100 neurons and a ReLU activation function is applied on the outputs of the hidden layers. A softmax function is applied to the output layer of the actor network in order to bound the total number of impressions. The input of the actor network is a tensor of shape $(n, 6)$ representing feature vectors of n sellers and the output is an n-dimensional action representing the impression allocation of the n sellers. The input the of critic network is a tensor of shape $(n, 6, n)$ representing the state-action pair and the output is the estimated Q-value of the state-action pair. We set the replay buffer size to 10^5, the batch size to 50 and the learning rate to 10^{-5} in the training of both actor and critic networks. We set the weight λ in $\mathcal{R}$ to 0.1. In the implementation of DDPG-ANP, we remove the softmax function at the output layer of

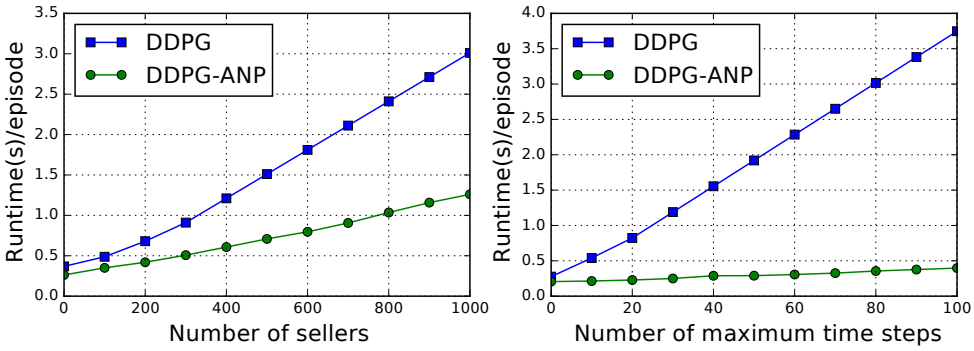


Figure 6.2: Scalability evaluation of DDPG and DDPG-ANP.

the actor network and set the weight δ of the action norm in $\widetilde{\mathcal{R}}$ to 0.01. All other parameters are the same as in the implementation of DDPG.

We evaluate the scalability of DDPG and DDPG-ANP with respect to the number of sellers and the number of maximum time steps T using the average runtime per episode as the evaluation metric. For each set of experiments, we run the algorithm for $1,000$ episodes and calculate an average runtime (seconds). Figure 6.2 shows the scalability of DDPG and DDPG-ANP. We set the maximum time step $T = 10$ in the left figure of Figure 6.2. We can see that the runtime of DDPG-ANP is lower than the half of the runtime of DDPG. We set the number of sellers $n = 100$ in the right figure of Figure 6.2. We can see that the run time of DDPG increases drastically while the runtime of DDPG-ANP increases slightly. This is because the actor network in DDPG has significantly more parameters and training time due to the high-dimensional action space compared with DDPG-ANP, which has a fixed dimension of the action space. Since the agent updates its actor network at every time step, the difference between the training times of DDPG and DDPG-ANP becomes very significant.

6.4.2 Solution Quality Evaluation

In this section, we evaluate DDPG, DDPG-ANP and a greedy impression allocation algorithm using the platform's accumulated reward in 10 time steps as metric. Since each time step represents 3 days, the accumulated reward represents the platform's utility in one month. The click-though rates of the slots are randomly sampled between $[0, 1]$ and ranked in a descending order. We use this simple setting since the click-through rates of slots are not important in our model as long as they are ranked in a descending order. The simulation of state transitions are described in Section **??**, where the number of fake transactions are predicted using the linear regression model described in Section6.2. The implementation of DDPG is similar to that in the last section except that the action is modeled as the platform's score function $\sigma^t = (\mathbf{w}^t, \beta^t)$, which is unbounded since $\mathbf{w}^t$ and β^t can be arbitrarily large. We also compare our results with a greedy impression allocation algorithm described as follows.

Greedy allocation: At each time step t, the platform displays the sellers at the slots in a descending order according to their numbers of real transactions r_i^{t-1} at the last time step. In other words, the platform considers only the number of real transactions as the ranking factor and ignores the number of fake transactions.

We did four sets of experiments with respect to different parameter settings to evaluate our algorithm. Figure 6.3 shows the results of the four sets of experiments. In the first two sets of experiments, we randomly sample 100 sellers from our dataset to form the initial state and randomly sample $1,000$ sellers for the last two sets of experiments. We froze learning every 100 training episodes to evaluate the learned policies across 50 episodes and plot the average platform's utility. As introduced in Section 6.3, λ represents the weight of the num-

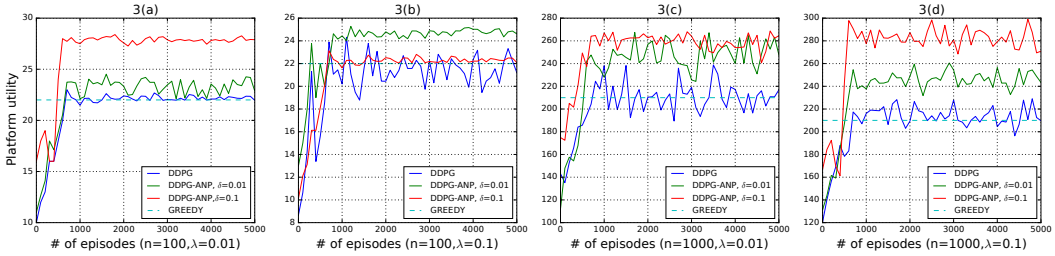


Figure 6.3: Learning curves of DDPG and DDPG-ANP with different parameter settings.

ber of fake transactions and δ represents the weight of action norm penalty. The settings of λ and δ are shown in Figure 6.3. From Figure 3(a) we can see that the platform's utility rapidly increases after 500 episodes and the policy learned by DDPG-ANP outperforms both baselines. Specifically, we can see that DDPG actually learns a sub-optimal policy which is clearly worse than that of DDPG-ANP. In the case of Figure 3(b), the policy learned by DDPG performs even worse than the greedy algorithm.

In our experiments, we found that the actions (values of $\mathbf{w}^t$ and β^t) outputted by the actor network of DDPG usually reach about $10,000$ while the actions outputted by the actor network of DDPG-ANP remains in the range $[-100, 100]$, although there is no imposed bound on the action space. We also found that the parameter δ can significantly influence the performance of DDPG-ANP. In our experiments, the values of δ are set empirically. Through the comparison of Figure 3(a) and Figure 3(b) and the comparison of Figure 3(c) and Figure 3(d) we can see that the platform's utility goes down if the weight of the number of fake transactions increases.

6.5 Chapter Summary

In this work, we study the problem of combating fraudulent sellers in e-commerce through a mechanism design approach. We focus on improving the impression allocation mechanism using deep reinforcement learning with consideration of both real and fake transactions. We

first learn a seller behavior model from real-world data to predict the number of fake transactions that the sellers intend to make. Then, we formulate the platform's decision making problem as an MDP with continuous state and action spaces. We simulate an impression allocation environment in e-commerce using the seller behavior model learned from real-world data. We propose a deep reinforcement learning algorithm DDPG-ANP based on the framework of DDPG for solving the MDP. DDPG-ANP incorporates the action norm penalty in the agent's reward function to facilitate learning. Experimental results show that DDPG-ANP significantly outperforms DDPG and heuristic approaches in terms of scalability and solution quality.

Chapter 7

Conclusion and Future Work

7.1 Conclusions

Along with the success and progress of machine learning technologies, the requirement of security in machine learning systems becomes increasingly urgent. Although machine learning technologies can already handle many complex tasks that we would have not imagined before, we are still far away from fully understanding their vulnerabilities, which could potentially be exploited by adversaries. Combating adversaries in machine learning systems is a complex and challenging task, which involves multidisciplinary techniques including game theory, optimization, human behavior studies and reinforcement learning. This thesis investigates several important problems in the area of adversarial machine learning in order to deepen the understanding of the vulnerabilities of machine learning. Moreover, this thesis studies two real-world problems of combating adversaries in machine learning systems.

The first contribution of this thesis is an efficient label contamination attack algorithm, which demonstrates that the adversaries could significantly bias the learning model by flipping a small proportion of labels of training data. It also demonstrates that the adversaries

could attack a black-box learning model by designing attacks on several substitute models and transferring the attacks to the target model. Based on the observation of transferability, we found that linear learning models are better substitute models than nonlinear ones.

The second contribution of this thesis is generalizing the data poisoning attacks from single-task learning models to multi-task learning models. We formulate the problem of computing optimal poisoning attacks on MTRL as a bilevel problem and propose an efficient algorithm called PATOM for computing optimal attack strategies. PATOM leverages the optimality conditions of the subproblem of MTRL to compute the implicit gradients of the upper level objective function. During experiments we found that MTRL models are very sensitive to both directly attacks and indirect attacks. We also found that the tasks being attacked are always strongly correlated, which provides a clue for defending against such attacks.

The third contribution of this thesis is the study of spear phishing attacks and their defenses. We model the spear phishing attack scenarios as s Stachelberg games played by an attacker and a defender. For the single-credential scenario, we demonstrate that the optimal defense strategy can be found by solving a binary combinatorial optimization problem called PEDS. For the multiple-credential scenario, we formulate it as a bilevel optimization problem for finding the optimal defense strategy and then reduce it to a single level optimization problem called PEMS using complementary slackness conditions. Experimental results show that both PEDS and PEMS lead to significant higher defender utilities than two existing benchmarks in different parameter settings. Also, both PEDS and PEMS are more robust than the existing benchmarks considering uncertainties.

The fourth contribution of this thesis is a novel reinforcement learning based impression allocation mechanism, which aims to combat fraudulent sellers and maintain the real transactions in e-commerce. First, we build a simulator to simulate the online e-commerce environment by learning seller behavior model from historical data. Then, we formulate the

platform's impression allocation problem as a continuous Markov Decision Process (MDP) with unbounded action space. In order to make the action executable in practice and facilitate learning, we propose a novel deep reinforcement learning algorithm DDPG-ANP that introduces an action norm penalty to the reward function. Experimental results show that our algorithm significantly outperforms existing baselines in terms of scalability and solution quality.

Overall, this thesis investigates two important type of adversarial machine learning problems and provides efficient attack algorithms. This thesis also studies real-world adversaries and develop effective defense strategies from the perspective of practical use.

7.2 Future Directions

The ultimate goal of adversarial machine learning is to develop defense strategies based on the analysis of the attacker's strategic behavior. Regarding the label contamination attacks, there are two main difficulties in developing defense strategies. First, the attacker's goal is hard to estimate. For example, the attacker can perform integrity attack, availability attack, or even a hybrid attack. Since the attacker's strategy is optimized with respect to his goal, it is difficult for the learner to accurately estimate the attacker strategy without knowing his goal. Second, most existing defense methods (e.g., robust learning) require a set of clean data (true labels), and future data will be judged based on the metrics developed using the clean data. However, in practice, it is often expensive to obtain enough true labels, especially when domain experts are employed.

The analysis of data poisoning attacks provides opportunities to develop alternative defense strategies in the future. Here we discuss two possible future directions to develop defense strategies. 1) Discovering the characteristics of poisoned data and identify the data that are most likely to be attacked. For example, from Figure 3.2 we can see that the at-

tacked data basically form two clusters (one cluster with big blue points and the other cluster with big red points). Therefore, if we have successfully identified an attacked point, we can look into its adjacent points. In addition, most attacked points are extreme points, which indicates that the extreme points are more likely to be attacked than those near centroid. 2) Game-theoretic modeling. Adversarial machine learning can be viewed as a game between the learner and the attacker. [98] model the test-set attack problem as a Stackelberg game [99]. They assume that the learner has a set of *explicit defense actions*, such as verifying data with third parties, and try to compute the optimal defense action. However, few works apply game-theoretic analysis to poisoning attacks. Although the study of poisoning attacks provide a framework to model the attacker behavior, the learner's defense actions have not been considered in the traditional poisoning attack setting. If we consider defense actions, such as giving penalty on detected attacker behavior, we might be able to develop realistic game models and more secure learning algorithms.

Regarding the security of MTL, in future work, we will consider two classes of potential defense strategies for protecting MTL: data sanitization and improving the robustness of MTL. First, as shown in our experiments, the tasks under attack show a strong correlation with 30% data injected. Therefore, the machine learner can examine the data from tasks that form strong local correlations, perhaps through human verifications. Moreover, once a task is demonstrated to be malicious, the learner can examine the tasks that strongly correlate to it, which will significantly reduce the learner's effort in examining the data. Second, improving the robustness of MTL could also be an effective approach to defend against data poisoning attacks. MTL exploits the task relatedness to improve the performance of individual tasks, where such relatedness can also be exploited by the attacker to launch indirect attacks. A possible approach to improve the robustness of MTL is to differentiate the normal relatedness and the malicious task relatedness, so that we can preserve the helpful relatedness and reduce the harmful relatedness during learning.

Chapter 8

Appendix

8.1 Proof of Lemma 1

Lemma 1. $u \in \Gamma(\pi_{\mathbf{x}})$ *if and only if* $x_u a_u k_u L > c_u$.

Proof. First we show that $V^*(s) \geq 0$ ($\forall s \in \mathcal{S}$):

$$V^*(s) = \arg\max_{a \in \mathcal{A}^s} Q(s, a) \geq Q(s, a = stop) = 0.$$

If direction: Consider state $s = \{u\}$, we have

$$\begin{aligned} V^*(s) &= (1 - x_u)(V^*(s) - c_u) + a_u x_u k_u (L - c_u) \\ &\quad + x_u(1 - a_u k_u)(V^*(s^{-u}) - c_u) \\ &\geq (1 - x_u)(V^*(s) - c_u) + a_u x_u k_u (L - c_u) \\ &\quad + x_u(1 - a_u k_u)(0 - c_u). \end{aligned}$$

If $x_u a_u k_u L > c_u$, then $V^*(s) > 0$, which means that s is a reachable state and the optimal

action at state s is to attack user u instead of stop attacking. Therefore, u belongs to the potential attack set $\Gamma(\pi_{\mathbf{x}})$.

Only if direction: First, consider state s and s^{-u}. If we restrict the attacker's policy so that he never attacks u, then s and s^{-u} are indifferent so that $V^*(s)=V^*(s^{-u})$. Without the restriction, we have $V^*(s)\geq V^*(s^{-u})$. In other words, adding a user to a state does not decrease its value. We prove that if $\pi_{\mathbf{x}}(s)=u$, then $x_u > \frac{c_u}{La_uk_u}$. By definition we have:

$$\begin{aligned} V^*(s) = (1-x_u)(V^*(s)-c_u) + a_u x_u k_u (L-c_u) \\ + x_u(1-a_uk_u)(V^*(s^{-u})-c_u). \end{aligned}$$

By adjusting the terms we have:

$$V^*(s) = -\frac{c_u}{a_u x_u} + Lk_u + (1-k_u)V^*(s^{-u}).$$

Since $V^*(s)\geq V^*(s^{-u})$, then:

$$-\frac{c_u}{a_u x_u} + Lk_u \geq k_u V^*(s^{-u}) \geq 0$$

Note that if $-\frac{c_u}{a_u x_u}+Lk_u=0$, we have $V^*(s)=V^*(s^{-u})=0$ and $s=\{u\}$. Due to the setting that the attacker always prefers stopping attack rather than launching another attack, we have $\pi_{\mathbf{x}}(s)=0$, which contradicts the assumption that $\pi_{\mathbf{x}}(s)=u$. Therefore, $-\frac{c_u}{a_u x_u} + Lk_u > 0$, equivalently, $x_u > \frac{c_u}{La_uk_u}$. □

8.2 Proof of Lemma 2

Lemma 2.

$$\theta(\mathbf{x}, \pi_{\mathbf{x}}) = \begin{cases} 1 - \prod_{u \in \Gamma(\pi_{\mathbf{x}})}(1 - a_u k_u), & \textit{if } \Gamma(\pi_{\mathbf{x}}) \neq \emptyset \\ 0, & \textit{if } \Gamma(\pi_{\mathbf{x}}) = \emptyset \end{cases}.$$

Proof. If $\Gamma(\pi_{\mathbf{x}}) = \emptyset$, meaning that the attacker stops attacking at the initial state s_0, therefore the probability that the credential accessed is 0. Otherwise, we write the reachable states set as $\Delta(\pi_{\mathbf{x}}) = \{s_0, s_1, ..., s_r\} \cup \{s^n, s^y\}$. We denote by $M_{\Delta(\pi_{\mathbf{x}})}$ the transition probability matrix, whose entry M_{ij} represents the probability that state s_i transitions to s_j under policy $\pi_{\mathbf{x}}$ (WLOG, we define $s_{r+1}{=}s^n$ and $s_{r+2}{=}s^y$). There are two cases for s_r: (1) $\pi_{\mathbf{x}}(s_r) = u \in \mathcal{A}^{s_r}$ and (2) $\pi_{\mathbf{x}}(s_r) = stop$.

If case (1), s_r could transition to itself, s^n or s^y. Hence $M_{\Delta(\pi_{\mathbf{x}})}$ has the form like (denote $d_i{=}a_{u^i}k_{u^i}$ and $x_i = x_{u^i}$):

$$\begin{bmatrix} 1-x_0 & x_0(1-d_0) & & & & & d_0x_0 \\ & 1-x_1 & x_1(1-d_1) & & & & d_1x_1 \\ & & \ddots & \ddots & & & \vdots \\ & & & 1-x_r & x_r(1-d_r) & & d_rx_r \\ & & & & 1 & & \\ & & & & & & 1 \end{bmatrix}$$

Precisely, $M_{\Delta(\pi_{\mathbf{x}})}$ can be represented as:

$$M_{\Delta(\pi_{\mathbf{x}})} = \left[\begin{array}{c:c} A & B \\ \hdashline 0 & I_2 \end{array}\right]$$

where A is $r{+}1$ dimensional square matrix, I_2 is 2 dimensional unit diagonal matrix and B

is $(r+1) \times 2$ matrix. We introduce a $(r+1) \times 2$ matrix E:

$$E = FB, \text{ where } F = (I_{r+1} - A)^{-1}$$

Note that s^n and s^y are absorbing states. According to the properties of absorbing Markov chain, s_0 will eventually end in state s^n or s^y with probability E_{11} and E_{12} respectively, and $E_{11}+E_{12}=1$. Therefore, the probability of losing the credential is equal to the probability that the attacker eventually ends in state s^y, i.e., $\theta(\mathbf{x}, \pi_{\mathbf{x}})=E_{12}$. We can directly calculate E_{11} based on the rules of matrix calculation:

$$\begin{aligned} E_{11} &= \sum_{i=1}^{r+1} F_{1i} B_{i1} \\ &= F_{1,r+1} B_{r+1,1} \\ &= \frac{\prod_{i=0}^{r-1}(1-d_i)}{x_r} x_r (1-d_r) \\ &= \prod_{i=0}^{r} (1-d_i) \\ &= \prod_{u \in \Gamma(\pi_{\mathbf{x}})} (1 - a_u k_u) \end{aligned}$$

Then $E_{12} = 1 - E_{11} = 1 - \prod_{u \in \Gamma(\pi_{\mathbf{x}})}(1 - a_u k_u)$.

If case (2), s_r transitions to s^n with probability 1. Thus $M_{\Delta(\pi_{\mathbf{x}})}$ has the form like (d_i=$a_{u^i} k_{u^i}$ and $x_i = x_{u^i}$):

$$\begin{bmatrix} 1-x_0 & x_0(1-d_0) & & & & & x_0k_0 \\ & 1-x_1 & x_1(1-d_1) & & & & x_1k_1 \\ & & \ddots & \ddots & & & \vdots \\ & & & 1-x_{r-1} & x_{r-1}(1-d_{r-1}) & 0 & x_{r-1}d_{r-1} \\ & & & & & 1 & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

Similarly,

$$\begin{aligned} E_{11} &= \sum_{i=1}^{r+1} F_{1i}B_{i1} \\ &= F_{1,r+1} \\ &= \prod_{i=0}^{r-1}(1-d_i) \\ &= \prod_{u\in\Gamma(\pi_{\mathbf{x}})}(1-a_uk_u) \end{aligned}$$

Then, we still have $E_{12} = 1 - E_{11} = 1 - \prod_{u\in\Gamma(\pi_{\mathbf{x}})}(1-a_uk_u)$. □

8.3 Proof of Theorem 1

Theorem 3. *The defender's expected utility remains the same no matter how the attacker breaks ties, i.e., choosing any optimal policy.*

Proof. Recall that in single-credential case the defender's utility function is

$$P_d(\mathbf{x}, \pi_{\mathbf{x}}) = -\rho^{\mathcal{T}}\theta(\mathbf{x}, \pi_{\mathbf{x}})L - \sum_{u \in U} \Lambda(x_u).$$

Based on the result of Lemma 1, $\Gamma(\pi_{\mathbf{x}})$ can be represented as $\{u \in U | x_u > \frac{c_u}{La_u k_u}\}$, then $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ can be represented as

$$\theta(\mathbf{x}, \pi_{\mathbf{x}}) = 1 - \prod_{u \in \{u' \in U | x_{u'} > \frac{c_{u'}}{La_{u'} k_{u'}}\}} (1 - k_u).$$

For any other optimal policy $\pi'_{\mathbf{x}}$, we have

$$\theta(\mathbf{x}, \pi'_{\mathbf{x}}) = 1 - \prod_{u \in \{u' \in U | x_{u'} > \frac{c_{u'}}{La_{u'} k_{u'}}\}} (1 - k_u).$$

Note that $\theta(\mathbf{x}, \pi_{\mathbf{x}}) = \theta(\mathbf{x}, \pi_{\mathbf{x}})'$, which indicates that the defender's expected utility will be the same when the attacker chooses any other optimal policy. □

8.4 Proof of Theorem 2

Theorem 4. *x_u^1 is an arbitrary point in $\arg\min_{x \in [0, \frac{c_u}{La_u k_u}]} \Lambda_u$ and x_u^2 is an arbitrary point in $\arg\min_{x \in (\frac{c_u}{La_u k_u}, 1]} \Lambda_u$.*

Proof. Recall that in single-credential case the defender's utility function is

$$P_d(\mathbf{x}, \pi_{\mathbf{x}}) = -\rho^{\mathcal{T}}\theta(\mathbf{x}, \pi_{\mathbf{x}})L - \sum_{u \in U} \Lambda_u(x_u).$$

Consider a user u, given all values of $x_{u'}$ $(u' \in U \setminus \{u\})$, $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ is constant for any $x_u \in [0, \frac{c_u}{La_u k_u}]$ since the potential attack set $\Gamma(\pi_{\mathbf{x}})$ remains the same when x_u varies among

$[0, \frac{c_u}{La_u k_u}]$. Therefore, any point in $\arg\min_{x \in [0, \frac{c_u}{La_u k_u}]} \Lambda_u$ maximizes $P_d(\mathbf{x}, \pi_{\mathbf{x}})$. Similarly, $\theta(\mathbf{x}, \pi_{\mathbf{x}})$ is constant for any $x_u \in (\frac{c_u}{La_u k_u}, 1]$. Therefore, any points in $\arg\min_{x \in (\frac{c_u}{La_u k_u}, 1]} \Lambda_u$ maximizes $P_d(\mathbf{x}, \pi_{\mathbf{x}})$. □

Bibliography

[1] Bo Li and Yevgeniy Vorobeychik. Feature cross-substitution in adversarial classification. In *Prodeedings of the 28th Advances in Neural Information Processing Systems*, pages 2087–2095, 2014.

[2] Mengchen Zhao, Zhao Li, Bo An, Haifeng Lu, Yifan Yang, and Chen Chu. Impression allocation for combating fraud in e-commerce via deep reinforcement learning with action norm penalty. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3940–3946, 2018.

[3] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[4] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Prodeedings of the 30th Annual Conference on Neural Information Processing Systems*, pages 1885–1893, 2016.

[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. *The 3rd Asian Conference on Machine Learning*, 20:97–112, 2011.

[6] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 870–875, 2012.

[7] Patrick PK Chan, Zhi-Min He, Hongjiang Li, and Chien-Chang Hsu. Data sanitization against adversarial label contamination based on data complexity. *International Journal of Machine Learning and Cybernetics*, pages 1–14, 2017.

[8] Sergiy Fefilatyev, Matthew Shreve, Kurt Kramer, Lawrence Hall, Dmitry Goldgof, Rangachar Kasturi, Kendra Daly, Andrew Remsen, and Horst Bunke. Label-noise reduction with support vector machines. In *Proceedings of the 21st International Conference on Pattern Recognition*, pages 3504–3508, 2012.

[9] George S Kimeldorf and Grace Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[10] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[11] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 733–742, 2010.

[12] Sulin Liu, Sinno Jialin Pan, and Qirong Ho. Distributed multi-task relationship learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 937–946, 2017.

[13] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1807–1814, 2012.

[14] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1452–1458, 2016.

[15] TrendLabs. Spear-phishing email: Most favored APT attack bait. Technical report, Trend Micro, 2012.

[16] Kim Zetter. Researchers uncover RSA phishing attack, hiding in plain sight. http://www.wired.com/2011/08/how-rsa-got-hacked/, 2011.

[17] André Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paaß, and Siehyun Strobel. New filtering approaches for phishing email. *Journal of computer security*, 18(1):7–35, 2010.

[18] Steve Sheng, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Cranor, and Jason Hong. Improving phishing countermeasures: An analysis of expert interviews. In *Proceedings of the 4th APWG eCrime Researchers Summit*, pages 1–15, 2009.

[19] Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal personalized filtering against spear-phishing attacks. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 958–964, 2015.

[20] Rohit Varma. Combating Aurora. Technical report, McAfee Labs, 2010.

[21] Gavin Watson, Andrew Mason, and Richard Ackroyd. *Social Engineering Penetration Testing*, chapter 4, pages 71–74. Elsevier, 2014.

[22] Renxin Mao, Zhao Li, and Jinhua Fu. Fraud transaction recognition: A money flow network approach. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1871–1874, 2015.

[23] Mengchen Zhao, Bo An, Wei Gao, and Teng Zhang. Efficient label contamination attacks against black-box learning models. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3945–3951, 2017.

[24] Mengchen Zhao, Bo An, Yaodong Yu, Sulin Liu, and Sinno Jialin Pan. Data poisoning attacks on multi-task relationship learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 2628–2635, 2018.

[25] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–904, 2016.

[26] Chang Xu and Jie Zhang. Towards collusive fraud detection in online reviews. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1051–1056, 2015.

[27] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 939–948, 2010.

[28] Pingzhong Tang. Reinforcement mechanism design. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 5146–5150, 2017.

[29] Qingpeng Cai, Aris Filos-Ratsikas, Pingzhong Tang, and Yiwei Zhang. Reinforcement mechanism design for e-commerce. In *Proceedings of the 27th International Conference on World Wide Web*, 2018.

[30] Qingpeng Cai, Aris Filos-Ratsikas, Pingzhong Tang, and Yiwei Zhang. Reinforcement mechanism design for fraudulent behaviour in e-commerce. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[31] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez,

Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[32] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pages 43–58, 2011.

[33] Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

[34] Marius Kloft and Pavel Laskov. Online anomaly detection under adversarial impact. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 405–412, 2010.

[35] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. Robust linear regression against training data poisoning. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 91–102. ACM, 2017.

[36] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *Proceedings of the 32th International Conference on Machine Learning*, pages 1689–1698, 2015.

[37] Shike Mei and Xiaojin Zhu. The security of latent dirichlet allocation. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 681–689, 2015.

[38] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the 29st AAAI conference on Artificial intelligence*, pages 2871–2877, 2015.

[39] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, pages 41–48, 2007.

[40] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6: 1817–1853, 2005.

[41] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 489–497, 1996.

[42] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402, 2013.

[43] Bo Li and Yevgeniy Vorobeychik. Evasion-robust classification on binary domains. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4):50, 2018.

[44] Fei Zhang, Patrick PK Chan, Battista Biggio, Daniel S Yeung, and Fabio Roli. Adversarial feature selection against evasion attacks. *IEEE transactions on cybernetics*, 46 (3):766–777, 2016.

[45] Paolo Russu, Ambra Demontis, Battista Biggio, Giorgio Fumera, and Fabio Roli. Secure kernel machines against evasion attacks. In *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, pages 59–69. ACM, 2016.

[46] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[47] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[48] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, pages 372–387, 2016.

[49] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[50] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[51] Jason Hong. The state of phishing attacks. *Communications of the ACM*, 55(1):74–81, 2012.

[52] Ponnurangam Kumaraguru, Yong Rhee, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Protecting people from phishing: the design and evaluation of an embedded training email system. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 905–914. ACM, 2007.

[53] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074. ACM, 2008.

[54] Deanna D Caputo, Shari Lawrence Pfleeger, Jesse D Freeman, and M Eric Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy*, 12(1):28–38, 2014.

[55] Prateek Dewan, Anand Kashyap, and Ponnurangam Kumaraguru. Analyzing social

and stylometric features to identify spear phishing emails. In *APWG Symposium on Electronic Crime Research*, pages 1–13, 2014.

[56] B Brij Gupta, Aakanksha Tewari, Ankit Kumar Jain, and Dharma P Agrawal. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications*, 28(12):3629–3654, 2017.

[57] Aron Laszka, Jian Lou, and Yevgeniy Vorobeychik. Multi-defender strategic filtering against spear-phishing attacks. In *AAAI*, pages 537–543, 2016.

[58] Aron Laszka, Waseem Abbas, S Shankar Sastry, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal thresholds for intrusion detection systems. In *Proceedings of the Symposium and Bootcamp on the Science of Security*, pages 72–81. ACM, 2016.

[59] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. Credit card fraud detection using hidden markov model. *IEEE Transactions on dependable and secure computing*, 5(1):37–48, 2008.

[60] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. Credit card fraud detection using convolutional neural networks. In *International Conference on Neural Information Processing*, pages 483–490, 2016.

[61] Haiqin Weng, Zhao Li, Shouling Ji, Chen Chu, Haifeng Lu, Tianyu Du, and Qinming He. Online e-commerce fraud: a large-scale detection and analysis. In *2018 IEEE 34th International Conference on Data Engineering*, pages 1435–1440, 2018.

[62] Weiran Shen, Binghui Peng, Hanpeng Liu, Michael Zhang, Ruohan Qian, Yan Hong, Zhi Guo, Zongyao Ding, Pengjun Lu, and Pingzhong Tang. Reinforcement mechanism design, with applications to dynamic pricing in sponsored search auctions. *arXiv preprint arXiv:1711.10279*, 2017.

[63] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, pages 1313–1320, 2009.

[64] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*, number 4, page 5, 2007.

[65] George S Kimeldorf and Grace Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[66] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[67] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[68] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(Aug):1871–1874, 2008.

[69] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[70] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, 2004.

[71] Laurent Jacob, Jean-philippe Vert, and Francis R Bach. Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, pages 745–752, 2009.

[72] Tsuyoshi Kato, Hisashi Kashima, Masashi Sugiyama, and Kiyoshi Asai. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems*, pages 737–744, 2008.

[73] David Kahn. Codebreakers: The comprehensive history of secret communication from ancient times to the Internet. *Naval War College Review*, 51(4):153–155, 1998.

[74] Frank L Greitzer, Andrew P Moore, Dawn M Cappelli, Dee H Andrews, Lynn A Carroll, and Thomas D Hull. Combating the insider cyber threat. *IEEE Security & Privacy*, 6(1), 2008.

[75] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647, 2005.

[76] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. Who falls for phish? A demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 373–382, 2010.

[77] Patrick G Kelley. Conducting usable privacy & security studies with amazon's mechanical turk. In *Proceedings of the 6th Symposium on Usable Privacy and Security (SOUPS'10)*, 2010.

[78] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[79] P Deshmukh, M Shelar, and N Kulkarni. Detecting of targeted malicious email. In *IEEE Global Conference on Wireless Computing and Networking (GCWCN'14)*, pages 199–202, 2014.

[80] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

[81] Remco R Bouckaert. Efficient auc learning curve calculation. In *AI 2006: Advances in Artificial Intelligence*, pages 181–191. Springer, 2006.

[82] Paul Hlatky. How does yesware tracking work? http://www.yesware.com/blog/how-does-yesware-tracking-work/, 2015.

[83] KimKwang Raymond Choo. The cyber threat landscape: Challenges and future research directions. *Computers & Security*, 30(8):719–731, 2011.

[84] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41(2):297–327, 2011.

[85] Jiarui Gan, Bo An, and Yevgeniy Vorobeychik. Security games with protection externalities. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 914–920, 2015.

[86] Yue Yin, Haifeng Xu, Jiarui Gan, Bo An, and Albert Xin Jiang. Computing optimal mixed strategies for security games with dynamic payoffs. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 681–687, 2015.

[87] Yue Yin, Bo An, and Manish Jain. Game-theoretic resource allocation for protecting large public events. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, pages 826–834, 2014.

[88] Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

[89] Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

[90] VS Borkar and MK Ghosh. Stochastic differential games: Occupation measure based approach. *Journal of optimization theory and applications*, 73(2):359–385, 1992.

[91] Seth Rosenblatt. Nuclear regulator hacked 3 times in 3 years. http://www.cnet.com/news/nuclear-commission-hacked-3-times-in-3-years/, 2014.

[92] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[93] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50, 2016.

[94] Dirk Van den Poel and Wouter Buckinx. Predicting online-purchasing behaviour. *European Journal of Operational Research*, 166(2):557–575, 2005.

[95] Catarina Sismeiro and Randolph E Bucklin. Modeling purchase behavior at an e-commerce web site: A task-completion approach. *Journal of marketing research*, 41 (3):306–323, 2004.

[96] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395, 2014.

[97] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distri-

bution. In *Proceedings of the International Conference on Machine Learning*, pages 834–843, 2017.

[98] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Explicit defense actions against test-set attacks. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*, pages 1274–1280, 2017.

[99] Bo An, Milind Tambe, and Arunesh Sinha. Stackelberg security games (SSG): Basics and application overview. *Improving Homeland Security Decisions*, 2015.