Learning and Planning Towards AI for Social Good

Zheyuan Ryan Shi

CMU-S3D-23-105

June 2023

Software and Societal Systems Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Societal Computing.

This work was supported in part by the National Science Foundation (NSF) under grants 1850477 and 2046640, the Siebel Foundation, the Army Research Laboratory Cyber Security Aliliance (CRA), and the Defense Advanced Research Projects Agency (DARPA). The views and conclusions are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

Keywords: AI for social good, machine learning, game theory, bandit

To my parents, who always support me for who I am.

Abstract

AI for Social Good (AI4SG) is a research theme that uses and advances AI to improve the well-being of society. We introduce three lines of work that center around learning and planning to address real-world challenges in cybersecurity, food waste and security, and environmental conservation. For cybersecurity, we provide a learning and planning pipeline for generic cyber deception and an algorithm to counter watering-hole attacks. In food waste and food security, we develop a predictive model for the rescue claim status and an online learning and planning algorithm for volunteer engagement through push notifications. We also ran a randomized controlled trial for our algorithm to show significant improvement in the real world. For environmental conservation, we develop a natural language processing-based media content monitoring system to provide early warning of infrastructure projects that might pose harm to conservation efforts. The system leverages active learning and learning with noisy labels algorithms to address challenges in applied learning and planning applications. The tool has been deployed in multiple places around the world, monitoring over 60,000 conservation sites worldwide since February 2022. Distilling lessons learned from these projects, we propose bandit data-driven optimization, the first iterative learning and planning framework to rigorously address the pain points in practical prediction-prescription workflows in lots of social good projects across application domains.

Acknowledgments

I would like to thank my advisor, Fei Fang, for everything throughout this journey. The past five years working with Fei was once in a lifetime. Her guidance, professionalism, and patience have been pivotal as we built stuff from zero to one. I am always grateful to her for encouraging me to stay true to my values, to explore unorthodox approaches, and for her dedication to my growth as a researcher. When I first met Fei years before this PhD, I could not expect going so far together, but apparently, even now is not the end.

I am very grateful to my thesis committee members: Rayid Ghani, Jason Hong, and Milind Tambe. Rayid has been my role model throughout these years. I have learned from Rayid about not just research, but also practical experience and career development, as well as his kindness and generosity. I will make sure to pay it forward. I thank Jason Hong and Milind Tambe for the insightful questions raised during both my proposal and defense. I enjoyed thinking through these questions and discovering new perspectives.

The work in this thesis was made possible by the continued support from our non-profit partners. I would like to thank Leah Lizarondo, Ameesh Kapoor, Anthony Levin-Decanini, Sean Hudson, Jake Tepperman, and everyone else at 412 Food Rescue for our joint effort on food rescue. I would like to thank David J. Patterson, Nirmal Bhagabati, Karun Dewan, Areendran Gopala, Pablo Izquierdo, Debojyoti Mallick, Ambika Sharma, Pooja Shrestha, Johanna Prussmann Uribe, and everyone else at World Wildlife Fund for our project on NewsPanda. I also thank everyone at all the other organizations that we have partnered with.

In the past five years, I have had the privilege of collaborating with many excellent researchers: Steven Wu, Ariel Procaccia, Hemank Lamba, Sridhar Venkatesan, and Aaron Schlenker. I appreciate all their insights and guidance throughout the collaborations. I would also like to thank Kush Varshney, Amulya Yadav, and Bistra Dilkina. I have not directly worked with them, but they have been there at multiple points of my career, and gave me the support that I needed so much. In addition, I want to thank Linda Moreci and Connie Herold for their tireless hard work making life easier for me and many other students at ISR.

I would like to also thank the precious friendships that lightened up these occasionally dark and often chaotic years. Many thanks to my lab mates: Chun Kai Ling, Stephanie Milani, Steven Jecmen, Rex Chen, Zhicheng Zhang, and Yinuo Du; and office mates: Melrose Roderick, Nimo Ni, Daye Nam, Morgan Evans, and Joshua Uyheng. I also thank Junzhi Yan, Xinyuan Ma, Violet Chen, Savannah Tang, Ziye Tang, and Yuyan Wang for sharing the time and laughter together.

Last but not least, I am immensely indebted to my family. My parents, Jing Luo and Wei Shi, have always loved me, believed in me, and supported me unconditionally. None of this would have been possible without you. Thank you so much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial intelligence (AI) is in many ways an important source of change in people's lives in the 21st century. Its impact is drastic and real: Youtube's AI-driven recommendation system would present sports videos for days if one happens to watch a live baseball game on the platform [37]; email writing becomes much faster with machine learning (ML) based auto-completion [26]; many businesses have adopted natural language processing based chatbots as part of their customer services [155]. AI has also greatly advanced human capabilities in complex decision-making processes ranging from determining how to allocate security resources to protect airports [122] to games such as poker [22] and Go [144]. All such tangible and stunning progress suggests that an "AI summer" is happening. As some put it, "AI is the new electricity" [96].

Meanwhile, in the past decade, an emerging theme in the AI research community is the so-called "AI for social good" (AI4SG): researchers aim at developing AI methods and tools to address problems at the societal level and improve the well-being of the society [140]. Over the years, there have been several successful AI4SG projects, such as guiding municipal water pipe replacement [3], protecting wildlife from poaching [46], and spreading HIV prevention information among homeless youth [163]. AI4SG has received recognition from both within and outside the academic community. Major AI conferences have featured various special tracks and workshops dedicated to AI4SG, with over 1000 papers on AI4SG topics formally published. Large companies are expanding their investments on AI4SG initiatives. The need for collaboration among researchers, governments, companies, and non-profit organizations to solve AI4SG problems has never been more widely appreciated.

1.1 Motivation

In this thesis, we focus on learning and planning in three applications domains: cybersecurity, food waste and security, and environmental sustainability. These various applications culminate in an abstracted model on iterative learning and planning.

The world today poses more challenges to cybersecurity than ever before. Despite the ever-improving security measures, malicious attackers work diligently and creatively to outstrip the defense [123]. Against an attacker with previously unseen exploits and abundant resources, the attempt to protect any target is almost surely a lost cause [69]. However, the defender could induce the attacker to attack a less harmful, or even fake, target. This can be seen as a case of deception. Deception has been extensively studied in cybersecurity [67, 71]. Security researchers have proposed many deceptive measures to manipulate a machine's response to these probes [9, 72], which could confound and mislead an attempt to attack. However, there lacks a general game-theoretic framework to rigorously reason about strategic deception. It is also unclear how such game-theoretic cyber deception can be applied to a specific type of cyber attacks. In Chapter 2, we propose a rigorous learning and planning pipeline to compute the optimal cyber deception strategy. In Chapter 3, we study cyber deception in a concrete watering-hole attack setting.

Food waste and food insecurity are two challenges that coexist in many communities. To mitigate the problem, food rescue platforms match excess food with the communities in need, and leverage external volunteers to transport the food. Relying on external volunteers to deliver the food comes with inherent uncertainty. What if no volunteer will claim the rescue? This uncertainty is prevalent in FR operations and it has serious consequences such as lost faith in the program from the donor and recipient organizations. Therefore, it is crucial to the food rescue organizations to be able to predict whether any volunteer is going to claim a given rescue trip, as well as engage with volunteers to raise their claim rate. In Chapter 4, we develop such a predictive model to help the food rescue dispatchers manage their outstanding rescues. In Chapter 5, we develop an online learning and planning recommender system which selectively sends push notifications to volunteers in order to improve volunteer engagement.

Early detection of built infrastructure projects such as roads and railways by conservation nonprofits could shift infrastructure planning towards more environmentally sustainable outcomes. However, information about conservation-related events and infrastructure plans threatening critical habitats is scattered across numerous sources and comes in different forms. NGOs typically learn of such information through word-of-mouth or a handful of news outlets that they check manually. This process is both time-consuming and ineffective, and it can potentially fail to capture critical information in a timely manner, leaving these NGOs out of key conversations during early or ongoing stages of these developments. In Chapter 7, we developed NewsPanda, an NLP toolkit to analyze news and documents describing emerging infrastructure threats to conservation areas.

From our work in the application domains above and several other domains emerges a pain point, which is common across many AI4SG applications. The success of ML often does not translate directly into a satisfactory solution to a real-world AI4SG problem. One obvious reason is supervised learning focuses on prediction, yet real-world problems, by and large, need prescription. The common practice is a two-stage procedure, where after training an ML model, the user makes prescriptive decisions based on some optimization problem parametrized by the prediction output. However, in a typical workflow in many AI4SG projects, the process does not stop here. Using the new data collected under the prescribed intervention, the researcher updates the ML model and recommends a new intervention, so on and so forth, leading to an iterative process. The principles of these steps are often not aligned. Without a rigorous, integrated framework to guide the procedure, this could lead to operation inefficiency, missed expectations, dampened initiatives, and new barriers of mistrust which are not meant to be. In Chapter 8, we propose the first iterative prediction-prescription framework to study this process.

1.2 Thesis Outline and Contributions

- Part I contains our work on using learning and planning for strategic cyber deception. The key question is how we interact with an unknown attacker by learning their behavioral model and design strategies to counter them. While we ground our work in cybersecurity, the same idea applies to public safety, environmental protection, etc.
    - In Chapter 2, to rigorously reason about strategic cyber deception, we propose the Feature Deception Problem framework. We present the first learning and planning pipeline to compute the optimal deception strategy against an unknown attacker with optimality guarantee. This work has been published at GameSec 2020 [138].
    - In Chapter 3, for a specific type of cyberattacks called watering-hole attacks, we propose the social engineering deception model. We devise the CyberTWEAK algorithm to strategically manipulate network packets to mitigate such attacks. The work in this and previous chapters is theoretical in nature. Yet, we also develop a browser extension based on CyberTWEAK which is publicly available on the Chrome Web Store. This work has been published at IAAI 2020 [139].
- Part II contains our work with 412 Food Rescue on using learning and planning to address food waste and security. The key question is how to engage volunteers on such crowdsourcing platforms by learning their activity patterns and optimize for desired outcomes. While we ground our work in food waste and security, the same idea applies to many volunteer-based crowdsourcing platforms.
    - In Chapter 4, we develop a prediction model to predict whether any volunteer will claim a given rescue within a predefined time frame. This model could help the food rescue dispatchers manage their outstanding rescues. In this chapter, we also use a data-driven approach to find the optimal dispatcher intervention and notification scheme. 412 Food Rescue has adopted our recommendation since January 2020. This work has been published at IAAI 2020 [142].
    - Chapter 5 advances from Chapter 4 in that we build a rescue-specific online algorithm to selectively send push notifications to volunteers in order to improve volunteer engagement. The algorithm is based on a recommender system and leverages historical data to control for unintended consequences. This work has been published at WWW 2021 [143].
    - We have deployed the recommender system in Chapter 5 with 412 Food Rescue and run a randomized controlled trial (RCT). The RCT showed that our algorithm improved both the hit rate and the claim rate of the food rescue operation. This is detailed in Chapter 6.
- Part III contains our work on using machine learning for environmental conservation. Here, we emphasize two common issues in applied learning and planning pipelines: label scarcity and label quality. We use techniques from active learning and noisy label learning

to address these challenges.

  - Chapter 7 details NewsPanda, a toolkit which automatically detects and analyzes online articles related to environmental conservation and infrastructure construction using a BERT-based model. For the identified articles, we perform further analysis, extracting keywords and finding potentially related sources. This work has been published at IAAI 2023 and won the IAAI deployed application award [76].

- In Part IV, going beyond the specific application domains in the previous parts, we distill research problem that addresses the common pain points we observed these AI4SG projects. In contrast to the one-shot learning and planning paradigm studied in previous chapters, here we propose and study iterative learning and planning.
  - In Chapter 8, we introduce bandit data-driven optimization as the first iterative prediction-prescription framework. We develop the PROOF algorithm and formally show that it achieves sublinear regret. We then apply it to the food rescue problem studied in Chapter 5. This work has been published at AAAI 2022 [141].

1.3 A Note on Real-World Impact

I believe AI4SG research that only stays on paper does not fulfill its purpose. This belief shaped my journey, affected my selection of research problems and research paradigm, and ultimately, manifested itself in the real-world impact that the work in this thesis has achieved.

The RCT in Chapter 6 was the first ever RCT on the food rescue platform. Thus, it involved a significant amount of preparation, both technical and non-technical. It is currently being rolled out to 16 different cities across North America. NewsPanda, as detailed in Chapter 7, has been a cross-team collaboration from the very beginning, as a project involving WWF offices in the UK, US, India, Nepal, Colombia, and Norway. It has been deployed since February 2022, and further improvement of NewsPanda has been a continuing effort. Even the work in Part I has resulted in a software publicly available online.

Part I

Learning and Planning for Cybersecurity

Chapter 2

Learning and Planning in the Feature Deception Problem

In many security domains, deception mitigates the defender's loss by misleading the attacker to make suboptimal decisions. In this chapter, we introduce the *feature deception problem (FDP)*, a domain-independent model to formally reason about deception. We present a learning and planning framework for finding the optimal deception strategy, taking into account the adversary's preferences which are initially unknown to the defender. We ground our presentation in the cybersecurity setting, but the model can be easily adapted to account for the learning and planning paradigm in other domains such as wildlife conservation, public safety, and so on.

2.1 Introduction

The world today poses more challenges to security than ever before. Consider the cyberspace or the financial world where a defender is protecting a collection of targets, e.g. servers or accounts. Despite the ever-improving security measures, malicious attackers work diligently and creatively to outstrip the defense [123]. Against an attacker with previously unseen exploits and abundant resources, the attempt to protect any target is almost surely a lost cause [69]. However, the defender could induce the attacker to attack a less harmful, or even fake, target. This can be seen as a case of deception.

Deception has been an important tactic in military operations for millenia [81]. More recently, it has been extensively studied in cybersecurity [67, 71]. At the start of an attack campaign, attackers typically perform reconnaissance to learn the configuration of the machines in the network using tools such as Nmap [97]. Security researchers have proposed many deceptive measures to manipulate a machine's response to these probes [9, 72], which could confound and mislead an attempt to attack. In addition, honey-X, such as honeypots, honey users, and honey files have been developed to attract the attackers to attack these fake targets [146]. For example, it is reported that country A once created encrypted but fake files with names of country B's military systems and marked them to be shared with country A's intelligence agency [108]. Using sensitive filenames as bait, country A successfully lured country B's hackers to these decoy targets.

Feature	Observed value	Actual value
Operating system	Windows 2016	RHEL 7
Service version	v1.2	v1.4
IP address	10.0.1.2	10.0.2.1
Open ports	22, 445	22, 1433
Round trip time for probes [136]	16 ms	84 ms

Table 2.1: Example features in cybersecurity

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

It is evident that the FDP model could be applied to many domains by appropriately defining the relevant set of features. To be concrete, we will ground our discussion in cybersecurity, where an attacker observes the features of each network node when attempting to fingerprint the machines (example features shown in the left column of Table 2.1) and then chooses a node to compromise. Attackers may have different preferences over feature value combinations when choosing targets to attack. If an intruder has an exploit for Windows machines, a Linux server might not be attractive. If the attacker is interested in exfiltration, he might choose a machine running database services. If the defender knows the attacker's preferences, she could strategically configure important machines appear undesirable or configure the honeypots to appear attractive to the attacker, by changing the observed value of the features, e.g. Table 2.1. However, to make an informed decision, she needs to first learn the attacker's preferences.

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

2.2 Related Work

Deception Deception has been studied in many domains, and of immediate relevance is its use in cybersecurity [131]. Studies have suggested that deceptively responding to an attacker's scanning and probing could be a useful defensive measure [9, 72]. Schlenker et al. [135] and Wang and Zeng [156] propose game-theoretic models where the defender manipulates the query response to a known attacker. Proposing a domain-independent model, we advance the state of the art by (1) providing a unified learning and planning framework with theoretical guarantee which can deal with unknown attackers, (2) extending the finite "type" space in both papers, where "type" is defined by the combination of feature values, to an infinite feature space that allows for both continuous and discrete features, and (3) incorporating a highly expressive bounded rationality model whereas both papers assume perfectly rational attackers.

For the more general case, Horak et al. [67] study a defender that engages an attacker in a sequential interaction. A complementary view where the attacker aims at deceiving the defender is provided in [50, 110]. Different from them, we assume no knowledge of the set of possible attacker types. In [50, 60, 110, 166] deception is defined as deceptively allocating defensive resources. We study feature deception where no effective tools can thwart an attack, which is arguably more realistic in high-stakes interactions. When such tools exist, feature deception is still valuable for strategic defense.

Learning in Stackelberg games Much work has been devoted to learning in Stackelberg games. Our work is most directly related to that of Haghtalab et al. [61]. They show that three defender strategies are sufficient to learn a SUQR-like adversary behavior model in Stackelberg security games. The only decision variable in their model, the coverage probability, may be viewed as a single feature in FDP. FDP allows for an arbitrary number of features, and this realistic extension makes their key technique inapplicable for analyzing the sample complexity. Our main learning result also removes the technical constraints on defender strategies present in their work. Sinha et al. [145] study learning adversary's preferences in a probably approximately correct (PAC) setting. However, their learning accuracy depends heavily on the quality of distribution from which they sample the defender's strategies. We provide a uniform guarantee in a distribution-free context. Other papers [20, 84, 99, 117] study the online learning setting with rational attackers. As pointed out in [61], considering the more realistic bounded rationality scenario allows us to make use of historical data and use our algorithm more easily in practice.

Planning with boundedly rational attackers Yang et al. [164] propose a MILP-based solution in security games. Our planning algorithm goes beyond the coverage probability and determines the configuration of multiple features, and adopt a more expressive behavior model. The subsequent papers that incorporate learning with such bounded rationality models do not provide any theoretical guarantee [45, 165]. A recent work develops a learning and planning pipeline in security games [118]. However, their algorithm requires the defender know a priori some parameters in the attacker's behavior model, and provides no global optimality guarantee.

2.3 The Feature Deception Problem

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

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

Table 2.1 shows an example in cybersecurity. In practice, there are many ways to implement deception. For example, a node running Windows (actual feature) manages to reply to reconnaissance queries in Linux style using tools like OSfuscate. Then the attacker might think the node is running Linux (observed feature). For IP deception, Jafarian et al. [70] and Chiang et al. [30] demonstrate methods to present to the attacker a different IP from the actual one. In addition, when we "fake open" a port with no real vulnerable service runs on it, an attack on the underlying service will fail. This could be done with command line tools or existing technologies like Honeyd [127].

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

Feasibility constraints For a feature k with actual value $\hat{x}_{ik}$, the defender can set its observed value $x_{ik} \in C(\hat{x}_{ik}) \subseteq [0, 1]$, where the feasible set $C(\hat{x}_{ik})$ is determined by the actual value. For continuous features, we assume $C(\hat{x}_{ik})$ takes the form $[\hat{x}_{ik} - \tau_{ik}, \hat{x}_{ik} + \tau_{ik}] \cap [0, 1]$ where $\tau_{ik} \in [0, 1]$. This captures the feasibility constraint in setting up the observed value of a feature based on its actual value. Take the round trip time (RTT) as an example. Shamsi et al. fingerprint the OS using RTT of the SYN-ACK packets [136]. Typical RTTs are in the order of few seconds (Fig. 4 [136]), while a typical TCP session is 3-5 minutes. Thus, perturbing RTT within a few seconds is reasonable, but greater perturbation is dubious.

For binary features, $C(\hat{x}_{ik}) \subseteq \{0, 1\}$. In addition to these feasibility constraints for individual features, we also allow for linear constraints over multiple features, which could encode natural constraints for categorical features with one-hot encoding, e.g. $\sum_{k \in M'} x_{ik} = 1$, with $M' \subseteq M$

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

being the subset of features that collectively represent one categorical feature. They may also encode the realistic considerations when setting up the observed features. For example, $x_{ik_1} + x_{ik_2} \leq 1$ could mean that a Linux machine ($x_{ik_1} = 1$) cannot possibly have ActiveX available ($x_{ik_2} = 1$).

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

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

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

$$f_w(x_i) = \exp\left(\sum_{k \in M} w_k x_{ik}\right). \tag{2.1}$$

We omit the subscript w when there is no confusion. This functional form is commonly used to approximate the agent's reward or utility function in inverse reinforcement learning and behavioral game theory, and has been empirically shown to capture many attacker preferences in cybersecurity [1]. For example, the tactics of advanced persistent threat group APT10 [128] are driven by: (1) final goal: they aim at exfiltrating data from workstation machines; (2) expertise: they employ exploits against Windows workstations; (3) services available: their exploits operate against file sharing and remote desktop services. Thus, APT10 prefer to attack machines with Windows OS running a file-sharing service on the default port. Each of these properties is a "feature" in FDP and a score function f in Eq (2.1) can assign a greater weight for each of these features. It can also capture more complex preferences by using hand-crafted features based on domain knowledge. For example, APT10 typically scan for NetBIOS services (i.e., ports 137 and 138), and Remote Desktop Protocol services (i.e., ports 445 and 3389) to identify systems that they might get onto [128]. Instead of treating the availability of ports as features, we may design a binary feature indicating whether each of the service is available (representing an "OR" relationship of the port availability features). We also show a more efficient way to approximately handle combinatorial preferences in Section 2.6.4. In addition, this score function also captures fully rational attackers in the limit.

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

2.4 Learning the Adversary's Preferences

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

In practice, there are two ways to carry out the learning stage. First, the defender can learn from historical data. Second, the defender can also actively collect data points while manipulating the observed features of the network. This is often done with honeynets [146], i.e. a network of honeypots.

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

To analyze the sample complexity of learning the adversary's preferences, we focus on the classical form score function f in Eq (2.1). We show that, in an FDP with m features, the defender can learn the attacker's behavior model correctly with high probability, using only m observed feature configurations and a polynomial number of samples. We view this condition as very mild, because even if the network admin's historical dataset does not meet the requirement, she could set up a honeynet to elicit attacks, where she can control the feature configurations of each target [146]. It is still not free for the defender to change configurations, but attacks on honeynet do not lead to actual loss since it runs in parallel with the production network.

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

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

Theorem 2.4.1. *Consider m observed feature configurations $x^1, x^2, \dots, x^m \in [0,1]^{mn}$. With $\Omega(\frac{\alpha^4 m^4}{\rho \epsilon^2} \log \frac{nm}{\delta})$ samples for each of the m feature configurations, with probability $1 - \delta$, we can learn a score function $\hat{f}(\cdot)$ with uniform multiplicative error ϵ of the true $f(\cdot)$, i.e., $\frac{1}{1+\epsilon} \leq \frac{f(x_i)}{\hat{f}(x_i)} \leq 1 + \epsilon, \forall x_i$.*

Proof. Let $\hat{D}^x(t) = \frac{\hat{f}(x_t)}{\sum_{i \in N} \hat{f}(x_i)}$. We leverage a known result from behavioral game theory [61]. It cannot be directly translated to sample complexity guarantee in FDP because of the correlation among feature configurations, but we use it to reason about attack probabilities in proving Theorem 2.4.1.

Lemma 2.4.2. *[61] Given observable features $x \in [0,1]^{mn}$, and $\Omega(\frac{1}{\rho\epsilon^2}\log\frac{n}{\delta})$ samples, we have $\frac{1}{1+\epsilon} \le \frac{\hat{D}^x(t)}{D^x(t)} \le 1+\epsilon$ with probability $1-\delta$, for all $t \in N$.*

Fix $\epsilon, \delta > 0$. From Eq. (2.1), for each x^i where $i = 1, 2, \ldots, m$, we have

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

Let

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

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

Let $\epsilon' = \frac{\epsilon}{4\alpha^2 m^2}$ and $\delta' = \frac{\delta}{m}$. Suppose we have $\Omega(\frac{1}{\rho\epsilon'^2}\log\frac{n}{\delta'})$ samples. From Lemma 2.4.2, for any node $r \in N$ and any feature configuration x^i where $i = 1, 2, \ldots, m$, $\frac{1}{1+\epsilon'} \le \frac{\hat{D}^{x^i}(r)}{D^{x^i}(r)} \le 1+\epsilon'$ with probability $1-\delta'$. The bound holds for all strategies simultaneously with probability at least $1 - m\delta' = 1-\delta$, using a union bound argument. In particular, for our chosen nodes s and t, we have

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

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

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

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

$$\begin{aligned} \frac{|(A^{st})^{-1}e| / |(A^{st})^{-1}b^{st}|}{|e|/|b^{st}|} &\le \max_{\tilde{e}, \tilde{b}^{st} \ne 0} \frac{|(A^{st})^{-1}\tilde{e}| / |(A^{st})^{-1}\tilde{b}^{st}|}{|\tilde{e}|/|\tilde{b}^{st}|} \\ &= \max_{\tilde{e} \ne 0} \frac{|(A^{st})^{-1}\tilde{e}|}{|\tilde{e}|} \max_{\tilde{b}^{st} \ne 0} \frac{|\tilde{b}^{st}|}{|(A^{st})^{-1}\tilde{b}^{st}|} = \max_{\tilde{e} \ne 0} \frac{|(A^{st})^{-1}\tilde{e}|}{|\tilde{e}|} \max_{y \ne 0} \frac{|A^{st}y|}{|y|} = ||(A^{st})^{-1}|| \cdot ||A^{st}|| \end{aligned}$$

This leads to

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

For any observable feature configuration x,

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

Therefore,

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

□

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

The α in Theorem 2.4.1 need not be large, especially if the defender can select the feature configurations to collect data and elicit preferences. Consider a sequence of m feature configurations $x^1, \ldots, x^m$, and focus on targets 1 and 2. For each x^j, let the features on target 1 be identical to target 2, except for the j-th feature, where $x^j_{1j} = 1$ and $x^j_{2j} = 0$. This leads to $A^{12} = I$, and $\alpha \leq 1$. This also shows that it is not hard to set up the configurations such that A^{st} is nonsingular.

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

Theorem 2.4.3. *In the setting of Theorem 2.4.1, if the attacker modifies a $\gamma \leq \frac{\epsilon\rho}{4\alpha m}$ fraction of the data points for each feature configuration, the function f can be learned within multiplicative error 3ϵ.*

Proof. Fix two nodes s, t. Recall that in Theorem 2.4.1, without data poisoning, we learned the weights w by solving the linear equations $A^{st}\tilde{w} = \tilde{b}^{st}$ based on the empirical distribution of attacks, where $\tilde{b}^{st} = (\ln \frac{\tilde{D}^{x^1}(s)}{\tilde{D}^{x^1}(t)}, \ldots, \ln \frac{\tilde{D}^{x^m}(s)}{\tilde{D}^{x^m}(t)})$. [2] Denote a parallel system of equations $A^{st}\hat{w} = \hat{b}^{st}$ which uses the poisoned data. We are interested in bounding $|\hat{w} - \tilde{w}| = |(A^{st})^{-1}(\hat{b}^{st} - \tilde{b}^{st})|$. Consider the k-th entry in the vector $\hat{b}^{st} - \tilde{b}^{st}$:

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

To simplify the notations, we denote $\tilde{D}^{x^k}(t) = \gamma^k_t$ and $\tilde{D}^{x^k}(s) = \gamma^k_s$, and without loss of generality, assume $\gamma^k_t \leq \gamma^k_s$. To find an upper bound of RHS of the above equation, we define function

[2] Refer to the proof of Theorem 2.4.1 for the notations used.

$g(\gamma_1, \gamma_2) = \frac{\gamma_t^k(\gamma_s^k+\gamma_1)}{\gamma_s^k(\gamma_t^k-\gamma_2)}$, and define function $h(\gamma_1, \gamma_2) = |\ln g(\gamma_1, \gamma_2)|$. The constraint that the attacker can only change γ fraction of the points translates into $|\gamma_1|, |\gamma_2|, |\gamma_1 - \gamma_2| \leq \gamma$. Since g is increasing in γ_1 and γ_2, g attains maximum at $(\gamma_1, \gamma_2) = (\gamma, \gamma)$ and minimum at $(\gamma_1, \gamma_2) = (-\gamma, -\gamma)$, which are the only two possible maxima of h. Observe that $g(\gamma, \gamma) \geq 1$ and $g(-\gamma, -\gamma) \leq 1$. It then suffices to compare $g(\gamma, \gamma)$ with $1/g(-\gamma, -\gamma)$:

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

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

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

Recall that

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

Thus, we get

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

Note that by Lemma 2.4.2, we have $\gamma_t^k \geq \frac{\rho}{1+\epsilon'} \geq \frac{\rho}{2}$. Since we assumed that $\gamma \leq \frac{\epsilon\rho}{4\alpha m} \leq \frac{\epsilon\rho}{4}$, we know that $\gamma \leq \gamma_t/2$. Thus, we get

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

From here, using the triangle inequality, we have

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

Thus, in the end, we get

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

□

For a general score function f_w, gradient-based optimizer, e.g. RMSProp [63] can be applied to learn w through maximum-likelihood estimation.

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

$$L_{w'}^j(N^j, x^j, y^j) = \log(f_{w'}(x_{y^j}^j)) - \log(\sum\nolimits_{i \in N^j} f_{w'}(x_i^j))$$

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

2.5 Computing the Optimal Feature Configuration

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

$$\min_x \quad \frac{\sum_{i\in N} f(x_i)u_i}{\sum_{i\in N} f(x_i)} \tag{2.2}$$

$$s.t. \quad \sum_{i\in N}\sum_{k\in M} \eta_{ik}|x_{ik} - \hat{x}_{ik}| \leq B \tag{2.3}$$

$$\text{Categorical feature constraints} \tag{2.4}$$

$$x_{ik} \in C(\hat{x}_{ik}) \qquad \forall i \in N, k \in M \tag{2.5}$$

This MP is typically non-convex and very difficult to solve. We show that the decision version of FDP is NP-complete. Hence, finding the optimal feature configuration is NP-hard. In fact, this holds even when there is only a single binary feature and the score function f takes the form in Eq. (2.1).

Theorem 2.5.1. *FDP is NP-complete.*

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

We construct an instance of FDP. Let the set of targets be $N = \{1, \dots, n+1\}$, and let there be a single binary feature, i.e. $M = \{1\}$ and $x_{i1} \in \{0,1\}$ for each $i \in N$. Since there is only one feature, we abuse the notation by using $x_i = x_{i1}$. Suppose each target's actual value of the feature is $\hat{x}_i = 0$. Consider a score function f with $f(0) = 1$ and $f(1) = 2$. For each $i \in N$, let $u_i = (1 - v_i)/\delta$ if $i \neq n+1$, and $u_{n+1} = (1 + V + \sum_{i=1}^n v_i)/\delta$. Choose a large enough $\delta \geq 1$ so that $u_{n+1} \leq 1$. For each $i \in N$, let $\eta_i = \omega_i$ if $i \neq n+1$, and $\eta_{n+1} = 0$. Finally, let the budget $B = \Omega$.

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

Despite the negative results for the general case, we design an approximation algorithm for the classical score function in Eq. (2.1) based on mixed integer linear programming (MILP) enhanced with binary search. As shown in Sec 2.6, it can solve medium sized problems (up to 200 targets) efficiently. Given $f(x_i) = \exp(\sum_{k\in M} w_k x_{ik})$, scaling the score by a factor of e^{-W} does not affect the attack probability, where $W = |w|$ is the L^1 norm of $w = (w_1, \dots, w_m)$. Thus, we treat the score function as $f(x_i) = \exp(\sum_{k\in M} w_k x_{ik} - W)$.

With slight abuse of notation, we denote the score of target i as f_i. Let $z_i = \sum_{k\in M} w_k x_{ik} - W \in [-2W, 0]$. We divide the interval $[-2W, 0]$ into $2W/\epsilon$ subintervals, each of length ϵ. On interval $[-l\epsilon, -(l-1)\epsilon]$ with $l = 0, 1, \dots, 2W/\epsilon$, we approximate the function e^{z_i} with the line segment of slope γ_l connecting the points $(-l\epsilon, e^{-l\epsilon})$ and $(-(l-1)\epsilon, e^{-(l-1)\epsilon})$. We use this method to approximate f_i in the following mathematical program $\mathcal{MP}1$. We represent $z_i = -\sum_l z_{il}$, where each variable z_{il} indicates the quantity z_i takes up on the interval $[-l\epsilon, -(l-1)\epsilon]$. The constraints in Eq. (2.9)-(2.10) ensure that $z_{i(l+1)} > 0$ only if $z_{il} = \epsilon$. While $\mathcal{MP}1$ is not technically a MILP, we can linearize the objective and the constraint involving absolute value following a standard procedure [147]. The full MILP formulation can be found in Appendix A.1.

$$
\begin{aligned}
(\mathcal{MP}1) \quad & \min_{f,z,x,y} \quad \frac{\sum_i f_i u_i}{\sum_i f_i} && (2.6)\\
s.t. \quad & f_i = e^{-2W} + \sum_l \gamma_l(\epsilon - z_{il}), \quad \forall i \in N && (2.7)\\
& \sum_{k \in M} w_k x_{ik} - W = -\sum_l z_{il}, \quad \forall i \in N && (2.8)\\
& \epsilon y_{il} \le z_{il}, z_{i(l+1)} \le \epsilon y_{il}, \quad \forall l, \forall i \in N && (2.9)\\
& z_{il} \in [0, \epsilon], y_{il} \in \{0, 1\}, \quad \forall l, \forall i \in N && (2.10)\\
& \text{Constraints (2.3)-(2.5)}
\end{aligned}
$$

We can now establish the following bound.

Theorem 2.5.2. *Given $\epsilon < 1$, the MILP is a $2\epsilon^2$-approximation to the original problem.*

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

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

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

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

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

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

Therefore, we obtain

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

□

Algorithm 1: Milp-bs

```
1 Initialize $L = -1, U = 1, \delta = 0, \epsilon_{bs}$
2 while $U - L > \epsilon_{bs}$ do
3     Solve the MILP $\mathcal{MP}1$ with objective in Eq. (2.11).
4     if objective value $< 0$ then
5         Let $U = \delta$
6     else
7         Let $L = \delta$
8 return $U$, the MILP solution when $U$ was last updated.
```

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

$$\min_{f,z,x,y} \quad \sum_i f_i u_i - \delta \sum_i f_i. \tag{2.11}$$

At each iteration, if the objective value of Eq. (2.11) is negative, we update the binary search upper bound, and update the lower bound if positive. We proceed to the next iteration until the gap between the bounds is smaller than tolerance ϵ_{bs} and then we output the solution x^{bs} when the upper bound was last updated. The complete procedure is given as Alg. 1. Since Eq. (2.11) is linear itself, we no longer need to perform linearization on it to obtain a MILP. This leads to significant speedup as we show later. We also preserve the approximation bound using triangle inequalities.

Theorem 2.5.3. *Given $\epsilon < 1$ and tolerance ϵ_{bs}, binary search gives a $(2\epsilon^2 + \epsilon_{bs})$-approximation.*

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

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

□

Now, we connect the learning and planning results together. Suppose we learned an approximate score function $\hat{f}$ (Theorem 2.4.1), and we find an approximately optimal feature configuration (Theorem 2.5.2) assuming $\hat{f}$. The following result shows that we can still guarantee end-to-end approximate optimality.

Theorem 2.5.4. *Suppose for some $\epsilon \le 1/4$, $\frac{1}{1+\epsilon} < \frac{\hat{f}(x_i)}{f(x_i)} < 1+\epsilon$ for all x_i. Then, $|U(x,\hat{f}) - U(x,f)| \le 4\epsilon$ for all x. Let $x^* = \arg\min_x U(x,f)$ and x'' be such that $U(x'',\hat{f}) \le \min_x U(x,\hat{f}) + \eta$, then $U(x'',f) - U(x^*,f) \le 8\epsilon + \eta$.*

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

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

we get

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

That is, $|\sum_k(\hat{w}_k - w_k)x_{ik}| < \epsilon$. The proof of Theorem 3.7 in [61] now follows to prove the first part of Theorem 2.5.4 if we redefine their $u_i(p_i)$ as $\sum_{k\in M} w_k x_{ik}$ and $\hat{u}_i(p_i)$ as $\sum_{k\in M}\hat{w}_k x_{ik}$. For completeness, we adapt their proof below using our notations.

As defined in Section 2.4, $D^x(t) = \frac{f(x_t)}{\sum_i f(x_i)}$ and $\hat{D}^x(t) = \frac{\hat{f}(x_t)}{\sum_i \hat{f}(x_i)}$. We have

$$\begin{aligned}\left|\ln\frac{\hat{D}^x(t)}{D^x(t)}\right| &= \left|\left(\sum_k(\hat{w}_k - w_k)x_{tk}\right) - \ln\frac{\sum_i \exp\{\sum_k \hat{w}_k x_{ik}\}}{\sum_i \exp\{\sum_k w_k x_{ik}\}}\right| \\ &\le \left|\sum_k(\hat{w}_k - w_k)x_{tk}\right| + \left|\ln\frac{\sum_i \exp\{\sum_k w_k x_{ik}\}\exp\{\sum_k(\hat{w}_k - w_k)x_{ik}\}}{\sum_i \exp\{\sum_k w_k x_{ik}\}}\right| \\ &< \epsilon + \max_i \left|\ln\exp\{\sum_k(\hat{w}_k - w_k)x_{ik}\}\right| < 2\epsilon\end{aligned}$$

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

$$\begin{aligned}|U(x,\hat{f}) - U(x,f)| &= \left|\sum_{i\in N}(\hat{D}^x(i) - D^x(i))u_i\right| \le \sum_{i\in N}\left|\hat{D}^x(i) - D^x(i)\right||u_i| \\ &= \sum_{i\in N}\left|\frac{\hat{D}^x(i)}{D^x(i)} - 1\right||u_i|\,D^x(i) \le 4\epsilon\sum_{i\in N}|u_i|\,D^x(i) \le 4\epsilon\max_{i\in N}|u_i| \le 4\epsilon\end{aligned}$$

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

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

$\square$

In addition, we propose two exact algorithms for special cases of FDP, which can be found in Appendix A.2. When the deception cost is associated with discrete features only, we provide an exact MILP formulation. When there is no budget and feasibility constraints, we can find the optimal defender strategy in $O(n \log n + m)$ time using a greedy algorithm. Inspired by this greedy algorithm, we introduce a greedy heuristic for the general case. Greedy (Alg.2 in Appendix A.1) finds the feature vectors that maximize and minimize the score, respectively, using gradient descent-based algorithm. It then greedily applies these features to targets of extreme losses. We show its performance in the following section as well.

2.6 Experiments

We present the experimental results for our learning and planning algorithms separately, and then combine them to demonstrate the effectiveness of our learning and planning framework. All experiments are carried out on a 3.8GHz Intel Core i5 CPU with 32GB RAM. We use Ipopt as our non-convex solver and CPLEX 12.8 as the MILP solver. All results are averaged over 20 instances; error bars represent standard deviations. Details about hyper-parameters can be found in Appendix A.4.

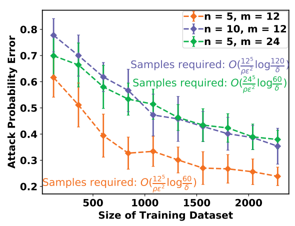
2.6.1 Learning

Classical score function First, we assume the adversary uses the classical score function in Eq (2.1). The defender learns this score function using the closed-form estimation (CF) in Theorem 2.4.1. We study how the learning accuracy changes with the size of training sample d. We sample the parameters of the true score function f uniformly at random from $[-0.5, 0.5]$. We then generate m feature configurations uniformly at random. For each of them, we sample the attacked target d/m times according to f, obtaining a training set of d samples. We generate a test set $\tilde{D}$ of 5×10^5 configurations sampled uniformly at random. We measure the learning error as the mean total variation distance between the attack distribution from the learned $\hat{f}$ and that of the true model f:

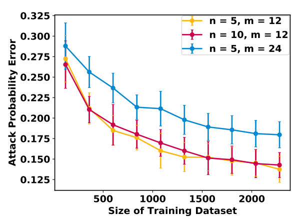
$$\frac{1}{|\tilde{D}|}\sum_{j=1}^{|\tilde{D}|} d_{TV}\left(\left(\frac{f(x_i^j)}{\sum_{t\in N} f(x_t^j)}\right)_{i\in N}, \left(\frac{\hat{f}(x_i^j)}{\sum_{t\in N} \hat{f}(x_t^j)}\right)_{i\in N}\right).$$

Figure 2.1a shows that the learning error decreases as we increase the number of samples. Theorem 2.4.1 provides a sample complexity bound, which we annotate in Figure 2.1a as well. The experiment shows that we need much fewer samples to learn a relatively good score function, and smaller games exhibit smaller learning error.

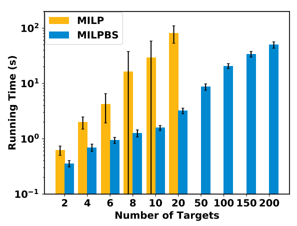
3-layer NN represented (NN-3) score function We assume the adversary uses a 3-layer neural network score function, whose details are in Appendix A.4. We use the gradient descent-based (GD) learning algorithm RMSProp as described in Section 2.4, with learning rate 0.1. For each sample size d, we generate d feature configurations and sample an attacked target for each of them in the training set. Fig. 2.1b shows GD can minimize the learning error to below



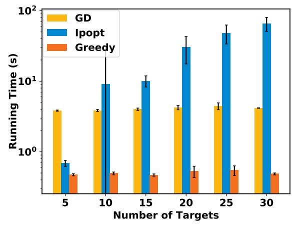
(a) Learning classical score function in Eq. (2.1)



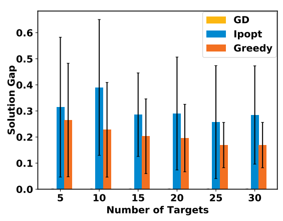
(b) Learning NN-3 score function



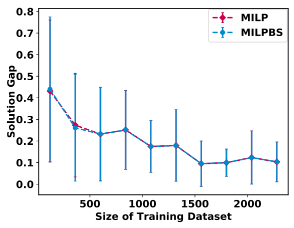
(c) Planning with classical score function in Eq. (2.1), $m = 12$



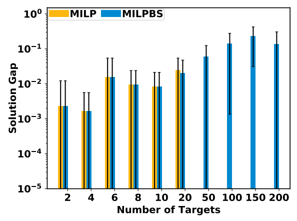
(d) Planning with NN-3 score function, $m = 12$



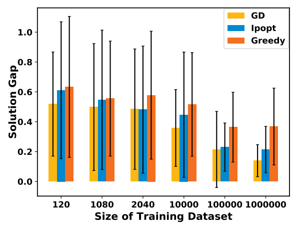
(e) Planning with NN-3 score function, $m = 12$



(f) Learning + planning, classical score function, $n = 5$, $m = 12$



(g) Learning + Planning, classical score function, $m = 12$



(h) Learning + Planning, NN-3 score function, $n = 5$, $m = 12$

Figure 2.1: Experimental results

0.15. Note that the training data are different in Fig. 2.1a and 2.1b, thus the two figures are not directly comparable.

We also measured $|\hat{\theta} - \theta|$, the L_1 error in the score function parameter θ, which directly relates to the sample complexity bound in Theorem 2.4.1. We include the results in Appendix A.3.

2.6.2 Planning

We test our algorithms on finding the optimal feature configuration against a known attacker model. The FDP parameter distributions are included in Appendix A.4.

Classical score function Fig. 2.1c shows that the binary search version of the MILP based on $\mathcal{MP}1$ (MILPBS) runs faster than that without binary search on most instances. MILPBS scales up to problems with 200 targets, which is already at the scale of many real-world problems. MILP does not scale beyond problems with 20 targets. In Appendix A.3, we show that MILPBS also scales better in terms of the number of features. We set the MILP's error bound at 0.005 and $\epsilon_{\text{bs}} = 1e - 4$; the difference in the two algorithms' results is negligible.

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

Fig. 2.1d and 2.1e show the running time and solution gap fixing $m = 12$. The running time of GD and Greedy does not change much across different problem sizes, yet Ipopt runs slower than the former two on most problem instances. GD also has smaller solution gap than Ipopt and Greedy. In Appendix A.3 we show the number of features affect these metrics in a similar way.

2.6.3 Combining Learning and Planning

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

Classical score function We test learning algorithm CF and planning algorithms P $\in$ {MILP, MILPBS}. Fig. 2.1f shows how the solution gap changes with the size of the training dataset. With $n \leq 20$ targets, all algorithms yield solution gaps below 0.1 (Fig. 2.1g). The reader might note the overlapping error bars, which are expected since MILP and MILPBS should not differ much in solution quality. Indeed, the difference is negligible as the smallest p-value of the 6 paired t-tests (fixing the number of targets for which they are tested) is 0.16.

NN-3 score function We test learning algorithm GD and planning algorithms P $\in$ {GD, Ipopt, Greedy}. Fig. 2.1h shows how the solution gap changes with the size of training dataset d. Paired t-tests suggest that GD has significantly smaller solution gap than Greedy ($p < 0.03$) at each size of training dataset except 1080. Ipopt also has significantly smaller solution gap than Greedy ($p < 0.01$) when on large datasets with $d \geq 10^5$ samples. On the largest dataset $d = 10^6$, GD also performs significantly better than Ipopt ($p = 0.04$).

Compared to the case with classical score functions, more data are required here to achieve a small solution gap. Since learning error is small for both cases (Fig. 2.1a,2.1b), this suggests planning is more sensitive to NN-3 score functions than classical score functions.

2.6.4 Case Study: Credit Bureau Network

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

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

As shown in Table 2.2, we consider a network of 10 nodes (i.e. targets) with 6 binary features: operating system (Windows/Linux) and the availability of SMTP, NetBIOS, HTTP, SQL, and

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

Table 2.2: Feature configuration of a typical credit bureau computer network.

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

Table 2.3: Learning + planning results for 2 types of attackers.

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

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

First, we consider an APT-like attacker, who wants to exfiltrate data by exploiting the SMTP service. They have expertise in Linux systems and want to maintain a high degree of stealth. Thus, their decision rule is based on the three requirement features: Linux, SMTP, and SQL. Without deception, the attacker would randomize attack over nodes 5-9, because these nodes satisfy 3 requirements and other nodes satisfy at most 2. As shown in Table 2.3, the optimal solution for the learning and planning problem leads to an expected defender's loss of 0.325, which is a 42% decrease from the loss with no deception. With limited budget, the defender makes the least harmful target, node 1, very attractive and the most harmful targets, nodes 8 and 9, less attractive.

We also consider a botnet attacker, who wants to create a bot by exploiting the NetBIOS service. They have expertise in Windows and want to maintain a moderate degree of stealth. Thus, their decision rule is based on two requirement features: Windows and NetBIOS. The results in Table 2.3 shows that the defender should set the NetBIOS observed value to be off for nodes 3 and 4, attracting the attacker to the least harmful nodes. This decreases the defender's expected loss by 50% compared to not using deception.

2.7 Discussion

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

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

Third, typically, actual features on functional targets are environmental parameters beyond the defender's control, or at least have high cost of manipulation. Altering them and defender's losses u_i does not align conceptually with deception. Thus, we treat them as fixed. For a target with no fixed actual values, e.g., a honeypot, the defender's cost is just the cost of configuring the feature, e.g., installing Windows. For consistency, we can set $\hat{x}_{ik}$ as the feature value with the lowest configuration cost, and η_{ik} is the additional cost for a different feature value.

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

Fifth, when faced with a group of attackers, in FDP we learn an average behavioral model of the population. To handle multiple attacker types, one could refer to the literature on Bayesian Stackelberg games [115].

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

Ethical implications The intended use case of feature deception problem is in defending the network of some entity against cyber attackers. In such scenario, if the network node is used by some individual, as opposed to organizational use, then feature deception might put certain individuals under bigger threats of cyber attack than others. Our model allows for technical way to address this by modifying the node's loss u_i. In practice, a more holistic approach is needed to protect the security of all users on the network.

Chapter 3

Draining the Water Hole: Mitigating Social Engineering Attacks with CyberTWEAK

Whereas the previous chapter contains a relatively general-purpose framework to reason about strategic deception, in this chapter, we dive into a particular type of cyber attacks, i.e. the watering hole attacks, which is a major threat to businesses and individuals alike. We present the Social Engineering Deception (SED) game model and the CyberTWEAK algorithm. The algorithm alters the environment information in web traffic to deceive the attackers. Based on our algorithms, we built a browser extension which is publicly available online.

3.1 Introduction

Social engineering attacks are a scourge for the well-being of today's online user and the current threat landscape only continues to become more dangerous [102]. Social engineering attacks manipulate people to give up confidential information through the use of phishing campaigns, spear phishing whaling or watering hole attacks. For example, in watering hole attacks, the attacker compromises a legitimate website and redirects visitors to a malicious domain where the attacker can intrude the user's network. The number of social engineering attacks is growing at a catastrophic rate. In a recent survey, 60% organizations were or may have been victim of at least one attack [5]. Such cybercrime poses an enormous threat to the security at all levels – national, business, and individual.

To mitigate these attacks, organizations take countermeasures from employee awareness training to technology-based defenses. Unfortunately, existing defenses are inadequate. Watering hole attackers typically use zero-day exploits, rendering patching and updating almost useless [148]. Sand-boxing potential attacks by VM requires high-end hardware, which hinders its wide adoption [47]. White/blacklisting websites is of limited use, since the adversary is strategically infecting trustworthy websites.

We propose a game-theoretic deception framework to mitigate social engineering attacks, and, in particular, the watering hole attacks. Deception is to delay and misdirect an adversary by incorporating ambiguity. Watering hole attackers rely on the identification of a visitor's system environment to deliver the correct malware to compromise a victim. Towards this end, the defender can manipulate the identifying information in the network packets, such as the user-agent string, IP address, and time-to-live. Consequently, the attacker might receive false or confusing information about the environment and send incompatible exploits. Thus, deceptively manipulating employees' network packets provides a promising countermeasure to social engineering attacks.

Our Contributions We provide the first game-theoretic framework for autonomous countermeasures to social engineering attacks. We propose the Social Engineering Deception (SED) game, in which an organization (defender) strategically alters its network packets. The attacker selects websites to compromise, and captures the organization's traffic to launch an attack. We model it as a zero-sum game and consider the minimax strategy for the defender.

Second, we analyze the structure and properties of the SED game, based on which we identify real-world scenarios where the optimal protection policy can be found efficiently.

Third, we propose the CyberTWEAK (Thwart WatEring hole AttacK) algorithm to solve the SED game. CyberTWEAK exploits theoretical properties of SED, linear program relaxation of the attacker's best response problem, and the column generation method, and is enhanced with dominated website elimination. We show that our algorithm can handle corporate-scale instances involving over 10^5 websites.

Finally, we have developed a browser extension based on our algorithm. The software is now publicly available on the Chrome Web Store.[1] The extension is able to manipulate the user-agent string in the network packets. We take additional steps to improve the its usability and explain the output of CyberTWEAK intuitively. We believe it will be vital to the continued development of social engineering defenses.

3.2 Related Work

Deception is one of the most effective ways to thwart cyberattacks. Recent papers have considered deception techniques for protecting an enterprise network from an attack by sending altered system environment information in response to scans performed during the reconnaissance phase of an attack [9, 72]. There is a rising interest in building game-theoretic models for deception [135], in particular in the use of honeypots [42, 121] in the enterprise network.

However, there is a fundamental difference between enterprise network defense and social engineering defense. In the former, an adversary targets an organization by compromising computers in the network while in watering hole attacks the attacker targets the user and compromises external websites. A website in SED cannot be properly modeled as a honeypot target, because the defender has no control over it. Neither can the user, because the attack depends on an external task – compromising a website. Instead of actively querying the network, watering hole attackers passively monitor the users' traffic. This necessitates the continuous action space for the attacker in SED, which is also different from most previous works on enterprise network defense.

[1]`http://bit.ly/CyberTWEAK`

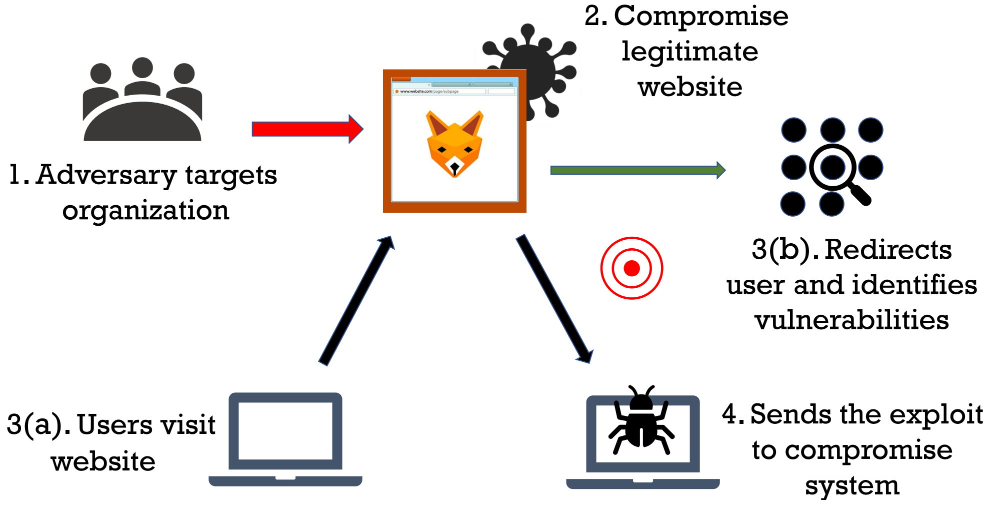

2. Compromise legitimate website
1. Adversary targets organization
3(b). Redirects user and identifies vulnerabilities
3(a). Users visit website
4. Sends the exploit to compromise system

Figure 3.1: Anatomy of a watering hole attack. We introduce uncertainty in step 3 of the attack, so that the attacker gets false information about the user and sends incompatible exploits.

Laszka et al. [80] study spear phishing, another form of social engineering attacks. The nature of watering hole attacks leads to additional complications. For example, watering hole attackers need to compromise a website and then scan the traffic. Thus, in SED the attacker has two layers of decision making: one continuous and one discrete. This leads to a different problem formulation and solution techniques than those in spear phishing.

3.3 Watering Hole Attacks

Watering hole attacks are a prominent type of social engineering used by sophisticated attackers. Before we describe our modeling decisions, it is useful to highlight the primary steps in executing a watering hole attack, as illustrated in Fig. 3.1. In step 1, the attacker identifies a target organization. They use surveys and external information like specialized technical sites to understand the browsing habits of its employees. This allows the adversary to determine the most lucrative websites to compromise for maximum exposure to employees from the targeted organization. In step 2, the adversary compromises a set of legitimate websites. Not only do these websites need to be lucrative, but the attacker also has to be strategic in this choice. For example, compromising Google.com is nearly impossible while the Polish Financial Authority, victim of the 2017 Ratankba malware attacks [150], cannot invest the same security resources. Indeed, in previous attacks the attacker was not observed to compromise all websites [114]. In step 3, employees visit the compromised website and are redirected to a malicious website which scans their system environment and the present vulnerabilities. To gather this information, attackers use techniques such as analyzing the user-agent string, operating system fingerprinting, etc. In Step 4, the attacker delivers an exploit for an identified vulnerability. After these steps, the attacker can navigate the target network and access the sensitive information.

Our algorithm and browser extension introduce uncertainty in step 3 of a watering hole attack. Identifying the vulnerabilities in a visitor relies on the information gathered from reconnaissance. The extension modifies the network packets so that the attacker gets false information about the visitor. Deception is not free, though. Altering the network packet can degrade the webpage rendered, e.g., displaying for Android on a Windows desktop. Thus, the defender needs to carefully trade off security and the quality of service.

In reality, sophisticated attackers typically do not send all exploits without tailoring to the packet information, as defense would become easier after seeing more such unknown exploits. Also, sending all exploits would be flagged as suspicious and get blocked. The attacker would need to get a new zero-day – a costly proposition. Thus, the attacker prefers scanning the system environment of the incoming traffic.

3.4 Social Engineering Deception Game

We model the strategic interaction between the organization (defender) and an adversary as a two-player zero-sum game, where the defender chooses an alteration policy and the adversary chooses which websites to compromise and decides the effort spent on scanning traffic. In everyday activities employees of a target organization O visit a set of websites W which includes legitimate sites and potential watering holes set up by an adversary. Let t_w^{all} denote the total amount of traffic to $w \in W$ from all visitors and t_w the total traffic to w from O. The defender's alteration policy is represented by $x \in [0,1]^{|W|}$ where x_w is the proportion of O's traffic to website $w \in W$ for which the network packet will be altered. We assume a drive-by download attack will be unsuccessful if, and only if an employee's packet is altered. However, it is easy to account for different levels of adversary and defender sophistication by adding an additional factor in Eq. (3.1) below. We consider a cost c_w to alter a single unit of traffic to w. The defender is limited to a budget B_d on the allowable cost.

The adversary first chooses which websites to compromise, represented by a binary vector $y \in \{0,1\}^{|W|}$. If $y_w = 1$, i.e., they turn website w into a watering hole, they must pay a cost π_w. The attacker has a budget B_a for compromising websites (w.l.o.g. we assume $\pi_w \leq B_a \; \forall w \in W$). The adversary then decides the scanning effort for each compromised website which can enable them to send exploits tailored to the packet information. We use e_w to denote how much traffic the attacker decides to scan per week for w, and refer to e as the *effort vector*. The discreet attacker has a budget B_e for scanning the incoming traffic. In the special case where the scanning effort is negligible ($B_e = \infty$), all our complexity and algorithmic results to be introduced still hold.

We consider an attacker who aims to maximize the expected amount of unaltered flow from target organization O that is scanned by them, as each unit of scanned unaltered flow can lead to a potential success in the social engineering attack, i.e., compromise an employee and discover critical information about O. We model it as a zero-sum game, and therefore the defender's goal is to minimize this amount.

Social engineering is a complex domain which we cannot fully model. However, we build our model and assumptions so that we can formally reason about deception, and even when our assumptions are not met, our work provides a sensible solution. For example, cyber attackers may have tools to circumvent existing deception techniques. Nonetheless, our solution increases the attacker's uncertainty about the environment as they cannot easily obtain or trust the information in the network packets. In Appendix B.4, we provide a detailed discussion of the generality and limitations of our work.

3.5 Computing Optimal Defender Strategy

In this section, we present complexity analysis and algorithms for finding the optimal defender strategy x^* in this game, which is essentially the minimax strategy, i.e., a strategy that minmizes the attacker's maximum possible expected amount of scanned unaltered flow. x^* should be the solution of the following bi-level optimization problem $\mathcal{P}_1$.

$$\begin{align}
\mathcal{P}_1 \ : \ \min_x \max_{y,e} \quad & \sum\nolimits_{w\in W} \kappa_w(1-x_w)e_w \tag{3.1}\\
\text{s.t.} \quad & \sum\nolimits_{w\in W} e_w \le B_e \tag{3.2}\\
& \sum\nolimits_{w\in W} \pi_w y_w \le B_a \tag{3.3}\\
& e_w \le t_w^{all} \cdot y_w, \forall w \in W \tag{3.4}\\
& y_w \in \{0,1\}, \forall w \in W \tag{3.5}\\
& e_w \in [0,\infty), \forall w \in W \tag{3.6}\\
& \sum\nolimits_{w\in W} c_w t_w x_w \le B_d \tag{3.7}\\
& x_w \in [0,1], \forall w \in W \tag{3.8}
\end{align}$$

In objective function 3.1, $\kappa_w = t_w/t_w^{all}$. Since $t_w(1-x_w)$ is the total amount of unaltered flow from the defender organization O and e_w/t_w^{all} is the percentage of incoming traffic that will be scanned, $\kappa_w(1-x_w)e_w$ is the total scanned unaltered traffic to w. Constraint 3.2-3.3 describes the budget constraint for the attacker, and Constraint 3.4 requires that the attacker can only scan traffic for the compromised websites. Constraint 3.7 is the budget constraint for the defender.

Unfortunately, solving $\mathcal{P}_1$ is challenging. It cannot be solved using any of the existing solvers directly due to the bi-level optimization structure, the mix of real-valued and binary variables and the bilinear terms in the objective function ($x_w e_w$). In fact, even the adversary's best response problem $\mathcal{P}_2(x)$, represented as a mixed integer linear program (MILP) below, is NP-hard as stated in Thm 3.5.1.

$$\begin{align}
\mathcal{P}_2(x) \ : \quad \max_{y,e} \quad & \sum\nolimits_{w\in W} \kappa_w(1-x_w)e_w \tag{3.9}\\
\text{s.t.} \quad & \text{Constraints } (3.2) \sim (3.6) \tag{3.10}
\end{align}$$

Theorem 3.5.1. *Finding adversary's best response is NP-hard.*

Proof. We reduce from the knapsack problem. In the knapsack problem, we have a set W of items each with a weight ω_w and value p_w $\forall w \in N$, and aim to pick items of maximum possible value subject to a capacity B. We now create an instance of the SED problem. Create a website for each item $w \in W$ with organization traffic and total traffic $t_w = t_w^{all} = p_w$ and attack cost ω_w. Assume that $x = \mathbf{0}^T$. Next, set $B_a = B$ and $B_e = \infty$. Notice that the objective function becomes

$\sum_{w\in W} e_w$ where $\sum_{w\in W} e_w \le \infty$ and $e_w \le p_w y_w$. Hence, $e_w = p_w$ whenever $y_w = 1$. Then, the adversary's best response problem is given by:

$$\max_{y} \quad \sum_{w\in W} p_w y_w \tag{3.11}$$

$$\text{s.t.} \quad \sum_{w\in W} \omega_w y_w \le B \tag{3.12}$$

$$y_w \in \{0,1\} \qquad \forall w \in W \tag{3.13}$$

This is exactly the knapsack problem described above. □

Therefore, we exploit the structure and properties of SED and $\mathcal{P}_1$ and design several novel algorithms to solve it. We first identify two tractable special classes of SED games which can be solved in polynomial time and discuss their real world implications. Then we present CyberTWEAK, our algorithm for general SED games.

3.5.1 Tractable Classes

The first tractable class is identified based on the key observation stated in Thm 3.5.2: the optimal solutions of SED games exhibit a greedy allocation of the attacker's effort budget. That is, for at most one website w will the attacker spend scanning effort neither zero nor t_w^{all}.

Theorem 3.5.2. *Let (x^*, y^*, e^*) be an optimal solution to $\mathcal{P}_1$, $W_F = \{w : e_w^* = t_w^{all}\}$, $W_Z = \{w : e_w^* = 0\}$, $W_B = \{w : e_w^* \in (0, t_w^{all})\}$. There is an optimal solution with $|W_B| \le 1$.*

Proof. For each $w \in W$, let $k_w = t_w(1 - x_w^*)/t_w^{all}$. Suppose there exist some $w_1, w_2 \in W_B$, and w.l.o.g assume $k_{w_1} \ge k_{w_2}$. Let $\Delta e = \min\{e_{w_2}^*, t_{w_1}^{all} - e_{w_1}^*\}$. Consider the solution $(x^*, y^*, \hat{e})$ where $\hat{e}_{w_1} = e_{w_1}^* + \Delta e$, $\hat{e}_{w_2} = e_{w_2}^* - \Delta e$, and $\hat{e}_w = e_w^*$ for all other websites $w \in W$. This is a feasible solution, and the objective increases by $(k_{w_1} - k_{w_2})\Delta e \ge 0$ compared to (x^*, y^*, e^*). Furthermore, at least one of w_1 and w_2 is removed from W_B. We can apply this argument repeatedly until $|W_B| \le 1$. □

As a result, if the attacker's scanning budget is so limited that he cannot even scan through the traffic of any website, he will use all the scanning effort on one website in the optimal solution. Thus, the optimal defender strategy can be found by enumerating the websites.

Corollary 3.5.3. *(Small Effort Budget) If $0 < B_e \le t_w^{all}, \forall w$, the optimal solution can be found in polynomial time.*

Proof. Since $B_e \le t_w^{all}$ $\forall w \in W$, we know $|W_F| \le 1$ for any feasible solution. If $|W_F| = 1$, then we have $|W_Z| = n - 1$ and $|W_B| = 0$. If $|W_F| = 0$, by Theorem 3.5.2, we have $|W_B| = 1$ and $|W_Z| = n - 1$. In either case, there is only website w^* such that $e_{w^*} > 0$. It follows that $w^* \in \arg\max_{w\in W} \frac{t_w(1-x_w)B_e}{t_w^{all}}$ given a defender strategy x. The optimal defender strategy can be found by solving the following LP.

$$\min_{x,v} \quad v \tag{3.14}$$

$$\text{s.t.} \quad v \geq \frac{t_w(1-x_w)B_e}{t_w^{all}} \qquad \forall w \in W \tag{3.15}$$

$$\sum_{w\in W} c_w t_w x_w \leq B_d \tag{3.16}$$

$$x_w \in [0,1] \qquad \forall w \in W \tag{3.17}$$

□

The second tractable class roots in the fact that if the scanning effort is negligible (or equivalently, $B_e = \infty$) the attacker only needs to reason about which websites to compromise. Further, if the attacker has a systematic way of compromising a website which makes the cost π_w uniform across websites, then the attacker only needs to greedily choose the websites with the highest unaltered incoming traffic and the defender can greedily alter traffic in the top websites. We provide details about these algorithms in the appendix.

Theorem 3.5.4. *(Uniform Cost + Unlimited Effort) If $\pi_w = 1, \forall w \in W$ and $B_e = \infty$, the defender's optimal strategy can be found in polynomial time.*

Proof. Under these assumptions, the problem $\mathcal{P}_1$ becomes

$$\min_{x} \max_{y,e} \quad \sum_{w\in W} t_w(1-x_w)y_w \tag{3.18}$$

$$\text{s.t.} \quad \sum_{w\in W} y_w \leq B_a \tag{3.19}$$

$$\sum_{w\in W} c_w t_w x_w \leq B_d \tag{3.20}$$

$$x_w \in [0,1], y_w \in \{0,1\} \qquad \forall w \in W \tag{3.21}$$

The constraint $\sum_{w\in W} y_w \leq B_a$ must be satisfied with equality because $t_w(1-x_w) \geq 0$ for all $w \in W$. The defender's problem is to minimize the sum of B_a largest linear functions $t_w - t_w x_w$ among the $n = |W|$ of them, subject to the polyhedral constraints on x_w. This problem can be solved as a single LP [112] as follows.

$$\min_{d^+,x,z} \quad B_a z + \sum_{w\in W} d_w^+ \tag{3.22}$$

$$\text{s.t.} \quad d_w^+ \geq t_w - t_w x_w - z \qquad \forall w \in W \tag{3.23}$$

$$\sum_{w\in W} c_w t_w x_w \leq B_d \tag{3.24}$$

$$x_w \in [0,1],\ d_w^+ \geq 0 \qquad \forall w \in W \tag{3.25}$$

□

Algorithm 2: CyberTWEAK

```
1 Remove $D \longleftarrow$ Find-dominated-websites() from $W$.
2 Get heuristic defender strategy $\hat{x}^*$ by solving $\hat{\mathcal{P}}_1$.
3 if $OPT(\mathcal{P}_2(\hat{x}^*)) \le OPT(\tilde{\mathcal{P}}_3(\hat{x}^*))$ then return $\hat{x}^*$ ;
4 Initialize max effort vector set $e^{\mathcal{A}} = e^{\mathcal{P}_2(\hat{x}^*)}$.
5 while new max effort vector was added to $e^{\mathcal{A}}$ do
6     $x \longleftarrow$ solution of $P_1^{\mathrm{LP}}(e^{\mathcal{A}})$.
7     $e \longleftarrow$ solution of $\mathcal{P}_2(x)$.
8     Add $e$ to $e^{\mathcal{A}}$.
```

3.5.2 CyberTWEAK

For the general SED games, we propose a novel algorithm CyberTWEAK (Alg 2). It first computes an upper bound for $\mathcal{P}_1$ leveraging the dual problem of the linear program (LP) relaxation of $\mathcal{P}_2(x)$. As a byproduct, the computation provides a heuristic defender strategy $\hat{x}^*$ (Line 2). It then runs an optimality check (Line 3) to see if $\hat{x}^*$ is optimal for $\mathcal{P}_1$. When optimality cannot be verified, it solves the original problem $\mathcal{P}_1$ by converting $\mathcal{P}_1$ to an equivalent LP and applying column generation [54], an iterative approach to compute the optimal strategy (Line 5-8). We further improve the scalability by identifying and eliminating dominated website as pre-processing (Line 1). Next we provide details about these steps.

Upper Bound for $\mathcal{P}_1$ Let $\hat{\mathcal{P}}_2(x)$ be the LP relaxation of $\mathcal{P}_2(x)$ and denote the dual variables of the (relaxed) constraints (3.2) ~ (3.5) as $\lambda_1, \lambda_2, \nu, \eta$. We then include the variable x for the defender strategy along with the dual problem, and obtain the minimization problem $\hat{\mathcal{P}}_1$.

$$\hat{\mathcal{P}}_1 \; : \; \min_{x,\lambda,\nu,\eta} B_e\lambda_1 + B_a\lambda_2 + \sum\nolimits_{w\in W} \eta_w \tag{3.26}$$

$$\text{s.t. } \kappa_w(1 - x_w) \le \lambda_1 + \nu_w, \quad \forall w \in W \tag{3.27}$$

$$\pi_w\lambda_2 - t_w^{all}\nu_w + \eta_w \ge 0, \quad \forall w \in W \tag{3.28}$$

$$\sum\nolimits_{w\in W} c_w t_w x_w \le B_d \tag{3.29}$$

$$x_w \in [0,1],\ \lambda_1, \lambda_2, \nu_w, \eta_w \ge 0,\ \forall w \in W \tag{3.30}$$

$\hat{\mathcal{P}}_1$ is an LP which can be solved efficiently. In addition, $\hat{x}^*$ in the optimal solution for $\hat{\mathcal{P}}_1$ is a feasible defender strategy in the original problem $\mathcal{P}_1$. Therefore, solving $\hat{\mathcal{P}}_1$ leads to a heuristic defender strategy as well as bounds for the optimal value of $\mathcal{P}_1$. Denote the optimal value of a problem $\mathcal{P}$ as OPT($\mathcal{P}$). We formalize the bounds below.

Theorem 3.5.5. *If $B_e \ge \max_w t_w^{all}$, $OPT(\hat{\mathcal{P}}_1) \le 3OPT(\mathcal{P}_1)$.*

Proof. Let x^* be the optimal solution to $\mathcal{P}_1$. Consider the problem $\hat{\mathcal{P}}_2(x^*)$. At optimal solution, the inequality $e_w \le t_w^{all} \cdot y_w$ in $\hat{\mathcal{P}}_2(x^*)$ is satisfied with equality, as if $e_w < t_w^{all} \cdot y_w$, then we can decrease y_w without changing the objective value and violating any constraints. Then, we can eliminate the variables e_w and $\hat{\mathcal{P}}_2(x^*)$ becomes a standard two-dimensional fractional knapsack problem $\hat{\mathcal{P}}_4(x^*)$. It is well-known that there exists an optimal solution to $\hat{\mathcal{P}}_4(x^*)$ which has at most 2 fractional values y_{w_1} and y_{w_2} [77]. We have

$$\begin{aligned} &OPT(\hat{\mathcal{P}}_1) \leq OPT(\hat{\mathcal{P}}_2(x^*)) = OPT(\hat{\mathcal{P}}_4(x^*)) \\ &\leq OPT(\mathcal{P}_2(x^*)) + t_{w_1}(1 - x^*_{w_1}) + t_{w_2}(1 - x^*_{w_2}) \\ &\leq 3OPT(\mathcal{P}_2(x^*)) = 3OPT(\mathcal{P}_1) \end{aligned}$$

Note that if $B_e = \infty$, $\hat{\mathcal{P}}_1$ is a 2-approximation. □

Theorem 3.5.6. *Let x^*, $\hat{x}^*$ be an optimal solution to $\mathcal{P}_1$, $\hat{\mathcal{P}}_1$.*

$$OPT(\mathcal{P}_1) \leq OPT(\mathcal{P}_2(\hat{x}^*)) \leq OPT(\hat{\mathcal{P}}_1) \leq OPT(\hat{\mathcal{P}}_2(x^*)).$$

Proof. Since $\hat{x}^*$ and its best response calculated by $\mathcal{P}_2(\hat{x}^*)$ form a feasible solution to $\mathcal{P}_1$, the first inequality holds. For any defender strategy x, OPT($\mathcal{P}_2(x)$) ≤ OPT($\hat{\mathcal{P}}_2(x)$) as adversary can choose fractional y_w's in $\hat{\mathcal{P}}_2(x)$. For $\hat{x}^*$ specifically, we have OPT($\hat{\mathcal{P}}_2(\hat{x}^*)$) = OPT($\hat{\mathcal{P}}_1$), since $\hat{\mathcal{P}}_1$ is, by strong duality, equivalent to $\mathcal{P}_1$ except that the adversary is allowed to choose fractional y_w's. This establishes the second inequality. The last inequality holds because x^* and its fractional best response calculated by $\hat{\mathcal{P}}_2(x^*)$ form a feasible solution to $\hat{\mathcal{P}}_1$. □

Optimality Conditions for $\hat{x}^*$ We present a sufficient condition for optimality, which leverages the solution of the following LP $\tilde{\mathcal{P}}_3(\hat{x}^*)$.

$$\tilde{\mathcal{P}}_3(\hat{x}^*) : \min_{x,v} \quad v \tag{3.31}$$

$$\text{s.t.} \quad v \geq \sum\nolimits_{w \in W} \kappa_w (1 - x_w) e_w, \ \forall e \in e^{\mathcal{P}_2(\hat{x}^*)} \tag{3.32}$$

$$\sum\nolimits_{w \in W} |x_w - \hat{x}^*| \leq \epsilon \tag{3.33}$$

Constraints (3.7) ~ (3.8)

ϵ is an arbitrary positive number and $e^{\mathcal{P}_2(\hat{x}^*)}$ denotes the set of optimal effort vectors in $\mathcal{P}_2(\hat{x}^*)$. The following claim shows the optimality condition.

Claim 3.5.7. *Given $\hat{x}^*$, an optimal solution to $\hat{\mathcal{P}}_1$, $\hat{x}^*$ is optimal for $\mathcal{P}_1$ if $OPT(\mathcal{P}_2(\hat{x}^*)) \leq OPT(\tilde{\mathcal{P}}_3(\hat{x}^*))$.*

Proof. Suppose $(\hat{x}^*, OPT(P_2(\hat{x}^*)))$ is not an optimal solution for the LP $\mathcal{P}_1^{\text{LP}}(\hat{e}^{\mathcal{A}})$ which is equivalent to $\mathcal{P}_1$. Thus, equivalently $\hat{x}^*$ not optimal for $\mathcal{P}_1$. Any of its neighborhood with radius ϵ contains some $(\hat{x}', v')$ as a better solution, meaning $v' < OPT(P_2(\hat{x}^*))$. This solution $(\hat{x}', v')$ satisfies constraint (3.35), which is strictly stronger than constraint (3.32). Therefore $(\hat{x}', v')$ is feasible for $\tilde{P}_3(\hat{x}^*)$; this contradicts $OPT(\tilde{P}_3(\hat{x}^*)) \geq OPT(P_2(\hat{x}^*))$. □

Clearly, when ϵ is large, $OPT(\tilde{\mathcal{P}}_3(\hat{x}^*))$ is lower and it is harder to satisfy the condition, so in CyberTWEAK, we use a small enough ϵ in $\tilde{\mathcal{P}}_3(\hat{x}^*)$.

Column Generation Define $\hat{e}^{\mathcal{A}}$ as the set of all *max effort vectors* which satisfy $\sum_w e_w = B_e$ and $|W_B| \leq 1$. According to Thm 3.5.2, restricting the attacker to only choose strategies from

$\hat{e}^{\mathcal{A}}$ will not impact the optimal solution for the defender. As a result, $\mathcal{P}_1$ is equivalent to the following LP, denoted as $\mathcal{P}_1^{\text{LP}}(e^{\mathcal{A}})$, when $e^{\mathcal{A}} = \hat{e}^{\mathcal{A}}$.

$$\mathcal{P}_1^{\text{LP}}(e^{\mathcal{A}}) : \min_{x,v} \quad v \tag{3.34}$$

$$\text{s.t.} \quad v \geq \sum\nolimits_{w \in W} \kappa_w (1 - x_w) e_w \qquad \forall e \in e^{\mathcal{A}} \tag{3.35}$$

$$\text{Constraints (3.7)} \sim \text{(3.8)}$$

Although existing LP solvers can solve $\mathcal{P}_1^{\text{LP}}(\hat{e}^{\mathcal{A}})$, the order of $\hat{e}^{\mathcal{A}}$ is prohibitively high, leading to poor scalability. Therefore, CyberTWEAK instead uses an iterative algorithm based on the column generation framework to incrementally generate constraints of the LP. Instead of enumerating all of $\hat{e}^{\mathcal{A}}$, we keep a running subset $e^{\mathcal{A}} \subseteq \hat{e}^{\mathcal{A}}$ of max effort vectors and alternate between solving $\mathcal{P}_1^{\text{LP}}(e^{\mathcal{A}})$ (referred to as the master problem) and finding a new max effort vector to be added to $e^{\mathcal{A}}$ (slave problem). In the slave problem, we solve the adversary's best response problem $\mathcal{P}_2(x)$ where x is the latest defender strategy found. This process repeats until no new effort vectors are found for the adversary. Recall that we get $\hat{x}^*$ and $e^{\mathcal{P}_2(\hat{x}^*)}$ when finding upper bound and verifying optimality of $\hat{x}^*$, which can serve as the initial set of strategies for column generation.

Dominated Websites Not all websites are equally valuable for an organization as some are especially lucrative for an adversary to target. In a Polish bank, many employees may visit the Polish Financial Authority website daily, while perhaps a CS conference website is rarely visited by a banker. Intuitively, attackers will not compromise the conference website and thus, the bank may not need to alter traffic to it. Identifying such websites in pre-processing could greatly reduce the size of our problem. A website w is *dominated by another website u* if the attacker would not attack w unless they have used the maximum effort on u, i.e. $e_u = t_u^{all}$, regardless of the defender's strategy. Thm 3.5.8 presents sufficient conditions for a website to be dominated and leads to an algorithm (Alg. 6) to find dominated website to be eliminated.

Theorem 3.5.8. *Consider websites $u, w \in W$. If the following conditions hold, the website w is dominated by u:*

$$x_u^{max} := B_d/(c_u t_u) \leq 1, \qquad \kappa_w \leq \kappa_u(1 - x_u^{max}),$$
$$\pi_w \geq \pi_u, \qquad t_w^{all} \leq t_u^{all}.$$

Proof. From conditions (1) and (2), we know that for the same amount of effort, the attacker will be better off attacking website u than w, regardless of the defender's strategy.

Suppose $e_w > 0$ and $e_u = 0$ (consequently $y_w = 1, y_u = 0$). Then we could let $e'_w = 0$ and $e'_u = e_w$. This is possible because from condition (4), $e_w \leq t_w^{all} \leq t_u^{all}$ so we have $e'_u \leq t_u^{all}$. Doing this does not increase the attack cost because now $y'_w = 0$ and $y'_u = 1$ and $\pi_w \geq \pi_u$ from condition (3).

Suppose $e_w > 0$ and $e_u > 0$ (consequently $y_w = y_u = 1$). Let $e'_w = e_w - \min\{e_w, t_u^{all} - e_u\}$ and $e'_u = e_u + \min\{e_w, t_u^{all} - e_u\}$. We know that if $e'_w > 0$, then $e'_u = t_u^{all}$. Of course, the attack cost does not increase as well. □

We conclude the section with the following claim.

Claim 3.5.9. *CyberTWEAK terminates with optimal solution.*

Algorithm 3: FIND-DOMINATED-WEBSITES

```
1 Define $U = \{w \in W : c_w t_w \geq B_d\}$. Let $D = \emptyset$.
2 Calculate $x_u^{max} = B_d / c_u t_u, \forall u \in U$
3 foreach website $w \in W$ do
4     Set $U_w = \{u \in U : \kappa_w \leq \kappa_u(1 - x_u^{max})\}$
5     if exists $U_w^* \subseteq U_w$ such that
6     (1) $\sum_{u \in U_w^*} \pi_u \leq \pi_w$, (2) $\sum_{u \in U_w^*} t_u^{all} \geq t_w^{all}$, and (3) $\sum_{u \in U_w^*} t_u^{all} \geq B_e$ then $D = D \cup \{w\}$ ;
7 return set of dominated websites $D$
```

Proof. Claim 3.5.7 has covered the case where CYBERTWEAK terminates after the optimality check on Line 3, Alg. 2. In the other case, CYBERTWEAK terminates when no new effort vectors are found for the adversary. Suppose x is the optimal solution to the defender's optimization problem (Line 6, Alg. 2), and suppose now $\mathcal{P}_2(x)$ does not find a new effort vector (Line 7, Alg. 2). This implies x would still be feasible for the LP $\mathcal{P}_1^{\mathrm{LP}}(e^{\mathcal{A}})$ even if $e^{\mathcal{A}}$ is replaced by the set of all max effort vectors $\hat{e}^{\mathcal{A}}$. Thus, x is an optimal solution. Indeed, at this point the optimal values of $\mathcal{P}_1^{\mathrm{LP}}(e^{\mathcal{A}})$ and $\mathcal{P}_2(x)$ are equal. □

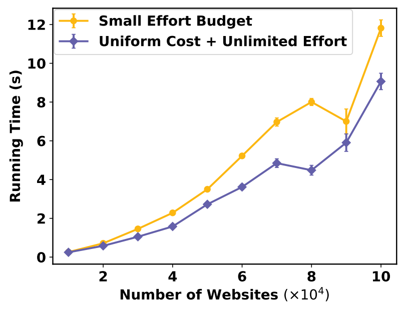
In light of the hardness of the attacker's best response problem (Thm 3.5.1), we also design a variant of CYBERTWEAK, which uses a greedy heuristic to find a new max effort vector to be added in each iteration of column generation (denoted as GREEDYTWEAK). The algorithm allocates the adversary's budget to websites in decreasing order of $r_w = \kappa_w(1 - x_w)\alpha_w$, where α_w is a tuning parameter. Another variant uses an exact dynamic programming algorithm for the slave problem. Details about these variants can be found in Appendix B.1. Also, we note that the SED problem is related to the recent work on bi-level knapsack with interdiction [24]. However, our outer problem of $\mathcal{P}_1$ is continuous rather than discrete, and the added dimension of adversary's effort makes the inner problem $\mathcal{P}_2(x)$ more complicated than that being studied in this work.

3.6 Experiments

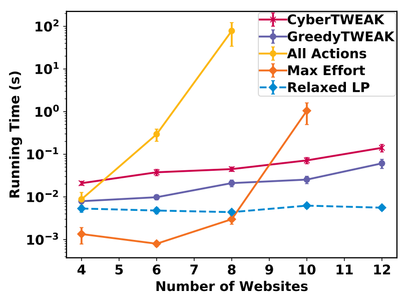
We introduce the experiment results in this section. Our goal is to evaluate the efficiency of CYBERTWEAK on a variety of problem instances, including very large ones. We evaluate the algorithms on these simulated problem instances. Unless otherwise noted, problem parameters are described in detail in Appendix B.3. All results are averaged over 20 instances; error bars represent standard deviations of the mean.

First, we run experiments on the polynomial time tractable cases (Corollary 3.5.3 and Theorem 3.5.4). Fig. 3.2a shows that in both cases, our solution can easily handle 10^5 websites, applicable to real-world corporate-scale problems.

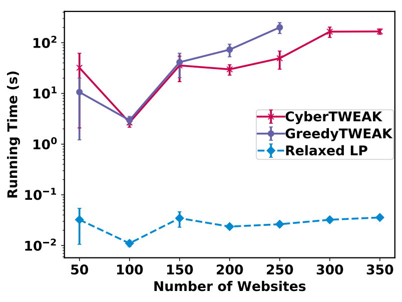
Moving on to the general SED games, we test 3 algorithms (CYBERTWEAK, GREEDYTWEAK, and RELAXEDLP) with two other baselines, MAXEFFORT and ALLACTIONS. RELAXEDLP refers to solving $\hat{\mathcal{P}}_1$. MAXEFFORT solves $\mathcal{P}_1^{\mathrm{LP}}(\hat{e}^{\mathcal{A}})$ directly without column generation. ALLACTIONS



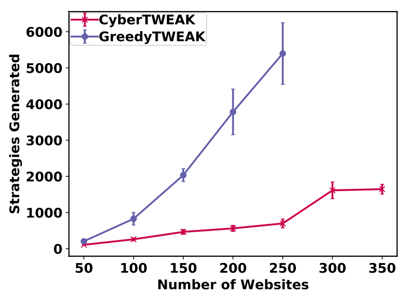
(a) Tractable cases



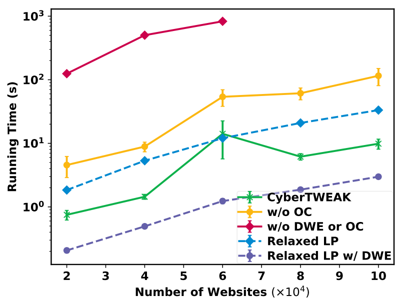
(b) Small instances



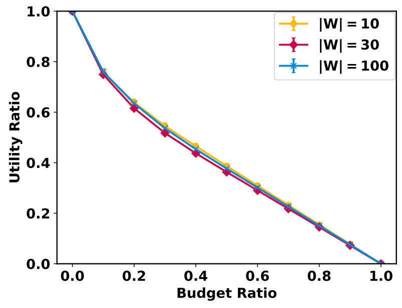
(c) Medium instances running time



(d) Medium instances #strategies



(e) Large instances



(f) Trade-off

Figure 3.2: Experiment results

decomposes SED into subproblems, each assuming some adversary's effort vector is a best response. Its details can be found in Appendix B.1. We test the algorithms with different problem scales. In small and medium sized instances, we skip dominated website elimination (DWE) step (Line 6) and optimality check (OC) step (Line 3) in Alg. 2 as the problem size is small enough, making these steps unnecessary. We use solid lines to represent methods with optimality guarantee and dotted lines for others (RELAXEDLP based methods).

For small instances (Fig. 3.2b), both baselines become impractical even on problems with less than 12 websites. However, CYBERTWEAK is able to find the optimal solutions rather efficiently. GREEDYTWEAK slightly improves over CYBERTWEAK. RELAXEDLP yields the fastest running time, despite a solution gap above 6% as shown in Table 3.1.

For medium-sized instances (Fig. 3.2c), baseline algorithms cannot run and GREEDYTWEAK stops being helpful, mainly because the "better" effort vectors generated in GREEDYTWEAK far outnumbers the "best" effort vectors in CYBERTWEAK (Fig. 3.2d) despite the saved time in each iteration. RELAXED LP has negligible running time and often solves the problem optimally (Table 3.1).

For large instances (Fig. 3.2e), CYBERTWEAK with both DWE and OC steps is able to handles 10^5 websites in 10 seconds. When we remove (denoted as "w\o") DWE and/or OC step, runtime increases significantly, showing the efficacy of these steps[2] Compared to RELAXEDLP

[2]The impact of DWE varies significantly across instances and relies heavily on the distribution of traffic. In less than 4 of the 20 instances DWE did not reduce the problem size by much. We report in Fig. 3.2e the majority group where DWE eliminated a significant number of websites. We provide further discussion in Appendix B.3.

$\lvert W \rvert$	Gap	# Exact	$\lvert W \rvert$	Gap	# Exact
4	13.19%	2/20	150	7e-8	16/20
8	8.11%	5/20	200	8e-10	19/20
12	6.63%	8/20	250	0	20/20
50	2e-6	18/20	300	2e-3	17/20
100	8e-9	19/20	350	2e-8	18/20

Table 3.1: Solution quality of RelaxedLP, with the number of instances where RelaxedLP solves the problem exactly.

or RelaxedLP enhanced with DWE step, which can also efficiently handles 10^5 websites, CyberTWEAK has optimality guarantee.

Finally, we consider the trade-off between the risk exposure and degradation in rendering websites, represented by the objective $OPT(\mathcal{P}_1)$ and defender's budget B_d, respectively. With budget $\bar{B}_d = \sum_{w \in W} c_w t_w$, the attacker would have zero utility. With zero defender budget, the attacker would get maximum utility $\bar{U}$. Fig. 3.2f shows how the utility ratio $OPT(\mathcal{P}_1)/\bar{U}$ changes with the budget ratio $B_d/\bar{B}_d$. As the organization increases the tolerance for service degradation, its risk exposure drops at a decreasing rate.

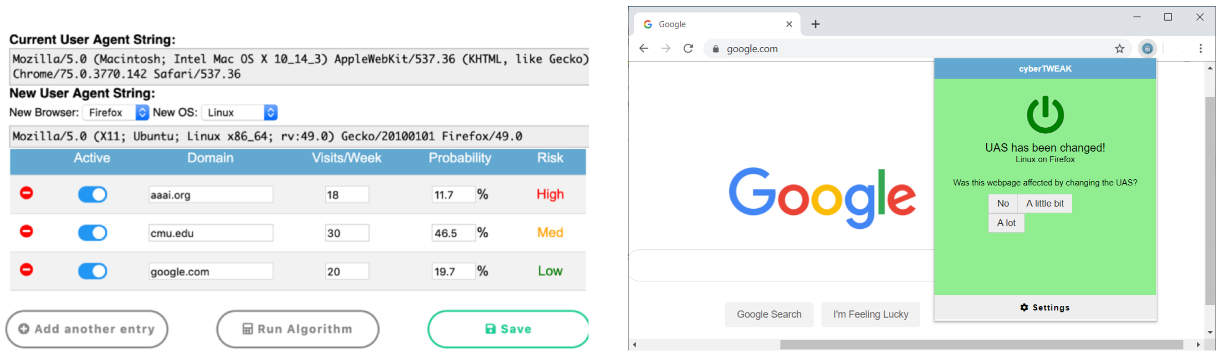
3.7 Deployment

Based on CyberTWEAK, we developed a browser extension (available on the Google Chrome Web Storefootnote 1). It can modify the user-agent string sent to websites automatically during browsing which contains information such as the operating system, browser, and services running on the user's machine. The extension receives from the user the websites visited W, number of visits per week t_w, the cost to alter the user-agent string c_w and budget B_d. The total traffic t_w^{all} and attack cost π_w are estimated from the Cisco Umbrella 1 Million list [33]. The attacker's budgets are set in scale with the previously mentioned parameters. The extension runs CyberTWEAK to set the probability of altering the user-agent string for each website. Note that it is the relative magnitudes, rather than the exact values, that matter.

The extension takes additional steps to make our algorithm more usable and interpretable. First, some users may find it hard to specify the cost of altering user-agent string c_w and budget B_d. Our extension will adjust the values based on the qualitative feedback provided by users about whether the degradation of the website's rendering is acceptable when they visit a website using the modified user-agent, as shown in Fig. 3.3. Second, in addition to showing the computed altering probabilities, the extension also displays a personalized "risk level" for each website, to help the user understand the algorithm's output. Less popular websites frequented more often by the user have higher risk, as shown in Fig. 3.3.

As mentioned in Section 3.4, advanced cyber attackers might sometimes circumvent the existing deception methods. Future versions of the extension will leverage the latest advances in anti-fingerprinting techniques, which entail manipulating more than the user-agent string.

We believe this CyberTWEAK extension is vital to the continued study and development of the countermeasure we develop for this domain and large scale deployments.


Current User Agent String:

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/75.0.3770.142 Safari/537.36
```

New User Agent String:

New Browser: Firefox New OS: Linux

```
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0
```

	Active	Domain	Visits/Week	Probability	Risk
⊖	●	aaai.org	18	11.7 %	High
⊖	●	cmu.edu	30	46.5 %	Med
⊖	●	google.com	20	19.7 %	Low

cyberTWEAK

UAS has been changed!
Linux on Firefox

Was this webpage affected by changing the UAS?

Figure 3.3: The CyberTWEAK browser extension collects and gives qualitative feedback to the user, helping the user decide on deception strategies for each website.

Ethical implications The intended use case of CyberTWEAK is in defending the network of some entity against watering hole attackers. CyberTWEAK involves visibility into the user's web browsing history. In a workplace context, this is of limited concern as it is reasonable to expect that all employees' network traffic is already monitored on corporate network regardless of CyberTWEAK. As for mass use of CyberTWEAK in public, the user should give informed consent prior to use, and the software should keep all activities involving user information locally.

Part II

Learning and Planning for Food Waste and Security

Chapter 4

Improving Efficiency of Volunteer-Based Food Rescue Operations

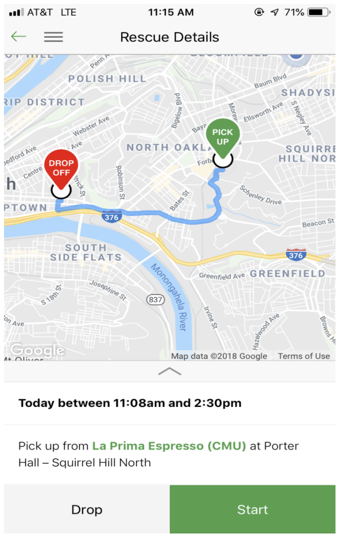
This chapter includes the work from our very first collaboration with 412 Food Rescue to tackle the challenges of food waste and food insecurity. The volunteering nature of food rescues brings significant uncertainty. We predict whether a rescue will be claimed to help the dispatcher better plan for backup options and alleviate their uncertainty. We develop a data-driven optimization algorithm to compute the optimal intervention and notification scheme. The learning and planning, albeit rather detached in this work, lay the foundation for subsequent chapters.

4.1 Introduction

In the US, over 25% of the food is wasted, with an average American wasting about one pound of food per day [36]. Meanwhile, 11.8% of American households struggle to secure enough food at some point [34]. Among the several responses to this inefficient food distribution, food rescue organizations are emerging in many cities. They receive edible food from restaurants and groceries ("donors") and send it to organizations serving low-resource communities ("recipients"). These food rescue organizations are an important force to fight against food waste and food insecurity, both included in the United Nations' Sustainable Development Goals.

A food rescue organization functions as a platform between the donors and the recipients. Upon receiving the notice from a donor, the organization matches the food to a recipient. Typically, it transports the food from the donor to the recipient, or stores the food at its own facility if necessary. This incurs cost and there are existing works on optimizing the matching process to minimize this cost [107], and some attempt to create a market [126]. However, many of these organizations operate under tight budget and human resource constraints. As a result, some outsource the transportation of food to local volunteers, which brings in a new dimension to the problem.

We collaborate with 412 Food Rescue (412FR), a food rescue organization serving over 1000 donors and recipient organizations in Pittsburgh, US. The dispatcher at 412FR matches the food by calling each recipient till some recipient accepts. The dispatcher determines the order of these calls based on numerous factors such as the proximity between the donor and recipient

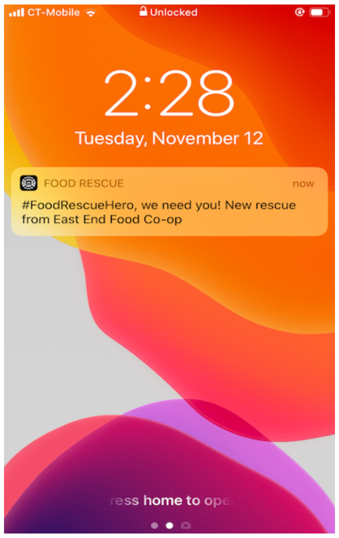


Today between 11:08am and 2:30pm

Pick up from La Prima Espresso (CMU) at Porter Hall – Squirrel Hill North

(a) Screenshot of a rescue task

(b) Donation picked up at a cafe



#FoodRescueHero, we need you! New rescue from East End Food Co-op

(c) App notification

Figure 4.1: Volunteers for 412 Food Rescue receive push notifications about new rescue opportunities, get detailed information about the rescue on the app, and then head out to complete the rescue.

and the estimated recipient's willingness to accept the food. This decision is not hard-coded but depends on the dispatcher's rich experience. After the matching, they post the rescue on 412FR's smartphone apps. 412FR's over 7000 volunteers can see the rescue's start and end location as well as the weight and type of food (Fig. 4.1a). A volunteer can claim the rescue on the app and then complete the rescue by picking up the food from the donor within its pickup window (Fig. 4.1b) and delivering it to the recipient.

Relying on volunteers saves cost for the organization, but there is a high degree of uncertainty in whether a rescue will be claimed and completed. Over the years, 412FR has used many methods to get more rescues claimed and completed. First, after a rescue request is posted, the app will push notification to volunteers within 5 miles (Fig. 4.1c). After 15 minutes, if no one has claimed the rescue, the app will push notification to all available volunteers. Second, dispatcher monitors all rescues to be claimed or to be completed. If no one has claimed a rescue by the last hour of its pickup window, the dispatcher calls *regular* volunteers they are familiar with to help with the rescue. For the ones claimed and to be completed, dispatcher needs to answer volunteers' inquiries about delivery details in real-time. As such, the dispatcher has a heavy workload. However, while it is helpful to engage with the volunteers, too many notifications might drive them away [48].

Our contribution In this chapter, we aim to reduce the dispatcher's workload and the redundant notifications sent to the volunteers, without decreasing the claim rate of the rescues. We make two main contributions. 1) We train a stacking model to predict whether a rescue will be claimed. Our stacking model achieves an AUC of 0.81, serving as a reliable reference of the risk of a rescue. The model informs the dispatcher how likely a rescue is going to be claimed, thus helping the dispatcher better plan for backup options. (2) We perform data-driven optimization to find the optimal *Intervention and Notification Scheme* (INS), i.e., when the dispatcher should intervene and seek help from regular volunteers and when and to whom the notifications should be sent. We estimate the counterfactual rescue outcomes and use a branch and bound method to improve computational efficiency. The resulting INS can improve over the current practice by reducing the number of notifications sent and the dispatcher interventions, while keeping the rescues' expected claim rates. Our analysis suggests to the platform some changes in their current INS, which can save the most valuable resources to food rescue: the dispatcher's attention and volunteers' interest.

We are working with 412FR to deploy our results. In fact, such organizations are not rare at all. In the US alone, similar organizations are already operating in over 55 cities, helping over 11 million people, and the numbers will only keep growing. Thus, our work could potentially improve the dispatching decisions at a large scale, not to mention the similar volunteer-based community services other than food rescue.

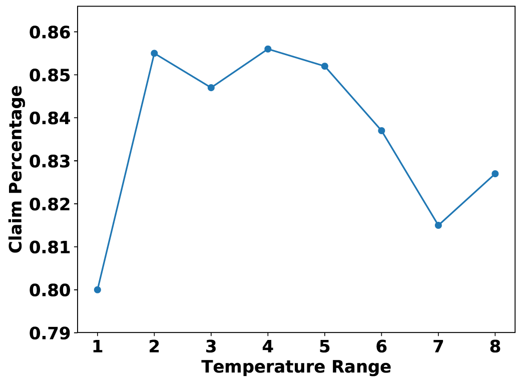
4.2 Related Work

The operational challenges of food rescue organizations have received much attention. Nair et al. [107] and Gunes et al. [59] study matching the donor and recipient with a routing problem. This is related to the more general problem of online matching [75, 101]. Prendergast [126] and Lundy et al. [95] consider the incentive of the agencies and design a market for the food rescue platform. Phillips et al. [120] explore predicting the future donations. However, all these works assume the organization manages the donations without the participation of volunteers, and thus they are not applicable to our problem. In addition, the well-studied task allocation problem [64] does not perfectly fit our scenario, as 412FR has no control over the volunteers. The only work which assumes similar operation is [83]. It studies the stakeholders' perception of fairness and democracy on the food matching decisions, while we focus on improving the efficiency of food rescue operations.

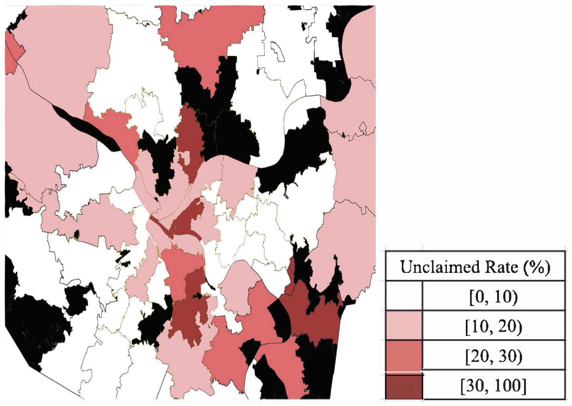
Our prescriptive analysis uses counterfactual estimation of rescue outcomes under various dispatching schemes. This relates to the extensive literature on causal inference with observational data [39]. However, 412FR has always used the same dispatching scheme for all rescues, and it is currently impossible to contact volunteers for pre- and post-intervention tests [125]. Thus, existing work is not applicable and we develop a new way of constructing counterfactual datasets.

4.3 Predicting the Claim of Rescues

Our first task is to predict whether a rescue would be claimed. We use the operational dataset of 412FR which contains rescues from March 2018 to May 2019. The dataset records the time log of each step in the rescue: posting, claimed by volunteer, and completion, along with the ID of the volunteer who claimed the rescue. We treat a rescue as unclaimed and assign a negative label if it was never claimed or if it was claimed within the last hour of the pickup window by



(a) Claim rate vs. temperature.



(b) Percentage of unclaimed rescues by zip code district.

Figure 4.2: Data analysis results. The temperature range i represents $(10.5i - 11.5, 10.5i - 1]$°F.

a selected group of volunteers who had done more than 10 rescues within the last two months. We assume the latter ones had gone through dispatcher's intervention and would not have been claimed otherwise. The dataset contains 4574 rescues with 749 negative ones among which 672 were not claimed by anyone and 77 were claimed within the last hour of the pick up window by the selected group.

4.3.1 Feature engineering

We use a number of features for the prediction. The first group of features are directly related to the rescue, such as the travel time and distance between the donor and the recipient generated by Google Maps Platform, the weight of the food, time of day, and which time slot the rescue belongs to.

We also used the weather information on the day of rescue from Climate Data Online, including the average temperature, precipitation and snowfall, as data analysis suggests that weather is correlated with the rescue outcome (Fig. 4.2a).

The third group of features involve the number of available volunteers near the donor and recipient's locations. Instead of using zip code, we evenly divide the area of operation of 412FR into a grid with 300 cells because the zip code districts vary a lot in size (Fig. 4.2b) and the grid allows for better specificity. Each volunteer could set in their app the time slots they do not want to receive any notifications, which can also be interpreted as their availability. An *active* volunteer (AV) in a grid cell for a rescue is one who had done a rescue in the cell and marked themselves as available for the rescue's pick-up time slot in the app. We use as feature the number of AVs in the donor's and recipient's cells and the number of them averaged over the cells adjacent to the donor's. The number of AVs who indicate they have vehicles is helpful as well, as those without vehicles might be more constrained in their choice of rescues.

We also tested some other features such as average household income and vehicles. However, they do not improve the performance of the model. An example of the features we use for the training the machine learning model is shown in Table 4.1. These two data points are for

Features	Rescue 1	Rescue 2
Fastest travel time of rescue	8 min	28 min
Travel distance of rescue	2.4 miles	18 miles
Weight of the food	5 lb	20 lb
Time of day	1pm	2pm
Time Slot	Weekday Afternoon	Weekend Afternoon
Precipitation	0	0.12 inch
Snowfall	0	0
Average temperature	62 $^{\circ}F$	76 $^{\circ}F$
AVs in donor's cell	20	91
Average AVs in donor's neighboring cells	40	250
AVs in recipient's cell	30	300
AVs in donor and recipient's cells with vehicle	21	116

Table 4.1: Two example data points for the predictive model.

illustration purpose and are not real rescues, as per our agreement with 412FR.

4.3.2 Stacking Model

We first attempted a few baseline models including Gaussian Process (GP) and Random Forest (RF) with different parameters but got unsatisfying performance, especially with the false positives, i.e. when the rescue is unclaimed but we predict it as claimed. In the context of food rescue, we want to inform human dispatchers which rescues will be unclaimed without human intervention and need extra attention. Thus, false positives can be costly because it may lead to the ignorance of a rescue in need of intervention and the waste of donated food, while false negatives are less concerning because it only leads to unnecessary extra attention from the human dispatcher. To deal with weak learners, we use a stacking approach inspired by [159], whose structure is shown in Fig. 4.3. First, we split the training data into two sets, D_A and D_B. We use D_A to train various base models (Fig. 4.3, ①) and then we use these trained models to make predictions on D_B (Fig. 4.3, ②). Finally, we train a meta learner using the base models' predictions on D_B to determine the stacking model's estimate (Fig. 4.3, ③). In our case, we use 5 GP regressors and 1 RF classifier as the base model. The 5 GPs have different kernels and parameters for length scales. The parameters for GPs are shown in Table 4.2. The Random Forest Classifier has 100 estimators and the max-depth for any decision tree is 9.

All the six models are trained on the same data D_A. We use the mean values of the GPs' predictions and the binary label of the RF classifier, on D_B, as the input to the neural network meta learner. We report the results in Sec. 4.5.

GP	1	2	3	4	5
Kernel	DP	DP	Matern	RBF	RBF
Alpha	0.5	0.01	0.3	0.1	0.03

Table 4.2: GP parameters. Alpha is the dual coefficient of training data points in kernel space. DP means dot product.

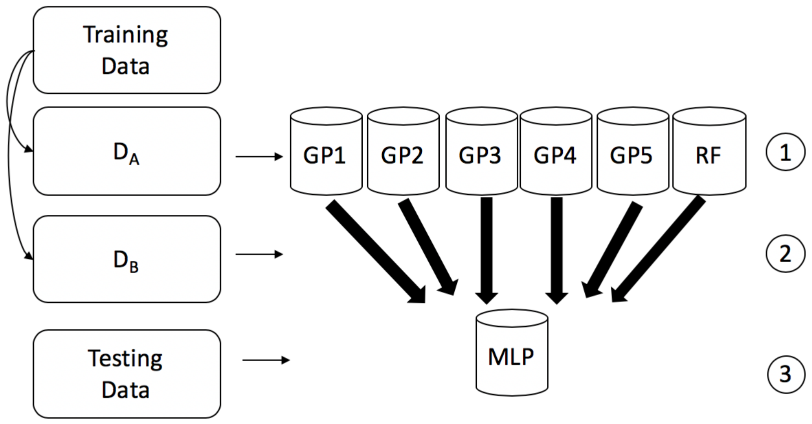


Figure 4.3: The stacking model.

4.4 Optimizing Intervention and Notification

We also perform prescriptive analysis to optimize the INS of 412FR, determining the guideline for dispatcher intervention and the rules for sending notifications. Our goal is to reduce the frequency that the dispatcher intervenes to "save" a rescue, or the mobile app notifications sent, ideally both.

We formalize the problem by defining an INS as a tuple (x, y, z), with x, y, z described below. When a rescue is posted, the mobile app first sends notifications to the volunteers who are within y miles from the donor. If no volunteer claims the rescue within the first x minutes, the app then sends the notification again to all volunteers who have indicated availability in the corresponding time slot. The dispatcher monitors the rescue after it is posted. If a rescue has not been claimed by z minutes before its pickup deadline, the dispatcher intervenes by directly contacting a group of regular volunteers and asking them if they are willing to claim it. If w_r is the duration from the posting time to the pickup deadline of rescue r, then the dispatcher intervenes $w_r - z$ minutes after the rescue is posted. We assume that upon the dispatcher's intervention, with probability μ the rescue immediately gets claimed, otherwise it has no effect.

412FR has always used a default INS: $\hat{x} = 15$ (minutes), $\hat{y} = 5$ (miles), and $\hat{z} = 60$ (minutes). We look for the optimal INS in a finite set S of candidate INSs which minimizes

$$\lambda \mathbb{E}_{r\sim R}[c_1(x, y, z, r)] + \mathbb{E}_{r\sim R}[c_2(x, y, z, r)] \tag{4.1}$$

where R is the distribution of rescues, and λ controls the trade-off between two quantities: the expected number of dispatcher interventions and the expected number of notifications sent to volunteers. c_1 is the average number of dispatcher intervention for rescue r, c_2 is the average

Notation	Meaning
x	Second round notification time, default $\hat{x}$
y	First round notification radius, default $\hat{y}$
z	Intervention time from deadline, default $\hat{z}$
μ	Dispatcher intervention success probability
r	r.v.: a rescue, following distribution R.
w_r	Duration from r being posted to deadline
λ	Trade-off of intervention and notification.
$s(\cdot)$	Average number of dispatcher interventions
$v(\cdot)$	Average number of 1st round notifications
$q(\cdot)$	Average number of 2nd round notifications
$p(a, \cdot)$	Proportion of rescues claimed in a minutes
S	Domain of optimization variables (x, y, z)
b_i	Claim rate lower bound

Table 4.3: Notations for the optimization problem.

number of notifications sent for r given the INS. We also want to maintain a high claim rate, i.e., $E_{r\sim R}[c_3(a_i, x, y, z, r)] \geq b_i$ for a given set of a_i, b_i where c_3 is the probability that rescue r is claimed within first a_i minutes.

Without knowing the exact distribution R, we can only estimate these expected values through data. Given a dataset D of rescues under INS (x, y, z), we define $p(a, x, y, z)$ as the proportion of rescues in D that are claimed in a minutes; $s(x, y, z)$ as the proportion of rescues in D that are not claimed by volunteers before the dispatcher intervenes; $v(y)$ as the average number of available volunteers who are within y miles of the donor who receive the first round notifications; $q(x, y, z)$ as the average number of available volunteers who receive the second round notifications. Formally,

$$\begin{aligned}
p(a, x, y, z) &= \frac{1}{|D|} \sum\nolimits_{r \in D} \mathbb{I}\left(\text{rescue } r \text{ claimed in } a \text{ min}\right), \\
s(x, y, z) &= \frac{1}{|D|} \sum\nolimits_{r \in D} \mathbb{I}\left(r \text{ not claimed in } w_r - z \text{ min}\right), \\
v(y) &= \frac{1}{|D|} \sum_{r \in D} \text{\# available volunteers within } y \text{ miles of } r \\
q(x, y, z) &= \frac{1}{|D|} \sum_{r \in D} \mathbb{I}\begin{pmatrix} r \text{ not claimed} \\ \text{in } x \text{ min} \end{pmatrix} \times \begin{matrix} \text{\# available} \\ \text{volunteers for } r \end{matrix}
\end{aligned}$$

Assuming data points in D are sampled from R, we have

$$\begin{aligned}
\mathbb{E}_{r\sim R}[c_1(x, y, z, r)] &\approx s(x, y, z) \\
\mathbb{E}_{r\sim R}[c_2(x, y, z, r)] &\approx v(y) + q(x, y, z) \\
\mathbb{E}_{r\sim R}[c_3(a, x, y, z, r)] &\approx p(a, x, y, z)
\end{aligned}$$

Our final optimization problem is as follows.

$$\min_{x,y,z} \quad C(x,y,z) = \lambda s(x,y,z) + v(y) + q(x,y,z) \tag{4.2}$$

$$s.t. \quad p(a_i, x, y, z) \geq b_i, \quad \forall i \in I \tag{4.3}$$

$$(x,y,z) \in S$$

From the historical data and dispatcher's advice, we could estimate μ, V_y, a_i, b_i, S. However, estimating $s(\cdot)$, $q(\cdot)$, $p(\cdot)$ poses significant difficulty. We need to estimate the counterfactual claim time (CCT) for all INSs $(x,y,z) \neq (\hat{x},\hat{y},\hat{z})$.

4.4.1 Counterfactual claim time (CCT) estimation

Given a rescue happened under the default INS $(\hat{x},\hat{y},\hat{z})$, we estimate its CCT under some other INS (x,y,z). We make the following assumptions.

- No matter when a volunteer receives the notification, upon receiving it they take the same amount of time to respond, and the effect of human intervention is independent of the app notification.
- The intervention outcome is not affected by the INS.
- Given a list of regular volunteers (provided by dispatchers or derived from data), if a rescue is recorded in the historical data as claimed by a regular volunteer after the dispatcher intervention time, i.e., $w - \hat{z}$ minutes after the rescue is posted, we give the credit to dispatcher intervention. If a rescue was claimed after the dispatcher intervention time by anyone else, we assume that the dispatcher's intervention have failed.

Suppose the rescue was claimed by volunteer i located d miles from the donor in the historical data. At a high level, in most cases we compute the claim time of volunteer i in the new INS (x,y,z) and take that as our CCT estimate. For example, suppose i is within the first round notification radius, i.e. $d \leq \hat{y}$ and claims the rescue in 7 minutes under $(\hat{x},\hat{y},\hat{z})$. This rescue would have a CCT of 12 minutes when $x = 5, z = \hat{z} = 60$ and $y < d \leq \hat{y}$, i.e., i is now outside the first round notification radius. This is because the volunteer i needs 7 minutes to respond after getting notification, but now they only receive the notification 5 minutes after the rescue is available. We also factor in the effect of dispatcher intervention when the intervention happens before the CCT k, i.e. $w_r - z < k$. For rescue r, we report the expected claim time $m_z(k) = \mu \min\{w_r - z, k\} + (1-\mu)k$. In another scenario, if in the historical data, volunteer i who is not in the first round notification radius claims the rescue before the second round notification, we assume the volunteer's action is due to actively checking the available rescues and is not affected by the notification. Thus, the CCT remains the same for all INS. The complete computation is shown in Fig. 4.4.

We claim that our estimation is conservative, i.e., we will never underestimate the claim time. This is important in practice, because overestimation may merely lead to unnecessary resource spent but underestimation may cause a rescue to fail. Our estimation is accurate when i is within the first round notification radius in the counterfactual INS but not in the default INS and intervention happens after the claim time, as i would still be the first volunteer to claim

Dispatcher intervention	Notification radius	Distance	Counterfactual claim time
$a < w_r - \hat{z}$	$y \leq \hat{y}$	$d \leq y$	$m_z(a)$
		$d \in (y, \hat{y}]$	$m_z(a + x)$
		$d > \hat{y}$	$m_z(a + x - \hat{x})$
	$y > \hat{y}$	$d \leq y$	$m_z(a)$
		$d \in (\hat{y}, y]$	$m_z(a - \hat{x})$
		$d > y$	$m_z(a + x - \hat{x})$
$a \geq w_r - \hat{z}$, by regular volunteer	any	any	z
$a \geq w_r - \hat{z}$, by other volunteer	$y \leq \hat{y}$	$d \leq y$	a
		$d \in (y, \hat{y}]$	$a + x$
		$d > \hat{y}$	$a + x - \hat{x}$
	$y > \hat{y}$	$d \leq y$	a
		$d \in (\hat{y}, y]$	$a - \hat{x}$
		$d > y$	$a + x - \hat{x}$

Figure 4.4: Construction of the CCT for INS (x, y, z) based on default INS $(\hat{x}, \hat{y}, \hat{z})$. a is the rescue's actual claim time. d is the distance from the rescue's volunteer to the donor.

the rescue under the counterfactual INS. In some other cases, there exists the unobservable possibility that some other volunteer might claim the rescue before i in the counterfactual INS, and hence we might overestimate the claim time.

4.4.2 Solving the optimization problem

Given the CCT estimate for each rescue, we can estimate the functions $s(\cdot), q(\cdot), p(\cdot)$ using the counterfactual dataset. However, there is no closed-form expression for them. Computing their values at every point in a brute force way is obviously inefficient. We propose a branch-and-bound algorithm and a feasibility check to find optimal INS more efficiently.

First, we note that the CCT, as detailed in Fig. 4.4, is increasing in x and decreasing in y and z. Since $p(\cdot)$ is the empirical estimate based on the claim time, if some infeasible INS (x, y, z) does not satisfy claim rate constraint (4.3), any INS $(\tilde{x}, \tilde{y}, \tilde{z})$ with $\tilde{x} \geq x, \tilde{y} \leq y, \tilde{z} \leq z$ is also infeasible. Thus, we need not generate CCT for $(\tilde{x}, \tilde{y}, \tilde{z})$.

Using a similar observation, we devise our main algorithm, Alg. 5. Note that $s(x, y, z)$ decreases as x, z decreases and y increases, $v(y)$ decreases as y decreases, $q(x, y, z)$ decreases as x, y, z increases. Therefore, if we replace all the variables in all terms with the extreme values in domain S that can minimize $C(x, y, z)$ (as shown in Table 4.4), we get a lower bound of $C(x, y, z)$. We define a subproblem as the original optimization problem with k of the variables

$s(x, y, z)$	x_{min}	y_{max}	z_{min}
$v(y)$		y_{min}	
$q(x, y, z)$	x_{max}	y_{max}	z_{max}

Table 4.4: Replace (unspecified) variables in each term with the extreme values to get a lower bound.

Algorithm 4: Solve-Relaxation

```
1  Optional input arguments: $x, y, z$
2  if all of $x, y, z$ specified then
3      Generate CCTs with $(x, y, z)$.
4      if feasible then
5          Compute cost $\bar{C} = C(x, y, z)$
6          return subproblem $(\bar{C}, (x, y, z))$
7  else
8      Generate CCTs with unspecified parameter replaced by extreme values in Table 4.4.
9      Compute lower bound $\bar{C}$
10     return subproblem $(\bar{C}, (x, y, z))$
```

in the INS specified and the remaining ones unspecified for $k = 0, 1, 2, 3$. To compute a lower bound for each subproblem, we replace the unspecified variables in each term with the extreme values according to Table 4.4. For example, if z is specified, and x, y are unspecified, we get a lower bound

$$\bar{C} = \lambda s(x_{min}, y_{max}, z) + v(y_{min}) + q(x_{max}, y_{max}, z)$$

In Alg. 5, we start with the original problem where none of the variables are specified ($k = 0$). We branch to lower level subproblems in the order of $z \longrightarrow y \longrightarrow x$, as this order tends to prune the fastest. For each subproblem, we either compute a lower bound, or when all variables are specified, compute the exact cost. We generate one counterfactual dataset for computing the exact cost (Line 3, Alg. 4), and at most two datasets when computing the lower bound (Line 8, Alg. 4), since $s(\cdot)$ and $z(\cdot)$ are minimized at two different INSs and $v(\cdot)$ does not depend on the CCT. The implicit pruning on Line 3 guarantees Alg. 5 finds the optimal solution.

4.5 Results

4.5.1 Prediction

We "predict the future with the past". As mentioned in Section 4.3, we treat rescues done by volunteers who have done over 10 rescues in the last two months in the last hour of the pick-up window as unclaimed. Thus, we exclude the rescues in the first two months from our prediction task as we do not have the volunteer history for these early entries. As a result, the training data consist of rescues from May 2018 to December 2018 and testing data consist of rescues

Algorithm 5: Branch-and-Bound

```
1 Push Solve-Relaxation({}) to Frontier.
2 while Frontier set is not empty do
3     Get subproblem with lowest $\bar{C}$ from Frontier.
4     if subproblem has all parameters specified then
5         return $(\bar{C}, (x, y, z))$    #(optimal solution)
6     else
7         Follow the order $z \longrightarrow y \longrightarrow x$ to expand the node, i.e., if the first $k$ variables are
            already specified, create a subproblem for each possible value of the $(k+1)^{th}$
            variable in $S$.
8         Add all subproblems Solve-Relaxation$(x, y, z)$ to Frontier.
```

Model	Accuracy	Precision	Recall	F1	AUC
GB	0.73	0.86	0.82	0.84	0.51
RF	0.71	0.87	0.78	0.82	0.54
GP	0.56	0.88	0.54	0.67	0.60
SM	0.69	1.00*	0.64	0.78	0.81

Table 4.5: Performance of selected models, GB: Gradient Boosting Classifier, RF: Random Forest; GP: Gaussian Process; SM: Stacking Model. * We run the experiments for SM for 3 times. The precisions are 1.0, 1.0, 0.9969.

from January 2019 to May 2019. In addition, since the dataset is imbalanced on the number of claimed and unclaimed rescues, we oversample the unclaimed rescues so that the ratio of claimed and unclaimed rescues is 1:1. The oversampling is applied to only the training dataset for the predictive model.

Table 4.5 shows the stacking model outperforms all baseline models. Moreover, it yields almost no false positive errors. This is especially important in the food rescue operation, as the cost of not taking actions to a rescue which turns out unclaimed due to a false positive is much higher than that of an unnecessary dispatcher intervention due to a false negative.

4.5.2 Optimization

After consulting the dispatcher, we take $\mu = 0.4$ as the probability that dispatcher intervention is effective. We require that the optimal INS's claim rate be no worse than default INS. That is, we use $a_i = 1, 2, \ldots 120$ and b_i being the empirical claim rate at the a_i-th minute under the default INS.

First, we demonstrate the effectiveness of the branch and bound algorithm (Alg. 5). We set the domain S as $x, y \in \{2, 4, 6, 8\}$ and $z \in \{30, 40, 50, 60\}$. As shown in Table 4.6, branch and bound needs to generate CCTs on much less INSs than the brute force approach, although advantage is less significant for smaller λ. In the sequel, we use Alg. 5 and set the domain S as $x \in \{1, 1.5, 2, \ldots, 25\}$, $y \in \{1, 1.5, 2, \ldots, 10\}$, $z \in \{30, 32.5, 35, \ldots, 90\}$.

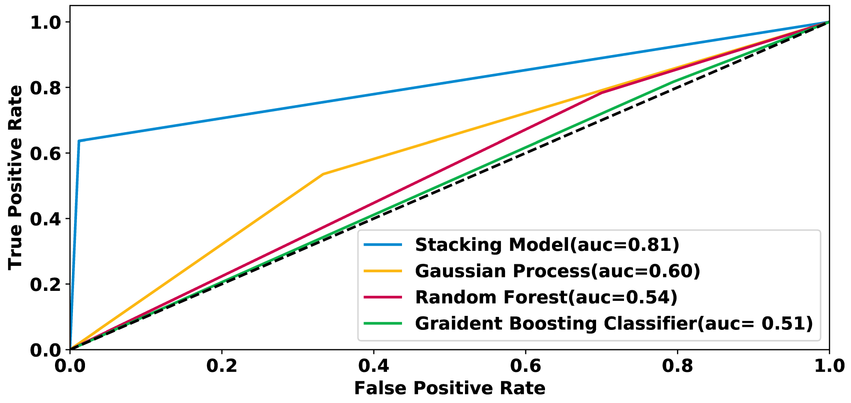


Figure 4.5: The ROC curves of the models

	Brute force search		Branch and bound	
λ	INSs	Time (s)	INSs	Time (s)
10^7	64	192.6	18	65.5
10^6	64	183.7	18	64.9
10^5	64	185.1	16	56.4
10^4	64	190.9	34	125.0
10^3	64	187.1	35	129.3

Table 4.6: Running time and the number of INSs for which the CCTs are generated.

Similar as above, we use the earlier data D_{past} to predict the more recent data D_{future}. First, we focus on computing the optimal INS on D_{past} in Fig. 4.6a. Both the 2nd round notification time and the dispatcher intervention time decrease as λ grows, i.e. the dispatcher's intervention matters more in the dispatching cost. This is aligned with the results in Table 4.4. When the app notification is the primary concern, the default INS is almost desirable, yet if we would like to minimize the interventions, the 2nd round notification needs to go out sooner. Fig. 4.6b, we show the Pareto frontier (in red) of optimizing on D_{past}. The optimal INSs in Fig. 4.6a are now shown in blue. The default INS lies within the frontier, suggesting that the numbers of both interventions and notifications can be improved. The orange rectangle indicates the INS region that is strictly superior to the default INS.

Of course, we would like to examine the quality of the optimization solutions on unseen data. Thus, in Fig. 4.6c, we show the projected number of interventions and notifications on D_{future} of the optimal INSs on D_{past}. For the same INS, the performance is different between Fig. 4.6b and Fig. 4.6c because the claim probabilities are estimated using the two datasets separately. Despite this difference, some optimal INSs on D_{past} still outperforms the current practice. We therefore suggest two INSs (see Fig. 4.6c) to 412 Food Rescue, as shown in Table 4.7. INS A is a strict improvement over the current practice, reducing the number of both intervention and notification. Our second solution, INS B, drastically reduce the labor of dispatcher by 24% at the expense of a mere 2% increase of notifications sent. Since 412FR handles 4574 rescues

INS	Interventions	Notifications
A: (16.5, 5.5, 45)	−13% (−0.06)	0% (−1)
B: (15.5, 5.5, 32.5)	−24% (−0.10)	+2% (+46)

Table 4.7: The projected change in the probability of interventions and number of notifications of the proposed INSs. The numbers in parentheses are the absolute change.

in a 430-day period, INS B can save the dispatcher over 390 times of intervention a year in expectation. An intervention takes the dispatcher at least the same amount of time as matching a new food rescue, and often more. Thus, the dispatcher could handle at least 390 extra rescues a year, which is over 7500 pounds of food by the average donation in our dataset. We choose INS B over the rightmost INS on Fig. 4.6c because 412FR has relatively more shortage of dispatcher than volunteers. Finally, Fig. 4.6d shows our two INSs have competitive claim rates on the unseen data. This suggests the promise of deploying our method in the future.

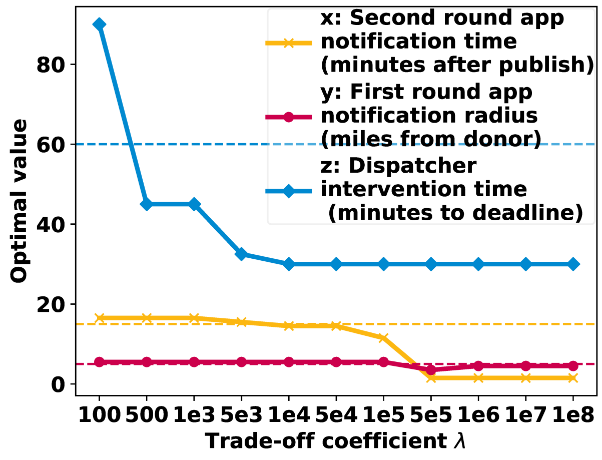
4.6 Discussion

As mentioned in Section 4.2, there are other facets of the food rescue problem that can be tackled with a computational approach, including optimizing the matching between donors and recipients, and directly incentivizing volunteers to be more active. We focus on predicting whether a rescue can be claimed and optimizing INS as they are of higher priority to our partners at 412FR with a clearer path for future deployment. Specifically, the matching is currently done manually at 412FR. Rather than a mechanical process, the dispatcher's job requires a high level of situational judgment, interpersonal skills, and the rapport developed over time. Through our multiple conversations with 412FR and direct experience of shadowing the dispatcher, we believe that currently, automating the matching would not benefit 412FR without a big change of the overall workflow involving all the donors, recipients as well as 412FR. Motivating the volunteers to boost claim rate would require 412FR to take new initiative that is not in place. In contrast, the INS is a current practice and it is easier to test our solution. Nonetheless, we will consider these directions as we continue to work with 412FR.

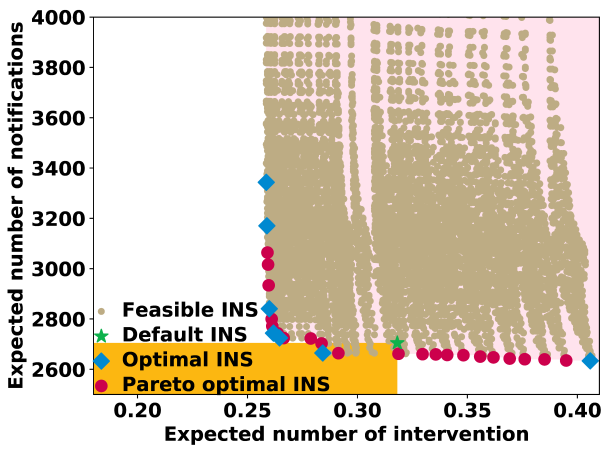
In addition, our optimization framework is already taking into account the volunteer retention problem implicitly as we try to reduce the notifications sent in the objective function. Our framework can be extended to different objectives, and one may design an objective that explicitly focuses on volunteer retention. For example, a user's probability of uninstalling an app could be modeled as a function $f(\cdot)$ of the number of notifications they receive in a week [53]. To minimize the number of volunteers lost, we could change the optimization objective from Eq. (4.1) to

$$\mathbb{E}_{n\sim N}\mathbb{E}_{r_1,\ldots,r_n\sim R^n}\sum_{i\in V}\mathbb{E}_{n_i\sim N_i(\{r_j\},x,y,z)}f(n_i)$$

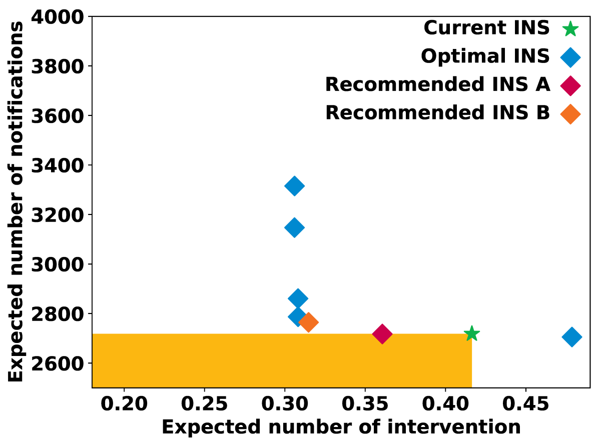
where n is the number of total rescues in a week following some distribution N, V is the set of volunteers, and n_i is the number of notifications volunteer i receives in a week following a distribution determined by the set of rescues and the INS. Similar to Eq. (4.1), this objective value can be estimated through the counterfactual datasets.



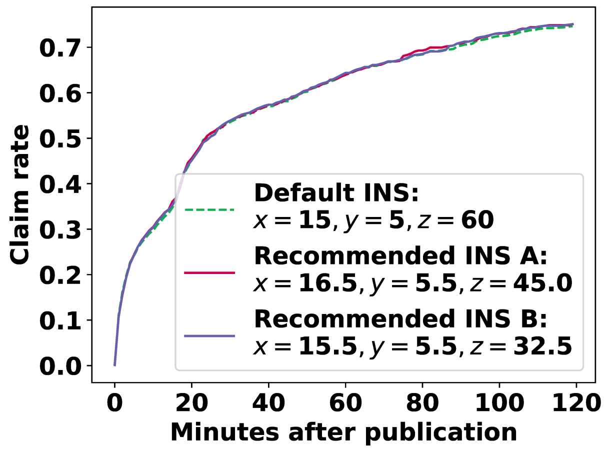
(a) Optimal INSs on D_{past}



(b) Pareto frontier on D_{past}



(c) Performance of the optimal INSs for D_{past} on D_{future}



(d) Projected claim rate of two recommended INSs on D_{future}

Figure 4.6: Experiment results of the data-driven optimization

Another promising direction is to combine our prediction model and the optimization algorithm to optimize rescue-specific INS. We investigate this in the following chapter.

4.7 Conclusion

We provide the first predictive and prescriptive analysis of volunteer-based food rescue operations. Our stacking model predicts the claim status of rescues with AUC of 0.81. Such prediction helps the dispatcher better prepare for interventions and alleviate their uncertainty. Our data-driven optimization reduces the frequency of dispatcher intervention and push notifications sent to volunteers, without harming the claim rate. The dispatcher can use the saved effort to handle an extra 7500 pounds of food a year that would otherwise go to waste. By improving the operation efficiency of inspiring organizations like 412FR, our research contribute to the fight against food waste and insecurity. Please refer to Section 6.1 in Chapter 6 for deployment results of the work in this chapter.

Ethical implications We centered the partner organization in this study, so that the work is intended as a decision aid to the food rescue dispatchers rather than make decisions for them. Meanwhile, it is important to study how the work could affect different regions, and hence demographics, differently. For example, although the prescriptive model in Section 4.4 enforces a threshold that the overall claim rate should not drop, it is critical to investigate if the claim rates in certain regions are improved at the expense of other regions. The use of human subject data in this chapter has been approved by the IRB.

Chapter 5

A Recommender System for Crowdsourcing Food Rescue Platforms

Following the work in the previous chapter, we realized that developing a rescue-specific volunteer engagement system is both necessary and feasible. This naturally became our focus in this chapter. We develop a recommender system to send push notifications to the most likely volunteers for each given rescue. On top of this, we leverage a mathematical programming based approach to diversify our recommendations, and propose an online algorithm to dynamically select the volunteers to notify without the knowledge of future rescues. In this way, we tightly integrate learning and planning in a single algorithm.

5.1 Introduction

Recall the last time you saw several full shelves of bread with an expiry date in two days at a grocery store, or the last time you saw a homeless person downtown asking for a meal. It is not a coincidence if both scenarios seem familiar to you. The simultaneous food waste and food insecurity are a serious problem shared by many parts of the world [35, 58]. Unfortunately, the ongoing COVID-19 pandemic is only making things worse [78]. Even after the pandemic hits its peak, the increased struggle with basic food security will not subside quickly on our long way back to normal. Thus, now more than ever, there is an urgent call for action to address the food security and food waste problem. In this chapter, we leverage our AI expertise to answer this call with our collaborators.

Food rescue organizations (FR) are a major non-profit force in fighting food waste and insecurity. They match fresh, unexpired food from donors to organizations serving low-resource communities, thereby facilitating food redistribution.[1] FRs rely on volunteers to transport the food. To engage with the volunteers, FRs use a web-based application on cell phones to post information of upcoming rescues. Please refer to Section 5.3 for a detailed description of the food rescue operation. This "crowdsourcing" paradigm has proved to be successful in engaging with the general public to address food waste and insecurity [158].

[1]We would like to emphasize that only fresh and unexpired food can be donated through FRs.

However, relying on external volunteers to deliver the food comes with inherent uncertainty. What if no volunteer will claim the rescue? This uncertainty is prevalent in FR operations and it has serious consequences such as lost faith in the program from the donor and recipient organizations. Since the primary way in which FRs contact volunteers is through push notifications, to improve the claim rate one would certainly want to send push notifications to volunteers who are likely to claim the given rescue. Currently, when a rescue is published, the mobile app sends notifications to volunteers who are within a certain radius of the donor. Although being close to the donor is clearly a positive factor, this is far from a perfect solution as its hit ratio is only 44%, which means it misses the "correct" volunteer more than half of the time. On the other hand, we also want to avoid sending push notifications to every volunteer all the time, because that would easily drive them away from the platform or prompt them to disable notifications altogether [48]. A customized push notifications, with good justifications, would better engage the user. Thus, it is crucial to send the push notifications to a selected set of volunteers who are likely to claim the rescue.

Our contributions In this chapter, we propose the first machine learning based model to select the right set of volunteers to notify in the food waste and security domain. By treating each rescue trip as a "user" and each volunteer as an "item", we study this problem from a recommender system perspective. Recommender systems have received a lot of interest from the research community in the past years [43, 62, 87, 170]. Our task is relevant to this literature but also brings several new challenges. We state these challenges and our approaches to address them as follows.

- First, since each rescue only happens once, we stay in the "cold start" phase of the recommender system forever, rendering collaborative filtering-based methods unsuitable. We leverage a sophisticated set of contextual features, an adaptive under-sampling technique, and a neural architecture to develop a content-based recommender system. We show that our model outperforms a number of baselines, and improves the hit ratio of recommending the correct volunteer to 73%. This is a 66% improvement from the current practice which has a 44% hit ratio.
- Second, not being able to recommend diverse items is a serious issue in the recommender systems literature. It is particularly concerning in our application because the "items" are human volunteers who contribute their time to the cause. We leverage a mathematical programming based approach by imposing diversity constraints on the output of the recommender system. This ensures that each volunteer receives only a limited number of push notifications every day.
- Third, most literature on recommender systems assumes an offline environment that has a static dataset. However, food donations arrive sequentially and thus the FR must accordingly make decisions without the knowledge of future rescues. We identify an important arrival pattern of the food rescue trips based on our experience in the domain. Relying on this insight, we develop an online planning algorithm which sequentially selects the volunteers to notify, while still satisfying the diversity constraint we imposed earlier. We show that our online algorithm achieves a hit ratio that is only 10% worse than the hypothetical offline mode where we assume knowledge of all the rescues at the beginning

of the day.

Food rescue organizations have made their presence in most major cities in the US and beyond. In the US alone, there are already over 50 cities where FRs are providing basic necessities to the communities, affecting over a million people. We are working with 412 Food Rescue (412FR), a large food rescue organization in Pittsburgh. [2] Since its incorporation in 2015, 412FR has served over 1,000 nonprofit partners and has grown a network of over 15,000 volunteers in the Greater Pittsburgh Region as of 2020. In Chapter 6, we will introduce the deployment of the work in this chapter. Furthermore, we believe that the problem we tackle is not limited to this particular application: it can be adapted to many domains with a crowdsourcing type of operation that relies on volunteers to perform the task.

5.2 Related Work

There is a growing literature on using AI or related tools to study the food rescue operation. Some formulate a vehicle routing problem to match the donation with recipients [59, 106], while others tackle the problem from a fair allocation and market design perspective [10, 95, 126]. Because the demand and supply or food are external to the FR, some works are focused on forecasting the future food supply [105, 120]. While all these works provide useful insights into the FR operation, the existing literature largely misses the volunteer side of the process. Among the few pieces of work that explicitly consider the volunteer crowdsourcing aspect of food rescue, Lee et al. developed a participatory democracy framework to allow volunteers and other stakeholders to decide on the matching of donations and recipients, which is orthogonal to our work [83]. Shi et al. developed a machine learning model to predict whether a rescue trip will be claimed and an optimization model to find the best intervention scheme [142]. Our work complements theirs and both can be used simultaneously by FRs. Yet we advance from them in two aspects. First, compared to their predictive model as a decision aid for downstream human interventions, our recommender system directly improves the upstream notification process which can reduce the need for the costly human intervention. Second, compared to their prescriptive model which sets system-level notification parameters, our recommender system is rescue-specific, thereby leveraging more information to make better decisions. Finally, Manshadi and Rodilitz design online volunteer notification algorithms in a similar setting [98]. Compared to their work, we take a pure data-centric approach and make few modeling assumptions about the volunteer and rescue patterns, with the sole purpose of finding the most likely volunteers in the real-world use case of this system.

The literature on recommender systems is vast and we will only discuss two topics relevant to our work – cold start and diversity. Cold start refers to the scenario where there is no previous label on a new user or new item [134]. An active approach to deal with cold start would be to interact with the user to request labels, which is often framed as a bandit problem [31, 57, 86, 132, 151, 168]. This is clearly not applicable to our setting, since each rescue is a one-shot business and the stake is too high for real-world trials. Thus, we turn to passive approaches which make do with the data we have. Content-based approaches are a natural candidate for this challenge.

[2] https://412foodrescue.org/

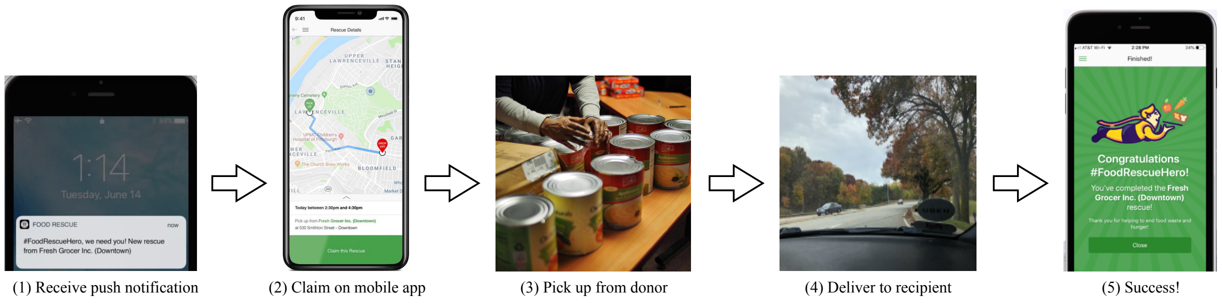


Figure 5.1: The workflow of a food rescue operation from the volunteer's perspective.

Collaborative filtering is not designed to perfectly handle cold start, though there have been methods to enhance it with side information towards addressing this problem [27, 28, 51]. In this chapter, we propose a content-based model for the following reasons. First, in our food rescue setting, cold start is not just a short unpleasant period at the beginning which might imply secondary concern. Instead, we stay in the cold start phase forever because every rescue (user) is new. Thus, handling it perfectly with content-based models is of utmost consideration. Second, we have identified a good set of interaction features based on our experience in the food rescue operation. Third, there is no collaborative filtering-based system currently in place that holds us back from using a content-based approach. Recent advances in leveraging neural architecture in recommender systems serve as a starting point for us to build our model [62].

Our problem is also related to the diversity of recommender systems, which concerns both individual diversity and aggregate diversity. Individual diversity refers to recommending diverse items within the recommended list for each user [167, 170]. Aggregate diversity refers to recommending diverse items between the recommended lists for different users [4, 23, 49, 104, 111]. In our problem, we hope to avoid sending push notifications to a small subset of volunteers (items) all the time. Thus, our problems concerns the aggregate diversity. At the technical level, our method of diversifying the recommendations can be considered as a variant of the integer programming approach by Adomavicius and Kwon [4]. However, our problem has an additional subtlety that makes it more challenging, as we discuss below.

In food rescue, each rescue trip arrives sequentially and thus we need to make the recommendation decision in an online fashion. This brings additional challenge to our effort to diversify the recommendations. This problem resembles the well-studied online adwords matching problem [7, 55, 100], and could fit under the more general online linear programming framework [8]. However, these works typically only guarantee asymptotic results or require prior knowledge of the number of rescues on any given day, which makes them impractical in our setting. There is also a literature on budget pacing in online advertisement [6, 82, 161]. Our application domain and the central problem are different from online advertisement, but our online planning for push notification budget can be considered as a novel way of pacing.

5.3 Anatomy of Food Rescue Operations

Food rescue organizations serve as an intermediary between the food donors and recipient organizations. Donors, typically grocery stores and restaurants, would call the FR when they have food items that they want to donate. After receiving the call, the FR dispatcher matches this donation with some recipient organization, typically some non-profit organization that serves a low-resource community. Once this matching is done, the dispatcher posts this matching on the FR's mobile app. Hereafter, the food rescue process becomes visible to the volunteers. As shown in Figure 5.1, a volunteer, who has the FR's mobile app installed on the phone, will then receive a push notification about the rescue. If they choose to claim it on the app, the app would provide them with the detailed information instructing them where to pick up the donation and where to deliver. The volunteer then goes out to complete the rescue trip.

Of course, the workflow described above is an ideal scenario. In reality, occasionally, some rescue trip stays unclaimed on the mobile app for a long time. FR dispatchers want to prevent this situation as much as possible, since each rescue comes with a deadline which is bounded by the nature of the food and the operation hours of the donor and recipient. Unclaimed rescues discourage the donors and recipients from participating in the program in the future. FRs have two ways to address this problem. First, it sends push notifications to possible volunteers to advertise the rescue. Second, the dispatchers might individually call some regular volunteers to ask for help. In a previous work, Shi et al. focus on the latter approach in order to help the dispatcher's decision-making [142]. We focus on the former by directly finding the best set of volunteers to send push notifications to.

5.4 Data

To develop a recommender system, we need both positive and negative labeled examples. A positive example means that a particular volunteer (item) claims a particular rescue (user); a negative example means otherwise. In this section, we detail our data acquisition, labeling, and feature engineering process.

5.4.1 Positive Labels

We obtained the rescue database from 412FR, covering the period from March 2018 to March 2020. The database keeps the log of each rescue. For most rescues, the database logs its timestamps from being drafted by the dispatcher, to being published on the mobile app, to being claimed and completed by a volunteer. For these rescues, we simply take the rescue plus the volunteer who claimed it as a positive data point. However, the food rescue operation is not always so neat. Occasionally, the dispatcher knows ahead of time that some volunteer would do the job, so they directly assign the volunteer for a particular rescue and bypass the app notification stage. In this case, we take this direct assignment as a positive example as well. Sometimes a volunteer might claim a rescue and then drop it, causing some rescue to have multiple volunteers in the log. In this case, we create our labels based on the last volunteer.

5.4.2 Negative Labels

A negative example means that a particular volunteer did not claim a particular rescue. Since almost all rescues have only one volunteer who claimed the rescue, obviously most of our data points will have negative labels. However, not all of these negative data points are necessarily true, because perhaps a volunteer would have claimed some rescue if someone else had not claimed it 10 minutes in advance. Thus, we use the following ways to construct a selected negative dataset. First, in the time period covered by our database, 412 Food Rescue used a mobile app push notification scheme which notifies volunteers within 5 miles when the rescue is first available and then notifies all volunteers 15 minutes later if the rescue has not been claimed. Thus, if a rescue is claimed within 15 minutes, we only treat the volunteers who were within 5 miles and did not opt out of push notifications as negative examples.

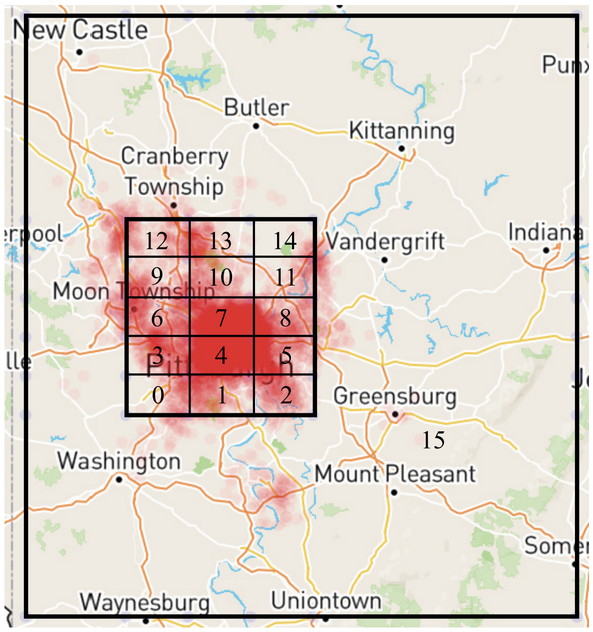
We also incorporate another data source to strengthen our negative sampling. In addition to mobile app notifications, the dispatcher at 412FR also manually call some regular volunteers to ask for help with a specific rescue. This usually happens when some rescue has been available for over an hour yet nobody has claimed it. We obtained the call history from 412FR, from which we identify the volunteers they reached out to within the time frame of each rescue. If these volunteers did not claim the rescues in the end, we treat them as negative examples. Compared to the negative examples derived from push notifications, we have more confidence in this set of negative examples, since declining on a phone call is a stronger indicator than ignoring a push notification.

5.4.3 Feature Engineering

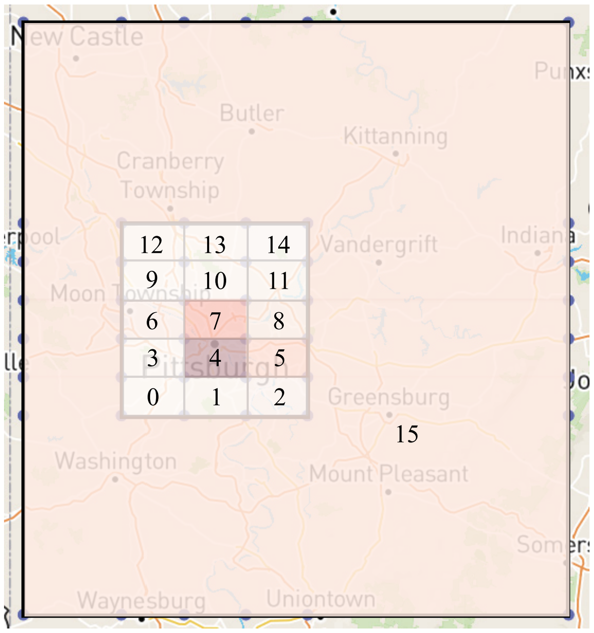
Based on our extended collaboration with 412FR, we carefully identify a selected set of useful features that are relevant in the food rescue operation.

First, the experience of food rescue dispatcher indicates that if a volunteer has completed a rescue at or near a donor or recipient, they are more likely to do a rescue trip again in the neighborhood. As shown in Figure 5.2, we divide the Greater Pittsburgh Region into 16 cells. We evenly divide a central rectangular region into a 3×5 grid, and label them grid cells 0 through 14. Then, we label the entire map outside the rectangular region cell 15. The rationale is that in the outer suburbs there are fewer donors, recipients, and volunteers, and furthermore volunteers who in suburbs are more willing to do long-distance, i.e. inter-cell, rescue than volunteers in downtown. For each rescue trip and each volunteer, we calculate the number of rescues the volunteer has done in the rescue donor's cell, in the rescue recipient's cell, and across all cells. These counts are only up to the date of the given rescue, so that we could prevent data leakage. We also tried to include as features the volunteer's historical rescues in each cell, not just the donor's and recipient's cell. However, they did not contribute any predictive power and thus we leave them out of the final model.

Closely related to this is the distance between the volunteer and the donor. It is unlikely that a volunteer would drive 30 miles to pick up a donation, as we show in Figure 5.3a. We measure the distance using the straight line distance based on geographic coordinates. Although the actual traveling distance might be a better indicator, we observe that the straight line distance already serves our purpose.



(a) Distribution of donor organizations. Darker colors mean more frequent donations. We plot the donor locations with random perturbations.



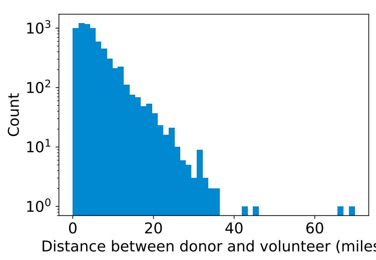
(b) Density of recipient organizations. Darker colors mean more recipient organizations in the grid.

Figure 5.2: We divide the Pittsburgh area into 16 grid cells, with cells 0–14 covering downtown Pittsburgh and its neighborhoods, and cell 15 containing the rest of the region.

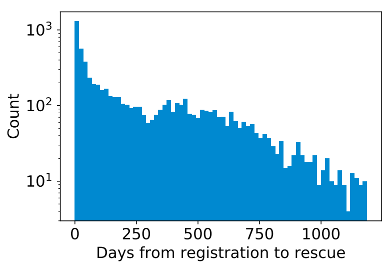
Aside from the geographical information, the length of time between volunteer's registration on the platform and the rescue is also an important factor, as suggested by our collaborators at 412FR. We plot the histogram of this variable in Figure 5.3b. Immediately after registration, the volunteer is eager to claim a rescue to get a feel of the food rescue experience. If a volunteer has stuck with the program for an extended period and remains active, it is likely that they are a regular and dependable one as well, which is substantiated with the upward trend and plateau in Figure 5.3b around days 300–600. Thus, we include this feature in our prediction model.

Weather information is also an important factor in the prediction. Presumably rainy and snowy days would see a lower volunteer activity in general. However, the impact of inclement weather would fall disproportionately on volunteers who do not have a car or live in suburban areas. We use the Climate Data Online (CDO) service provided by the National Oceanic and Atmospheric Administration to access the weather information.[3] The CDO dataset contains weather information at the discretization level of days and weather station. There are multiple weather stations in the Pittsburgh area and for each rescue we select the data for the date of rescue and the station that is closest to the donor organization. As shown in Figure 5.3c, on wet days, relatively more volunteers who claim the rescue reside in downtown Pittsburgh (cell 4 and 7). Whereas on dry days, a lot more volunteers who live in the outer suburbs of Pittsburgh (cell 15) are active. In fact, we also saw a significant difference in the average distance between

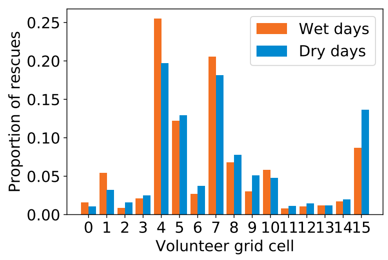
[3]https://www.ncdc.noaa.gov/cdo-web/



(a) Histogram of rescues, based on the distance between the donor and the volunteer who claimed the rescue.



(b) Histogram of rescues, based on the length of time between the rescue and the registration of the volunteer who claimed the rescue.



(c) Histograms of rescues under wet and dry weather, based on the location of the volunteer who claimed the rescue.

Figure 5.3: Rescues are often claimed by volunteers who are nearby. New volunteers are often the most active. Rainy days see more rescues in downtown than dry days.

Layer	Operation	Hidden Units
1	Dense (ReLU)	384
2	Dense (ReLU)	2048
3	Dense (ReLU)	512
4	Dense (Logistic)	16

Table 5.1: Neural network architecture

volunteer and donor for dry days (5.94 miles) and rainy days (5.22 miles), with a t-test p-value 3×10^{-8}.

We also explored a number of other features but did not incorporate them into our final model. These features include the rescue's time of day and day of week, the volunteer's availability, whether the volunteer uploaded an avatar to their profile or not, whether the volunteer is located in the same grid as the donor or recipient, and so on. Although these are intuitive factors, we did not find them improve the predictive power of our model and hence left them out.

5.5 Recommender System

We build a neural network-based recommender system. We first detail our network architecture, and then discuss our approaches to address the unique challenges in the food rescue domain.

We show the neural network architecture in Table 5.1. The input to the neural network is the feature vector of a rescue-volunteer pair. The feature vector passes through four dense layers. Each layer is followed by a ReLU activation function, except for the last layer where we output a single number which is then converted to a number between 0 and 1 by the logistic function. This output represents the likelihood that this volunteer will claim this rescue trip. We use the cross entropy loss to train the neural network. To output a list of k volunteers to whom we send push notifications for a particular rescue at prediction time, we pass the feature vectors of the rescue-volunteer pairs for all volunteers on a fixed rescue through the network and rank the output to take the top k of them.

5.5.1 Negative Sampling

As mentioned earlier, there is an extremely high label imbalance in our dataset. From March 2018 to March 2020, there are 6757 rescues available for the training. Each rescue typically has only one volunteer who claimed it, and there are 9212 registered active volunteers in the Pittsburgh area. This means, theoretically, the ratio between negative and positive examples is over $9000 : 1$. Using the method introduced in Section 5.4.2, we can obtain a selected set of negative examples D_n derived from push notifications and another set of negative examples D_c derived from dispatcher calls. The set D_c is slightly smaller than the positive examples D_p, while $|D_n| : |D_p| \approx 700 : 1$. When training the neural network, we always use all the examples from D_p and D_c. However, we randomly sample a subset of examples from D_n at each episode of the training. By doing this, we ensure that the negative examples from D_n do not dominate the training set, and at the same time the "more certain" negative examples from D_c gets emphasized more than D_n. This whole procedure leads to an overall ratio between negative and positive samples around $3 : 1$ in each single batch.

5.5.2 Diversity and Online Planning

Recommender systems in general suffer from the diversity issue, where "hot" items get recommended to all the users. In commercial applications, this might lead to the "rich gets richer" phenomenon on superstar items and the missed revenue opportunity on the less popular items. All these are valid. However, as we have emphasized several times in this chapter, the "items" on the other side of our recommender system are humans. The aforementioned consequences of the lack of diversity is only going to be more problematic in our case. If a popular volunteer received push notifications for every single rescue throughout the day, they would possibly get annoyed and mute the notifications. On the other hand, for volunteers who are already not very active, if our system never sent them push notifications, they would probably just forget about the platform and would be unlikely to return. Therefore, it is crucial that we properly handle the diversity issue.

We distinguish between two notions of diversity: individual diversity and aggregate diversity. The former means that each user (rescue) gets recommended a diverse set of items (volunteers). The latter means that the recommended items (volunteers) across different users (rescues) combined cover a large portion of the item space. Our human-centric approach determines that we focus on aggregate diversity here. In fact, we focus on a slightly different metric: how many times each volunteer gets recommended for a rescue every day. We wish to put a cap on this metric, which is directly linked to the user experience of each volunteer.

To this end, we can formulate the following mathematical program for a given day of food rescue operation.

$$
\begin{aligned}
(\Pi) \quad \max_{x} \quad & \sum_{i\in R}\sum_{j\in V} p_{ij}x_{ij} \\
s.t. \quad & \sum_{j\in V} x_{ij} \le k, \quad \forall i \in R \\
& \sum_{i\in R} x_{ij} \le b, \quad \forall j \in V \\
& x_{ij} \in \{0,1\}, \quad \forall i \in R, \forall j \in V
\end{aligned}
$$

Let V denote the set of volunteers. On a particular day, we have a set of rescues R. The binary decision variable x_{ij} is equal to one if we decide to send push notification to volunteer j about rescue i. The first constraint indicates that for each rescue we will notify the top k volunteers, as introduced at the beginning of Section 5.5. The second constraint is our diversity constraint, which makes sure that each volunteer receives at most b push notifications a day. The p_{ij} in the objective is the output from our trained neural network, representing the predicted likelihood that volunteer j is going to claim rescue i.

While this optimization problem Π is a valid method to improve diversity in generic recommender systems, it does not solve the problem in our setting. The reason is that donations, and hence food rescue trips, arrive in our system sequentially throughout the day, and the dispatcher must also act in real-time. It is unacceptable to wait till the end of the day, run the optimization problem above, and then send the push notifications. Therefore, we need an online algorithm.

An intuitive approach is to resort to the literature on online linear programming [4]. Indeed, we could imaging solving π where at each time step, a new rescue is revealed with a new column in the x matrix and p matrix. However, we do not know how many rescues there will be at the beginning of the day. This is a major obstacle in applying the established algorithms with theoretical guarantees. Instead, the daily rescue pattern is hardly adversarial in nature and thus we propose a simple heuristic, as shown in Algorithm 6.

In Algorithm 6, when a food rescue arrives in the system, we sample the historical rescue data for trajectories. Typically, we would sample the rescues on the same weekday a week ago, two weeks ago, and so on. The underlying idea is that the same weekdays might have similar rescue patterns. This is because most donations that come to 412FR come from grocery stores or large companies/universities. Grocery stores often perform inventory counts on a weekly basis. Companies and universities often hold weekly events, with catered food. For each sampled day, we only take the trajectory from the time of the current rescue to the end of the day. Then, for each trajectory along with the current rescue, we obtain the neural network's predicted claim probabilities and solve the optimization problem Π_i. Π_i is similar to Π except that each volunteer now has their own remaining budget of push notification. Note that now everything in Π_i is observed and known, whereas in Π the future rescues are unknown at the decision-making time. We keep only the part of the optimal solution that concerns the current rescue and discard the rest. Later, on Line 8 in Algorithm 6, for each volunteer, we sum over its value in the optimal solution across all the sampled trajectories. We take the top k volunteers as voted by these solutions, who become the ones we will send push notifications to for this current rescue.

```
Algorithm 6: Online Planning for Optimizing Push Notifications
  input: A trained neural network predictor
1 while a new rescue $i$ arrives do
2     Flush $X_i$
3     for dayToSample $= 1, 2, \dots H$ do
4         Sample the set of rescues $R$ on the dayToSample that occured from the time of
            the current rescue $i$ till the end of the day.
5         Compute predicted claim probabilities $p_{ij}$ and $p_{i'j}$ for all $i \in R$, for all $j \in V$.
6         Solve the following optimization problem:
              $$(\Pi_i) \quad \max_x \sum_{j \in V} \left( p_{ij} x_{ij} + \sum_{i' \in R} p_{i'j} x_{i'j} \right)$$
              $$s.t. \quad \sum_{j \in V} x_{i'j} \leq k, \quad \forall i' \in R$$
              $$\sum_{j \in V} x_{ij} \leq k$$
              $$x_{ij} + \sum_{i' \in R} x_{i'j} \leq b_j, \quad \forall j \in V$$
              $$x_{ij} \in \{0, 1\}, \quad \forall i \in R, \forall j \in V$$
7         Keep in $X_i$ the optimal solution $x^*_{ij}$ for the current rescue only.
8     Sum $X_i$ over the sampled histories, find the top $k$ volunteers.
9     Send push notifications to them. Update the remaining budget $b_j$ for each
        volunteer $j$.
```

We note that the optimization problem Π and Algorithm 6 are extremely flexible to account for many additional considerations. For example, we could use personal budget b_j in Π and add additional constraints to represent the volunteer's push notification preferences. We could also add weights to the objective function to emphasize the importance of a particular rescue.

5.6 Experiments

We introduce the experiment results of our algorithms in this section. There are two desired goals in using such a recommender system: we hope more rescues will be claimed by volunteers, and we hope volunteers will stay with the platform. Towards these two goals, we aim to evaluate the efficiency and diversity of the models using historical data. The main efficiency metric is the hit ratio, that is, for what proportion of all rescues does the model's recommended short list of volunteers contains the volunteer who actually claimed the rescue in the historical data. To evaluate diversity, we look at the histogram of the number of times each volunteer is selected in the short list.

5.6.1 Recommender System

We use a training set containing rescues from March 2018 to October 2019, which is 80% of the entire dataset. We use the remaining 1373 rescues from November 2019 to March 2020 as the test set. We conducted all of our experiments on an Intel i7-7700K 4.20GHz CPU with 64GB RAM.

First, we only consider the prediction part of our algorithm. We compare our neural network recommender system with several competitive baselines that are commonly used, including random forest (RF), gradient boosted decision trees (GBDT), and stacking model (SM). To determine the hyper-parameters of the baseline models and the neural network model, we separate a validation set which consists of the last 1/8 of our training set and then run a grid search according to the performance on the validation set. For experiments on the baselines, we use the same negative sampling method on D_c and D_p as described in Section 5.5.1. As for negative examples from the app notifications D_n, since the baselines are not gradient descent-based methods, we sample them in two schemes such that the ratio between the positive and negative examples is roughly $1 : 1$ and $1 : 20$, respectively. We consider the latter because that is roughly the number of negative examples that the neural network approach has seen throughout the training, in order to ensure a fair comparison.

We show the results of all these algorithms on the test set, averaged over 5 runs, in Table 5.2. We consider the hit ratio at k (HR@k) and the normalized discounted cumulative gain at k (NDCG@k) in Table 5.2. However, we note that our main metric of interest is the hit ratio, because when sending push notifications, we do not care about the particular order in which each volunteer ranks on the list. Also because of this, HR@k is our primary metric during the grid search on hyper-parameters for all the predictive models. We choose the value of k to be 964, since this is the average number of push notifications sent per rescue under the current notification scheme. The current distance-based notification scheme has a hit ratio of 0.4392. All baselines show better performance than the current method, with random forest and GBDT being better than the stacking model. However, the neural network based prediction model outperforms all the baselines.

The hit ratio of the neural network model is a 66% improvement over that of the current distance-based method. This means that we would be able to reach the would-be volunteer in approximately 900 more rescues every year. Each of these rescues has a donor and a recipient organization that serves tens or hundreds of people behind it. A smooth food rescue experience would not only provide basic food necessities to these people, but also encourage these organizations to keep up the engagement in a sustainable way.

5.6.2 Diversity and Online Planning

As mentioned in Section 5.5.2, recommender systems, in general, suffer from the diversity issue. This problem also exists in our model. In Figure 5.4, we plot the histogram of the number of push notifications received by each volunteer for the test set rescues. The notification budget in the online planning algorithm is enforced by leveraging the date and time associated with each rescue in the test set. This is doable because the train-test split is done in a temporal fashion, as mentioned at the beginning of Section 5.6.1.

Model	HR@k (SD)	NDCG@k (SD)
NN	**0.7269 (0.0310)**	**0.1898 (0.0147)**
RF(1:1)	0.5989 (0.0395)	0.1319 (0.0303)
RF(1:20)	0.6035 (0.0511)	0.127 (0.0053)
GBDT(1:1)	0.6235 (0.0549)	0.1613 (0.0098)
GBDT(1:20)	0.5394 (0.0152)	0.1023 (0.0086)
SM(1:1)	0.4996 (0.0005)	0.1332 (0.0002)
SM(1:20)	0.5219 (0.0125)	0.0948 (0.0030)
Default	0.4392 (N/A)	N/A (N/A)

Table 5.2: Neural network based recommender system achieves better hit ratio and NDCG than several baselines. All experiments are repeated five times with the mean and standard deviation shown in the table.

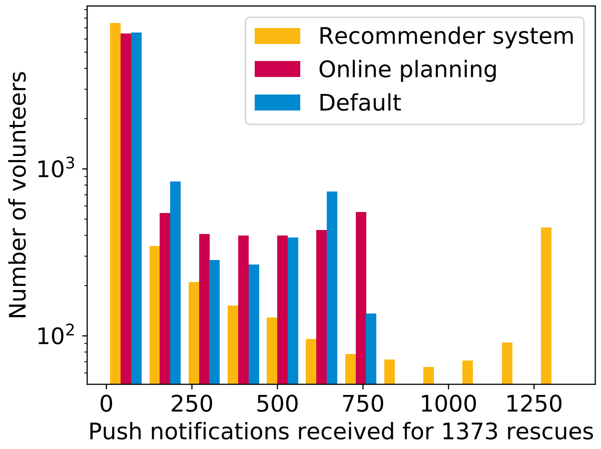


Figure 5.4: Histograms of the number of push notifications received by each volunteer over all the 1373 rescues in the test set. The online planning algorithm has a budget of 6 notifications per day.

The neural network based recommender system, shown in yellow in Figure 5.4, exhibits an alarming bimodal distribution: most volunteers either receive almost no push notifications, or receive push notifications for almost every single rescue. We remark that although the number of volunteers in the rightmost bin (446 out of 9312) is much smaller than that in the leftmost bin (7458 out of 9312), the former is much more concerning. This is because they are typically the most "active" volunteers who have contributed the most to the food rescue program. In fact, these 446 volunteers contain 39 of the top 50 most frequent volunteers, and 51 of the top 100. If they left the platform due to too many notifications, which is likely to happen should the proposed recommender system get deployed, the loss to 412FR would be disproportionately high. On the other hand, the default distance-based notification scheme does not suffer from this issue, as shown in red in Figure 5.4. Although the majority of the volunteers still receive few push notifications, the notification frequency for each volunteer is capped at roughly once every two rescues.

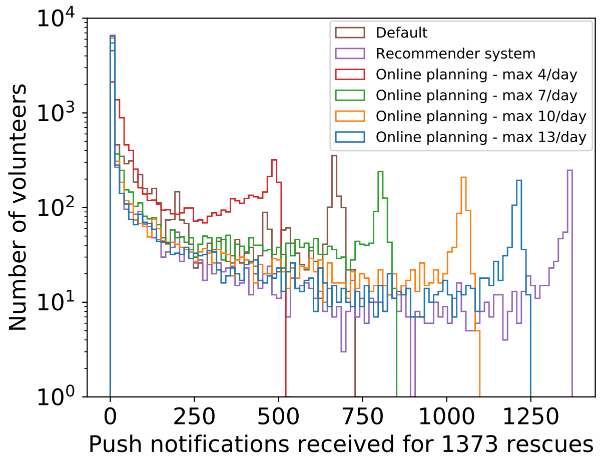


Figure 5.5: Histograms of the number of push notifications received by each volunteer over all the 1373 rescues in the test set, compared across different budget values. Higher budget leads to some approximately 500 volunteers receiving more notifications.

This phenomenon should not be too surprising, because we used the volunteers' past activities as input features to the ML model, and it turns out that these features are very important. In fact, if we select volunteers by directly ranking their past number of rescues in the neighborhood, that would give us a hit ratio around 60%, and of course, that approach would make this diversity issue even worse.

The bottom line is, figure 5.4 serves as a stern warning against the premature deployment of machine learning algorithms in the real world. That a certain model outperforms the current practice by 66% in some important metric (here, the hit ratio) does not mean it would not cause other problems.

We use our Algorithm 6 to improve the diversity of volunteer recommendations. As a preliminary and straightforward comparison, we ran our online planning Algorithm 6 with budget $b = 6$ push notifications per day using the rescues seven days ago as the sampled history. We plot its notification histogram in yellow in Figure 5.4. It is easy to see that the online planning algorithm achieves a push notification distribution much more similar to the default scheme, than the recommender system alone. It completely avoids sending push notifications about every single rescue to any particular volunteer.

Indeed, the effect of Algorithm 6 on recommendation diversity depends on the budget parameter b. In Figure 5.5, we plot the notification distributions for different choices of the budget value, and compare them against those of the recommender system and the default notification scheme. As the budget increases, the distribution of push notifications from Algorithm 6 approaches that of the recommender system. We note that the position of the rightmost peak of each histogram should not be interpreted as an indicator of the total number of push notifications sent. In all of these experiments, we limit the number of notifications for each rescue at $k = 964$. Except for when the budget is extremely small, the algorithm always notifies exactly 964 volunteers for each rescue. The diversity goal here is to make the histogram occupy as little space as possible on the right side of the figure.

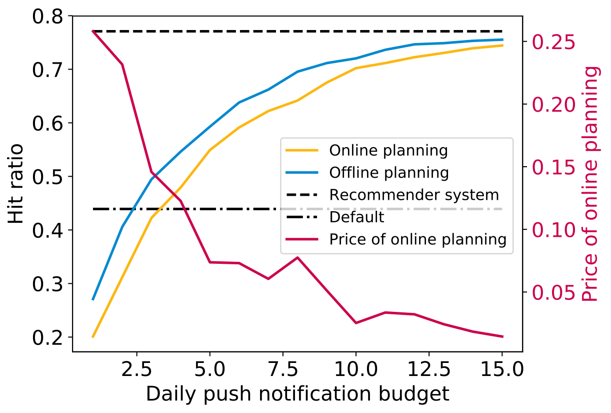


Figure 5.6: Hit ratio of the online planing algorithm. Price of online planning, computed as $1 - \frac{HR_{\text{online}}}{HR_{\text{offline}}}$, is shown on the right axis.

Much as we demonstrate the improvement of recommendation diversity, we would also like to ensure that the recommendation accuracy of our algorithm does not drop too much. The budget parameter b captures the inherent trade-off between diversity and accuracy. As we show in Figure 5.6, the yellow curve represents the hit ratio of Algorithm 6. Algorithm 6 outperforms the existing notification scheme when the budget is more than four notifications per day, which is a relatively trivial amount. When the budget rises to 10 notifications per day or more, the hit ratio is very close to the bare bone recommender system.

In order to further evaluate the quality of *online* planning in Algorithm 6, we also solve an offline version of the problem, where we solve the mathematical program Π separately for each day, assuming full information about the rescues on that day. We show the hit ratio of the recommendation decision from this offline version in blue in Figure 5.6. Since having full information is always better, the blue curve always lies above the yellow curve representing the online planning. However, the difference is not big. We term the difference as the "price of online planning", which is computed as $1 - \frac{HR_{\text{online}}}{HR_{\text{offline}}}$. In fact, Figure 5.6 shows that the price of online planning is decreasing as the budget grows, and is consistently smaller than 0.1 when the algorithm is of potential deployment interest (performing better than the current practice). This validates our earlier claim in Section 5.5.2 that the rescues on the same weekday of the previous week are a reasonably good indicator of the rescues on the present day.

5.7 Conclusion and Future Directions

A critical goal in the food rescue operation is to be able to reach the "right" volunteers in time. Working with 412 Food Rescue, we developed a machine learning model to recommend the most probable volunteers to send push notifications to for each given rescue. Our machine learning model improved the hit ratio of the current notification scheme by 66%. The food rescue operation features two main challenges: the recommendation diversity is of utmost importance to ensure volunteer experience, and the recommendation must be made in an online fashion. We proposed a novel algorithm to dynamically recommend volunteers for rescues in real time, while diversifying the recommendations and still managing to keep the hit ratio well above the current practice.

The problem of low hit ratio is a real problem that needs to be addressed. This is also a problem natural for a data-driven approach, and we do have the relevant data available. There is an existing approach to this problem (distance-based notification), so our technological intervention does not introduce new initiatives. Instead, we amplify the existing initiative. Lots of previous endeavors have shown that this amplification approach is more likely to achieve deployment and sustainable impact [153]. In fact, our technological intervention does not replace, reduce, or attempt to dictate any human employee's job at the FR.

There are two immediate future directions that are useful in the food rescue operation. First, our Algorithm 6 features a classical predict-then-optimize framework where the learning objective and the optimization objective are not perfectly aligned. It would be interesting to consider the recent literature on data-driven optimization [19, 44] to further improve the results shown in Figure 5.6. Of course, the online nature of our problem brings an additional interesting challenge that has never been addressed in that literature. Second, recommendation is necessarily limited by the counterfactuals. In Section 5.4.2, we proposed several approaches to select the most credible negative examples. It would be interesting to identify credible positive examples beyond the explicitly labeled ones, which are rather scarce in the dataset.

A pilot study of the model and algorithm described in this paper is detailed in Section 6.2 of the following chapter.

Ethical implications We addressed a main unintended consequences of our recommender system as a key contribution of this work, that is, the over-concentration in recommendations. In practice, there are more ways to ensure that volunteers are not harassed with too many notifications. The food rescue mobile app allows each volunteer to specify the periods where they do not want to receive any notifications ((morning, afternoon, evening) x (weekday, weekend)). On the other hand, the recommender system might be ignoring certain volunteers more than the status quo. While this is a less serious concern than the one we focused on in the work, there are also ways to address it. For example, we can ensure that new volunteers always get all notifications for all the rescues in their vicinity. The use of human subject data in this chapter has been approved by the IRB.

Chapter 6

The Field Deployment of Food Rescue Algorithms

As we mentioned at the beginning of this thesis, AI4SG research that only stays on paper does not fulfill its purpose. In the previous two chapters, we evaluated our algorithms on historical data. However, historical data could only tell us very limited information. In this chapter, we took both our algorithms to the field. The first deployment of the work in Chapter 4 is a purely observational study. The second deployment of the work in Chapter 5 is a randomized controlled trial. Only by taking the algorithms to the field can we verify the implicit assumptions we made when evaluating the algorithms on historical data.

6.1 Deployment of the Generic Notification Scheme

The intervention and notification scheme in Section 4.5.2 has been adopted by 412FR since February 2020. Table 6.1 shows the key food rescue metrics before and after the adoption. Because COVID-19 started to impact the life in the area in March 2020, we split the period after adoption into before COVID and after COVID in order to provide a more objective picture of the results. As shown in Table 6.1, after the adoption of our recommended INS, the rescue claim rate went up, the rescues got claimed faster, and 412FR had to send fewer push notifications. This indicates that all these three key metrics improved in the desirable direction. After COVID hit the area, the improvement stuck around, and got even better in both the claim rate and the claim speed. Nevertheless, we need to emphasize that this is not a controlled experiment. There could be confounding factors.

Condition	Claim Rate	Average time from publish to claim (min)	Average # push notifications sent
Before 2/10/2020 (Previous scheme)	0.84	78.43	11499.45
2/10/2020 - 3/1/2020 (New scheme)	0.88	43.05	9167.52
After 3/1/2020 (After COVID)	0.92	39.73	9735.54

Table 6.1: Food rescue metrics before and after the adoption of the recommended INS.

6.2 Rescue-Specific Notification Scheme

The initial results from the deployment of the generic notification scheme encouraged us to also test the recommender system notification scheme in the real world. It is also the limitations with the previous observational study motivated us to run a randomized controlled trial (RCT).

6.2.1 Setting up the RCT

In contrast to the previous deployment, this deployment was much more involved, where we had to integrate the ML model into 412FR's food rescue management platform "FoodRescueHero" (true name hidden for double-blind review). The platform is a web application built in Ruby on Rails. When a rescue is published, the application will start a notification job on Sidekiq. Prior to this work, the notification job would select a subset of available volunteers based on distance (Chapter 4). The selected volunteer IDs would be sent to the push notification service. With the recommender system we replace the distance-based selection with a python script that runs the ML model and returns the selected volunteer IDs. A redis server keeps track of each volunteer's remaining push notification budget. We update the remaining budget after each rescue and reset the budget at the beginning of the day.

To make the pilot study go through, we had to make one compromise. After we implemented the ML system within the FoodRescueHero platform, we realized that it would take the optimization part (Section 5.5.2) 40 seconds to run on 412FR's server. Since they only had a single server, this would block all other jobs. Therefore, we jointly decided to drop the optimiation part from the RCT. While obviously we might be sending some users more push notifications, it is also possible that those people who tend to be more frequent volunteers actually want more notifications. The only way to find that out is through a field test. Meanwhile, volunteers can always mute the notifications. Also, since this is just a short-term pilot study, negative consequenses, if any, could be controlled to a minimum.

With this in mind, we designed our pilot study as follows. When each rescue is published, it is uniformly randomly assigned to one of the three treatment groups: control, ML, and ML-Random. The control group is the distance-based notification scheme that is currently in use. The ML group is to run the ML model and greedily send notificaions to the top k volunteers. The ML-Random group is to select a uniform random 80% sample from the top k ML predictions, and $20\%k$ volunteers uniformly randomly. We include the ML-Random group as a way to explore volunteers who would claim a rescue but would be ignored by the ML or distance-based model. This is also, in spirit, compensating for the drop of the optimization part of the algorithm. In the RCT, whichever group a rescue is assigned to, we use the distance-based criteria to get the number of users to select (k) in the case of either test groups.

6.2.2 RCT Results

The pilot started on May 23, 2022 and ended on June 21, 2022. The results are shown in Table 6.2. We measured four main food rescue metrics: hit rate, claim rate, claim time, and notifications sent. Hit rate measures whether the push notifications cover the volunteer who claimed the rescue. This is essentially the metric that the ML model is optimized for. As expected, we

Group	Rescues	Hit rate	Claim rate	Claim time (min)	Notifications sent
Control	212	0.468	0.807	74.708	5752.240
ML	169	0.651 (0.001)	0.882 (0.047)	84.396 (0.537)	5647.107 (0.758)
ML-Random	211	0.489 (0.696)	0.844 (0.317)	70.556 (0.741)	6033.815 (0.373)

Table 6.2: Results from the pilot study. Numbers in the parentheses represent the two-tailed p-value from two-independent-sample proportion z-test (hit rate and claim rate) or two-independent-sample t-test (claim time, notifications sent), with respect to the control group.

see that the ML group significantly improved the hit rate over the control from 46% to 65%. Claim rate refers to what proportion of rescues get claimed (by anyone). On this metric, the ML notification system also significantly improved over the control from 80% to 88%. This is, in a way, the most important result of this pilot study. The claim rate is not directly part of the training signal for the ML model, yet the ML model was able to significantly improve this arguably the most important indicator of rescue notification efficacy. While it is hard to find out the exact reason for certain, we suspect it could be due to volunteers' decision to claim a rescue could also depend on the timing of receiving notifications. Getting notified for a "fresh" rescue could make some likely volunteers cross their threshold to actually claim it.

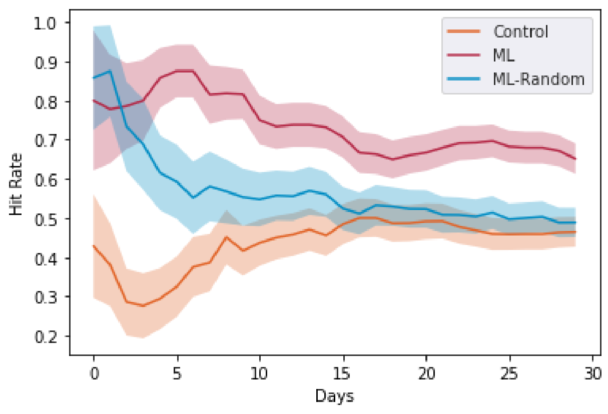
The two positive results above are essentially all that we could hope for based on the work in Chapter 5, and possibly a little bit more. Meanwhile, the other results also yielded interesting insights.

ML-Random group

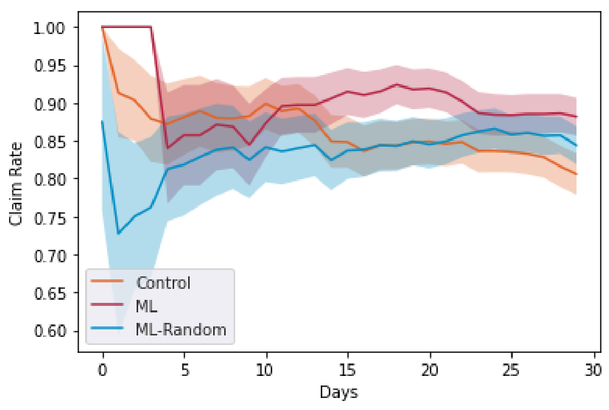
First, the ML-Random group showed better hit rate and claim rate than control, but the difference is not statistically significant. The hope was that randomness could discover some volunteers who would not be notified otherwise and nudge them to participate. In practice, ML contributed 83 hits while the random sampling contributed only 4 hits. It still remains as an open question how to encourage more volunteers to claim the rescue trips.

Temporal analysis

Second, as shown in Figure 6.1a, the ML model's hit rate appears to be decreasing over time. The ML group had a hit rate of over 80% in the first week but gradually reduced to around 70% at the end. The downward trend also shows up in the ML-Random group. It seems that the steep decline in the first week could suggest more of the variance in the small number of data points than the actual trend. That said, the decline in hit rate continued, albeit at a slower pace, for the rest of the trial. This drop in hit rate could be explained by not updating the model throughout the RCT, since the hit rate is precisely the signal that the ML model is trained to maximize. This suggests that in permanent deployment, we need to have mechanism to automatically retrain the model frequently to maintain the highest level of performance. Meanwhile, the claim rate shows an almost opposite trend over time in Figure 6.1b. Both ML and ML-Random, after the initial week of smaller number of data points, see their claim rate slightly improving over the next 3 weeks. This might be because over time, the recommender notification scheme has led



(a) Hit rate



(b) Claim rate

Figure 6.1: The hit rate and claim rate for all treatment groups as time progressed. Each day's data represent the hit/claim rate for all rescues up to that day.

to an increase in the intrinsic engagement level of some frequently notified volunteers. On the other hand, the claim rate for control appears to decay over time.

Spatial analysis

Third, let us look at the spatial distribution of these key metrics. Recall that in the feature engineering for the ML model, we divided the region into 16 grid cells (Figure 5.2). For each treatment group, we further divide the rescues into 16 subgroups depending on in which grid cell their donor is located. As shown in Table 6.3, for the hit rate and claim rate that we have observed statistically significant results as a whole, we further observed that in grid cell 5, 7, and 15, the ML group and/or the ML-Random group significantly outperforms the control. However, in grid cell 4, the results are too close to call. We chose these 4 regions because these are the 4 regions with the most rescue activities. Cell 4 captures most of the downtown urban area, while cells 5 and 7 are near suburbs and cell 15 is all the outer suburbs combined. The results here suggest that the ML tool is particularly useful outside downtown where volunteer resources are not as abundant. For region 15 in particular, the control has a disastrous 12.5% hit rate and 45.7% claim rate, while the ML model improved them to 40.9% and 68.8%, respectively. The ML-Random group also showed improvement in both metrics, albeit at a smaller scale. This spatial difference is quite expected. In urban areas where volunteers are abundant and transportation is easier, there could be less need to engage volunteers through push notifications, because enough people would just open their app on their own and look for outstanding rescues. However, in suburban areas, there are fewer people living around any given donor location, hence the band performance of control and the need for the ML tool.

For claim time and number of notifications sent, we did not observe significant differences between treatment groups overall. We did not expect any significant differences, because of the huge variance in the historical data coupled with the limited sample size in the RCT. That said, when we zoomed into each geographical region, there were some significant differences in

Group	Rescues	Hit rate	Claim rate	Donor Region
Control	68	0.541	0.897	4
ML	37	0.688 (0.172)	0.865 (0.620)	4
ML-Random	63	0.593 (0.577)	0.857 (0.486)	4
Control	12	0.4	0.417	5
ML	15	0.357 (0.865)	0.933 (0.003)	5
ML-Random	10	0.444 (0.872)	0.9 (0.019)	5
Control	38	0.568	0.974	7
ML	48	0.804 (0.019)	0.958 (0.700)	7
ML-Random	56	0.491 (0.472)	0.946 (0.521)	7
Control	35	0.125	0.457	15
ML	32	0.409 (0.057)	0.688 (0.057)	15
ML-Random	40	0.174 (0.677)	0.575 (0.308)	15

Table 6.3: Results from the pilot study for specific regions. Numbers in the parentheses represent the two-tailed p-value from two-independent-sample proportion z-test, with respect to the control group.

certain regions as shown in Table 6.4. Overall, no significant differences are observed in densely populated regions such as cells 4, 5, and 7. In regions 8 and 15, ML-Random reduced the claim time; in regions 8 and 10, ML-Random and ML reduced the notifications sent, respectively. In region 1, however, both ML and ML-Random increased the notifications sent over control. Despite some successes and rare failure in the regions above, these two metrics have quite big variances in general, and the ML push notification might not be the most direct way of impacting them.

What did we lose by dropping the optimization component?

Lastly, before the start of this RCT, we made the compromise to drop the optimization component from the trial. We would like to see if removing that part would lead to any negative consequences that we had expected earlier. We compared each volunteer's notification preferences before and after the trial and show the results in Table 6.5. For a comparison benchmark, we used the data from the three months immediately before the RCT. For each period, we consider two groups of volunteers. In the app, each volunteer could select whether they wish to receive notifications during each of the 6 time slots ((weekday, weekend) × (morning, afternoon, evening)). We look at volunteers who at the end of the time period, have at least unselected one of their previous elections, and volunteers who at the end of the time period, have at least selected one of their previous non-elections. During the RCT period, 30 volunteers tuned down their notification elections among them one was among the 50 most frequent volunteers in the past year. Meanwhile, 14 volunteers tuned up their notifications, and similarly one was among the 50 most frequent volunteers. The pre-RCT period is 3 times as long as the RCT period. But since the total number of volunteers and the seasonality and other circumstances are different, we cannot directly compare these two groups. Rather, we assume that in each period, the ratio between the metrics for volunteers who tuned up the notifications and the metrics for

Group	Rescues	Claim time (min)	Notifications sent	Donor region
Control	8	20.625	1729.5	1
ML	8	44.143 (0.471)	4928 (0.074)	1
ML-Random	3	47.333 (0.424)	6653.667 (0.032)	1
Control	9	130.833	7762.222	8
ML	7	154 (0.674)	6343.571 (0.123)	8
ML+Random	15	58.077 (0.038)	5887.2 (0.047)	8
Control	10	14.333	3733.5	10
ML	7	6 (0.269)	753.714 (0.035)	10
ML+Random	5	27 (0.358)	4283.2 (0.768)	10
Control	35	145	7447.4	15
ML	32	175.636 (0.618)	7115.531 (0.487)	15
ML+Random	40	65.826 (0.010)	6664.85 (0.148)	15

Table 6.4: Results from the pilot study for specific regions. Numbers in the parentheses represent the two-tailed p-value from two-independent-sample t-test, with respect to the control group.

Time period	Volunteers who...	Total	Top 100	Average rescues in past year	Average ranking
5/23/22 - 6/21/22	reduced notifications	30	2	11.13	555.3
(RCT)	increased notifications	14	1	15.86	790.6
2/25/22 - 5/23/22	reduced notifications	50	1	7.84	734.8
(Pre-RCT)	increased notifications	29	1	6.52	646.5

Table 6.5: Changes in user notification preferences before and after the RCT, compared to a previous period.

volunteers who tuned down the notifications should be similar, if the ML model had no effects. Then, we observed that during the RCT period, there has been relatively more volunteers who tuned down their notifications, and more most frequent volunteers tuned down. In addition, those who tuned down tend to rank higher in their past participation frequency, although the number of rescue that they have done is smaller.

There are two caveats. First, there was not an RCT on this matter and all results in Table 6.5 are observational. Hence there could be potential confounders in the results. Second, sometimes a user might enable or disable notifications not through the app but through their phone's settings, which we have no way to track. Based on the data that we do have, we believe there is a weak signal that the removal of optimization component indeed led to more volunteer disabling push notifications. The negative impact is not huge and hence the one month's RCT likely caused limited harm. However, the negative impact is likely real. Hence, in the permanent deployment, we will bring back the optimization component to minimize this negative impact. Furthermore, encouraged by the positive results from this RCT, 412FR will migrate all of its data science plans to the ML platform of Amazon Web Service. This will enable more compute and hence eliminate the compute bottleneck that prevented the inclusion of the optimization component in the RCT.

6.2.3 A Note on Interference

We note that the RCT design above suffers from interference. Interference is when the treatment received by one unit might affect the outcomes of other units. This is a violation of the assumption commonly referred to as SUTVA: stable unit treatment value assumption. In our RCT, interference is present because although we randomize on the rescue level, the outcomes of all rescues are determined by the same group of (all the) volunteers. Specifically, interference could come in at least the following three forms:

1. Suppose rescue 1 arrives (regardless of which treatment group we assign it to), we send out notifications. Then, immediately after, another rescue 2 arrives, we send out notifications again. The volunteer who just claimed rescue 1 almost certainly will not claim rescue 2.
2. Suppose rescue 1 arrives, and gets assigned to the ML (or ML-Random/control) treatment group, we send out notifications. Then, a few hours later, another rescue 2 arrives, gets assigned to the ML (or ML-Random/control) treatment group again, we send out another round of notifications. Whether a volunteer got notification for rescue 1 might affect their reaction to the notification for rescue 2.
3. Suppose rescue 1 arrives, and gets assigned to the ML (or ML-Random/control) treatment group, we send out notifications. Then, a few hours later, another rescue 2 arrives, gets assigned to the ML-Random or control group, we send out another round of notifications. Whether a volunteer got notification for rescue 1 might affect their reaction to the notification for rescue 2.

The first form of interference is unavoidable. Luckily, this is a relatively rare case, and the large number of volunteers relative to rescues help alleviate its impact. We will not focus on this in the subsequent discussion.

The second and third forms of interference both concern how previous notifications might impact volunteer's reaction to current notification. We could get rid of these two sources of interferences by making an assumption. The assumption is that all volunteers have no memory, that is, they make decision immediately and purely based on the current notification. In this sense, volunteers are essentially part of the fixed environment under which we treat the rescues as the one and only subject of the study. *Our results presented earlier would be valid under this assumption.*

If one hopes to (at least partially) lift this assumption, one might propose the following design: at the beginning of the experiment, randomly partition all volunteers into three groups, each corresponding to a treatment group on the rescue trips. This might eliminate the third form of interference but the second one still remains. And more seriously, this approach is problematic for the following reason. This approach is similar to the graph cluster-based design method, which works well when the underlying user-item graph is sparse and is close to having perfect clusters. This is not the case here. We have a well-connected graph - any rescue could potentially be associated with any volunteer. If we partitioned the volunteers as proposed above, we would be cutting off many edges. In the end, we would be testing our algorithms in an environment different from the true environment. Here is a simple way to show that this approach would not work. For simplicity, let's take the ML-Random group out of the picture. Suppose there is only one special volunteer who has 50% chance claiming a rescue under the control notification scheme, and has 60% claiming a rescue under the ML notification scheme. All other volunteers have $\epsilon \approx 0$ chance claiming a rescue under either notification scheme. This is obviously an extreme case, but it is not too far from the reality: in reality, a very small group of volunteers claim the majority of the rescues, and most volunteers never claim anything. If we were to partition the volunteers into two groups, then with 1/2 chance we would conclude that control is better than ML no matter how long we keep the experiment running. In practice, this would greatly reduce the power of our experiment.

Of course, this is not to say we should give up. In fact, our problem is similar in flavor to testing a recommender system on a two-sided market. This is an active and open area of research. As explained above, clustering-based method [124, 154] is not applicable to our problem, at least not until we roll out experiments in multiple different cities. In the advertising literature, previous work consider splitting user's "budget" over copies of universe, each corresponding to a treatment group [17, 90]. This "budget" is in the same direction as our intent to consider the effect of volunteers' previous notifications' on current decision-making. However, this effect is far more complex in our setting than a straightforward "budget", and one might have to go down the undesirable path of explicit modeling. In addition, there has been some recent work on designing the rank-ordering of recommender systems to compute the "producer" side of the market [109]. However, it does not offer a direct characterization of the average treatment effect estimate (only the rank-ordering) and our problem does not concern rank-ordering.

6.3 Permanent Deployment

Encouraged by the results from the trial, we are working together to deploy this push notification recommender permanently. Rather than have the model as a static file living on the server, we migrate the model to a separate AWS service. In this way, we could automatically update the model on a regular basis. Not only is this separation more reliable and sustainable, but it creates the compute to run the optimization component of the algorithm as well. This would put a safety belt on the model's potential negative impact on volunteers' interest.

6.4 Lessons Learned

Collaboration between nonprofit organizations and academic researchers is challenging. In what follows, we attempt to gather some lessons we learned in this collaboration that are actionable for academic researchers.

When developing AI in the nonprofits context, it is almost cliché that the design of algorithms and models should fit within the often limited computational resources available at the nonprofits. This is easier said than done. Some choices are obvious; for example, we used a small neural network that does not require a huge memory or GPU access. However, it was much harder to foresee that it would be undesirable for the optimization component to take

40 seconds to run because of how the system architecture was built. That realization would only happen when one dove deep into the codebase of the entire system. That said, what we learned is that although this saying has some truth to it, it should not be taken only at face value. The computational infrastructure, although limited, is not fixed. When we demonstrated the impact of the model and how the optimization component could make it even better, not only through backtesting but also in real world trials, our collaborator could be convinced to invest in the computational infrastructure. The trade-off between performance and resource usage is not always inevitable. With time, effort, deliverables, and genuine collaboration, the trade-off could be lifted.

Academic researchers come into these collaborations with a certain toolkit. They have every incentive to drive the collaboration towards a direction that would best fit their research agenda. However, this risks them ending up not working on something the nonprofit organization wants the most. This is disrespectful to their nonprofit collaborators. And they would pay the price eventually – because the nonprofit does not need it, they probably would not want to deploy it. The simple and obvious way out, we believe, is real listening. At the beginning of our collaboration with 412FR, we had a long list of research questions that would leverage our technical expertise. However, the collaboration only took off when we really listened to their needs and threw our own research agenda off the table, even if that meant we had to explore a new research area.

After throwing our research agenda out of the room, we had to throw ourselves into the problem setting. The several meetings with our food rescue partners gave us an initial understanding of the problem setting, but to really understand the problem, we had to be part of that problem as well. We completed food rescue trips ourselves as volunteers, sat with the food rescue dispatchers in the office to observe how they work, and went on rides with the food rescue truck drivers to meet the communities they serve. Such personal experience helped a lot in our later problem formulation and research.

On problem formulation, it is nothing new, but probably worth reiterating, that technology amplifies existing initiatives rather than create new ones [153]. Push notifications and dispatcher interventions are something that FRs had already been doing prior to our work. Thus, our work simply suggests new parameters that FRs may reference in their standard procedure. This made our work more easily accepted and deployed.

Ethical implications We have discussed the ethical implications of the algorithms being deployed in previous chapters. Here, we will focus on ethical implications of the experiment itself on both the volunteers and donor/recipient organizations. It is true that removing the optimization component made our model more susceptible to over-concentrate on certain volunteers. However, the experiment is also a chance to test that hypothesis. The experiment is short enough to control the damage, if any at all, as we showed in the results. The experiment design has been approved by IRB.

Part III

Machine Learning for Conservation

Chapter 7

NewsPanda: Media Text Monitoring for Timely Conservation Actions

In my previous work, I have explored using game theory to design incentive structures for security games [137], as well as using reinforcement learning to design patrol strategies for wildlife conservation [157]. In this chapter, we continue this effort on environmental sustainability in another application. Here, we develop a toolkit which automatically detects and analyzes online articles related to environmental conservation and infrastructure construction using a BERT-based model. We emphasize two common issues in applied learning and planning pipelines: label scarcity and label quality. We use techniques from active learning and noisy label learning to address these challenges.

7.1 Introduction

Massive floods, poaching, waste pollution – every week, new threats impacting our environment come to light. Each of these can cause a long chain of negative impacts if not addressed. As such, monitoring these conservation-related events is of great importance for non-governmental organizations (NGOs) focused on environmental conservation such as the World Wide Fund for Nature (WWF) to take timely action and participate in relevant conversations.

In addition to conservation as a whole, many NGOs are particularly interested in monitoring news on certain subtopics. One such area is the ongoing or upcoming infrastructure projects such as roads, railways, and pipelines. These are usually more long-term and actionable than events like disasters or animal activity which occur in the past or present (hence limiting intervention impact). Conservation NGOs such as WWF play a key role in advocating for more sustainable infrastructure development. Early detection and engagement of these projects could shift infrastructure planning towards more environmentally sustainable outcomes while benefiting the people that the projects intend to serve.

However, information about conservation-related events and infrastructure plans threatening critical habitats is scattered across numerous sources and comes in different forms. NGOs typically learn of such information through word-of-mouth or a handful of news outlets that they check manually. This process is both time-consuming and ineffective, and it can poten-

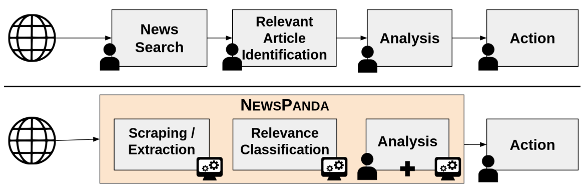


Figure 7.1: Top: Current costly and time-consuming information gathering pipeline at NGOs. Bottom: NewsPanda automates multiple steps in the pipeline, enabling humans to perform the more critical tasks (analysis and action).

tially fail to capture critical information in a timely manner, leaving these NGOs out of key conversations during early or ongoing stages of these developments.

To fill this gap, we develop **NewsPanda**, a natural language processing (NLP) toolkit to automatically detect and analyze news and government articles describing threats to conservation areas. **NewsPanda** has five main components, which we detail in Section 7.3. At the core of **NewsPanda** is a classification module built using a BERT-based language model, which we fine-tune to classify whether articles are relevant to conservation and to infrastructure.

Developing such a tool in the conservation nonprofit setting poses several unique challenges. First, labeling data is expensive. We propose an active learning-based method to selectively acquire labels on the most critical data points. Second, the data labels could be noisy since labeling for relevance is ultimately a subjective judgement, even if we fix a labeling rubric. We adopt a noise reduction algorithm [29] to improve our model's performance.

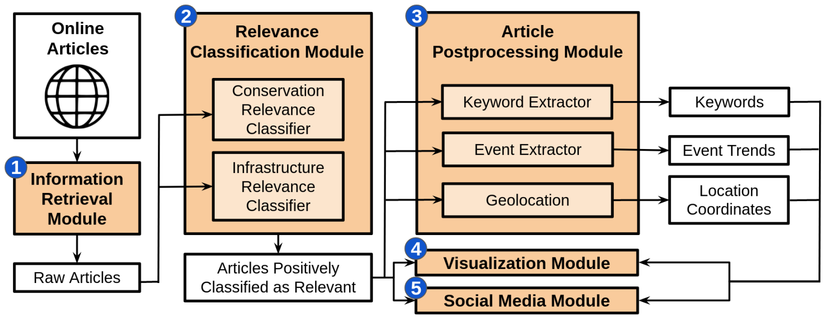
NewsPanda was developed as a collaboration between WWF and Carnegie Mellon University (CMU). It has been successfully deployed since February 2022 and has been used by the WWF teams in the UK, India, and Nepal to monitor developments in conservation sites. The entire pipeline runs on a weekly basis, scraping and classifying relevant news articles regarding conservation and infrastructure construction related events that occurred in the past week. These articles are then visualized in WWF's GIS systems for the field teams to investigate. We also share some results through social media for the benefit of the broader civil society. Through the deployment of **NewsPanda**, the WWF teams have been able to save over 30 hours weekly on collecting news, which allows us at WWF to instead focus on analyzing the news and taking actions (Figure 7.1) [1].

7.2 Related Work

News Monitoring Systems

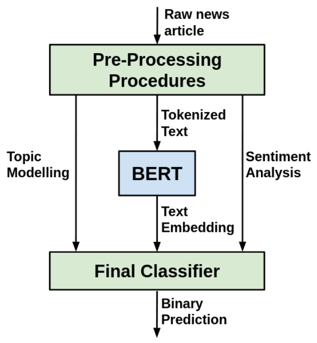
Although there is a rich literature on news information extraction in general domains [113, 129] as well as some specific applications [73, 103], there has been hardly any media monitoring tool for environmental conservation and infrastructure construction. Directly using generic media monitoring tools often lead to unsatisfactory results that are not localized enough to be actionable for a specific conservation site or not relevant enough to be reliable. As a result, conserva-

[1]We are happy to work with interested researchers and nonprofits on sharing our code and data.



Online Articles
1 Information Retrieval Module
Raw Articles
2 Relevance Classification Module
Conservation Relevance Classifier
Infrastructure Relevance Classifier
Articles Positively Classified as Relevant
3 Article Postprocessing Module
Keyword Extractor
Event Extractor
Geolocation
Keywords
Event Trends
Location Coordinates
4 Visualization Module
5 Social Media Module

(a) Diagram of overall **NewsPanda** pipeline, with the five key modules in orange boxes. Generated outputs of **NewsPanda** are in the white boxes.



(b) Conservation and infrastructure classification models.

Figure 7.2: NewsPanda pipeline (7.2a) and model diagram for conservation and infrastructure relevance classifiers (7.2b).

tion NGOs still use a manual process to collect articles. The only work on conservation news monitoring that we are aware of is a preliminary attempt by Hosseini and Coll Ardanuy [68] that apply BERT to classify news articles. Compared to that, with **NewsPanda** we provide a classification module with algorithmic contributions to address challenges in using the tool in the nonprofit context, a full end-to-end information extraction and processing pipeline, and most importantly, results and lessons learned from a large scale deployment of the tool. This is the first comprehensive and actionable media monitoring tool for conservation and infrastructure.

NLP for Conservation & Infrastructure

Outside of news monitoring, NLP tools have been used for various applications in conservation and infrastructure. Some analyze the relevant news articles for general insights on conservation reporting [133] or study their spread and impact [160]. These studies are descriptive in nature and orthogonal to our work. The few studies that take the civil society stakeholder's perspective are focused on different links in the process from us. Luccioni et al. [94] use BERT-based models to analyze corporate environment sustainability reports. Boutilier and Bahr [21] explore mining-related texts to analyze the social license of a particular project. They target different problems from us. They assume a relevant text is readily available and try to extract meaningful insights from it. On the other hand, we work on identifying that relevant text from thousands of irrelevant texts in the first place and leave the insight extraction to professional organizations like WWF that have been doing that for years.

7.3 NewsPanda Overview

NewsPanda toolkit consists of five modules as illustrated below and in Figure 7.2a. During pilot study and deployment (Section 7.8), this entire pipeline is run on a weekly basis.

1. **Information Retrieval Module**: We use the NewsAPI scraper [89] with the names of conservation sites taken from a curated list of conservation areas.
2. **Relevance Classification Module**: We classify articles along two dimensions, namely *Conservation Relevance* and *Infrastructure Relevance*, through a large pretrained language model fine-tuned with our collected dataset. Details of this model are explained in Section 7.5.
3. **Article Postprocessing Module**: The article postprocessing module has 3 parts: a keyword extractor which extracts keywords, an event extractor which extracts event trends, and a geolocator which provides location coordinates. We discuss these features in Section 7.6.
4. **Visualization Module**: After the relevant articles are identified, we visualize them in our GIS system at WWF, which we can further analyze and act upon (Section 7.8).
5. **Social Media Module**: In parallel to the visualization module, another downstream application for **NewsPanda** is WildlifeNewsIndia, [2] a Twitter bot we built from **NewsPanda** that shares weekly relevant conservation-related articles on social media (Section 7.8).

7.4 Dataset

We use two main datasets for developing **NewsPanda**. First, we use an existing corpus (WHS-Corp) by Hosseini and Coll Ardanuy [68] consisting of articles scraped using World Heritage Sites as keywords and labelled by domain experts. Second, we scrape and label our own corpus (InfraCorp), which is a more focused, timely, and fine-grained upgrade over WHS-Corp. The datasets differ in terms of the locations of the conservation sites used, as well as the time frame of the articles.

7.4.1 WHS-Corp Dataset

WHS-Corp contains over 44,000 articles from 2,974 different sources covering 224 World Heritage Sites globally. Scraping was done using NewsAPI's Python library from a list of curated conservation sites of interest. Besides the title and content, it also contains metadata such as the publication site, the author, and the date of publication. Articles in WHS-Corp span from January 2018 to October 2019.

After these articles were gathered, a subset of 928 articles were sampled and manually annotated for *Conservation Relevance* by domain experts familiar with conservation. *Conservation Relevance* denotes whether an article discusses threats or impacts to wildlife and environment conservation in general, e.g. poaching, forest development, natural disasters. We use this labelled dataset for training our model.

[2]https://twitter.com/WildlifeNewsIND

7.4.2 InfraCorp Dataset

As opposed to WHS-Corp which focuses on global conservation sites, InfraCorp specifically focuses on conservation sites in India and Nepal. The InfraCorp corpus contains 4,137 articles (150 for Nepal and 3,987 for India) from 1,074 conservation sites across the two countries. All articles were taken in the two-year span from November 2019 to November 2021. We use NewsAPI to search for the official names of the conservation sites, or alternative and/or local names for the sites as recorded at WWF.

Given the data availability as well as the annotator capacity of the local domain experts from India and Nepal, we labeled all 150 articles from Nepal and only 1,000 articles from India. Annotation for InfraCorp was done along two dimensions: *Conservation Relevance* and *Infrastructure Relevance*. *Conservation Relevance* is similar to the one described for WHS-Corp in Section 7.4.1. Among the articles which were labelled as positive for *Conservation Relevance*, we further categorize whether it is relevant to infrastructure. This covers issues such as new roads in forested areas and construction projects near national parks. Each article was annotated by two domain experts, one from WWF UK, and another from either WWF India or WWF Nepal. We provided the annotators with a descriptive rubric for labeling in each dimension, as well as concrete examples of edge cases. The following was one such example in our instructions:

> Articles describing tourism or wildlife or natural beauty of a national park, but without talking about environmental impacts or threats to wildlife and conservation, do not count as positive for *Conservation Relevance*.

Where the two sets of labels disagree, the authors closely inspect the articles and decide on the final labels.

7.5 Relevance Classification Module

We highlight the structure of our **NewsPanda** classification module and other key techniques used during training.

7.5.1 Classification Model

The backbone of the **NewsPanda** classification model is a BERT model [40] with a linear classification head. BERT is a Transformer-based language model trained using masked language modelling and next sentence prediction objectives on large-scale corpora of books and articles. This large-scale pretraining, as well as its ability to effectively encode context, leads to superior performance on a wide variety of tasks. We adapt BERT to the domain of conservation and infrastructure, and we fine-tune it to perform news article classification. In Section 7.7, we explore different variants of the BERT model (such as RoBERTa).

One key change we make to the BERT model is that in the final linear head after the main BERT layers, instead of only considering the BERT vector outputs, we also incorporate other features, namely sentiment analysis and topic modelling, as shown in Figure 7.2b. We hypothesize that including these additional features will provide the model with more useful information that will help classify whether or not a particular article is relevant to infrastructure or conservation. For instance, if an article has topic vectors that align with other articles covering forest habitats, but it has an overwhelmingly positive sentiment, then we may suspect that it could be a tourism-related feature article instead of a conservation-related news article (which are often more neutral or negative in terms of sentiment).

For sentiment analysis, we extract the sentence polarity scores of the article title, its description, and its content, giving us three sentiment scores per article. This is done on a scale of −1.0 to +1.0, with −1.0 representing the most negative score and +1.0 representing the most positive score. Sentiment analysis was done using the `textblob` package [93]. Meanwhile, for topic extraction, we consider the entire training corpora of WHS-Corp and InfraCorp, and train a Latent Dirichlet Allocation (LDA) model to identify topic clusters. We use 50 topics for the LDA model and implemented it using `scikit-learn` [116]. Lastly, for the main BERT model, we concatenate the title, description, and content of each article, and we use this concatenated text as input to our classifier. For cases where the article is missing certain features (e.g. no description), we simply supply an empty string for that feature. The vectors from the three steps (i.e. BERT model, sentiment analysis, topic modelling) are then concatenated, and this final vector is used as the input to the final classification head to generate a binary prediction. Specific implementation settings and other hyperparameters can be found in Section 7.7.1.

7.5.2 Active Learning

Annotating a dataset is costly. In curating our InfraCorp dataset, we need to be mindful of which specific articles to label in order for our model to learn most efficiently. For this selection process, we first fine-tune a pretrained RoBERTa-base model on the existing WHS-Corp dataset, based on the *Classification Relevance*. To make this preliminary model as close to our final model as possible, we also incorporate the topic modelling and sentiment analysis features, as shown in Figure 7.2b. Because this is only a preliminary model, we forego doing extensive hyperparameter tuning and decided to just select a setting that worked decently well: with a learning rate of 1e-5, batch size of 16, and training for 10 epochs, we were able to get an F-score of 0.61 on WHS-Corp. Using this trained model, we then generate *Classification Relevance* predictions for all articles in the InfraCorp corpus, together with the corresponding softmax scores. We treat these softmax scores as a measure for the classification confidence of the model: if the softmax is close to 0 or close to 1, then it means that the model is very certain with its prediction, while if the softmax is close to 0.5, then it means the model is unsure with its prediction.

We then select 300 articles which our model is least confident about. We hypothesize that selecting these "difficult" rows will have the greatest impact on model performance. We call this active learning-based dataset InfraCorp-A. To verify the effectiveness of active learning, we also randomly sample 300 articles to label, which we call InfraCorp-R. We will later evaluate how this compares with the actively selected dataset on a randomly selected test set of 400 samples in our ablation study (Section 7.7.3).

7.5.3 Noisy Label Correction

Our dataset is labelled by two sets of domain expert annotators from WWF. Although we provided detailed criteria for labelling each article, there is always room for some subjectivity in the process. This resulted in the two sets of labels not agreeing with each other on over 10% of the data points. Although, as mentioned in Section 7.4.2, we did manage to obtain the "ground truth" label for a small subset of InfraCorp for model evaluation purposes, doing that for every single article is prohibitively expensive – much more expensive than the (not cheap) process of having either annotator providing a (noisy) label. Therefore, in order for **NewsPanda** to work well once deployed, we need to be able to learn well from the potentially noisy labels only.

More formally, let x_n be the embedding of an article along with its sentiment and topic modeling vectors as described in Section 7.5.1. Let y_n be the true label of this article. The task is to make an accurate prediction on the dataset $\{(x_n, y_n) : n = 1 \dots N\}$ when we only have access to the noisy data $\{(x_n, \tilde{y}_n) : n = 1 \dots N\}$ where $\tilde{y}_n$ is the label that we get from either of the two annotators, and the true labels y_n are the final labels that we decide on after resolving conflicts.

To address this challenge, we adapt the CORES^2 loss [29] noise correction algorithm, which is an extension of the earlier peer loss [91]. Peer loss frames the task of learning from noisy labels as a peer prediction problem. In practice, the loss for each (x_n, y_n) data point can be calculated using the standard cross entropy loss with (x_n, y_n), modified with a loss calculated using a randomly sampled input x_{n_1} and an *independently* randomly sampled label y_{n_2}. That is, we have

$$\ell_{\text{PEER}}(f(x_n), \tilde{y}_n) := \ell(f(x_n), \tilde{y}_n) - \alpha \cdot \ell(f(x_{n_1}), \tilde{y}_{n_2})$$

where $\alpha > 0$ is a tunable parameter. Meanwhile, CORES^2 replaces the random sampling from peer loss with a confidence regularizer defined as follows:

$$\ell_{\text{CORES}}(f(x_n), \tilde{y}_n) := \ell(f(x_n), \tilde{y}_n) - \beta \cdot \mathbb{E}_{\mathcal{D}_{\tilde{Y}|\tilde{D}}}[\ell(f(x_n), \tilde{Y})]$$

where $\tilde{D}$ is the dataset, $\tilde{Y}$ is a noisy label, and $\beta > 0$ is a tunable parameter. Following Cheng et al. [29], we calculate this confidence regularizer term using an estimate of the noise prior probability. We test both peer loss and CORES^2 loss, and report results in our ablation study (Section 7.7.3).

7.6 Article Postprocessing Module

Once the relevant articles are identified using the model, we then perform a few post-processing steps to extract key information and make them easier to analyze and visualize.

7.6.1 Keyword Extractor

Keywords are important, as they allow the easy summarization, categorization, and grouping of news articles. Furthermore, we also use these keywords as hashtags in our social media module (Section 7.8). To extract keywords, we use an extensive list of conservation-related keywords maintained at WWF. and search the article for exact matches. In addition, we also use Named Entity Recognition systems to extract the salient words in each article. To perform this, we

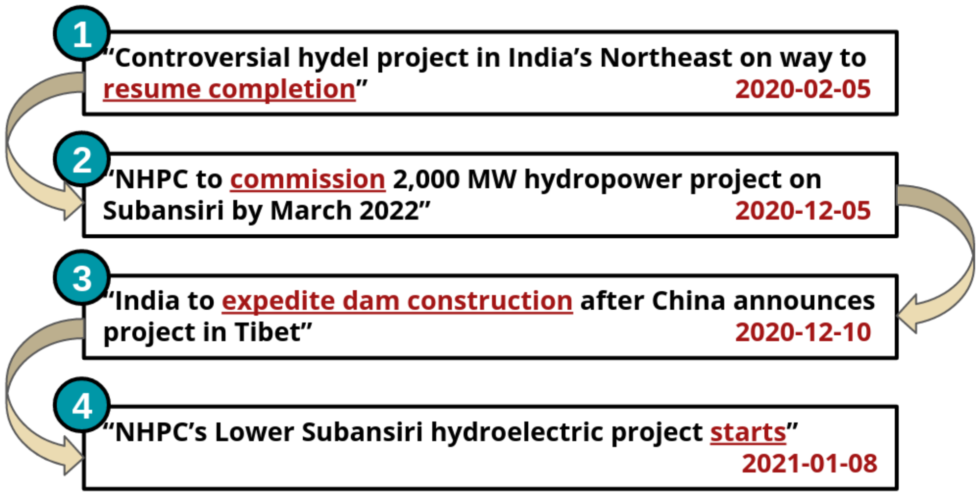

1
"Controversial hydel project in India's Northeast on way to resume completion"
2020-02-05
2
"NHPC to commission 2,000 MW hydropower project on Subansiri by March 2022"
2020-12-05
3
"India to expedite dam construction after China announces project in Tibet"
2020-12-10
4
"NHPC's Lower Subansiri hydroelectric project starts"
2021-01-08

Figure 7.3: Example of events selected by the Event Extractor (Section 7.6.2) by date. The progression of the project is highlighted by the phrases in red underline.

use a BERT-based model trained on the CoNLL 2003 Named Entity Recognition dataset [152]. The keywords extracted using these two methods are then concatenated to form the final set of keywords.

7.6.2 Event Extractor

To track the progress of infrastructure projects, it is often not enough to just view a single article in isolation. Rather, news regarding these projects often builds up over a period of weeks or months. To help provide this context, we create an automated event extractor, which leverages our InfraCorp dataset, including both the labelled articles as well as the unlabelled articles. Given a new article a, our goal is to find past articles P_a which are closely related to a. We first gather all previous articles which are from the same conservation site. Next, we create a graph G_a, where each article is a node, and two nodes share an edge if the corresponding articles share $\geq k$ common keywords (from Section 7.6.1). Here, k is an adjustable parameter depending on how loosely connected we want G_a to be. For our data, we use $k = 3$. Once the graph G_a is constructed, we then define an "event" to be the maximal clique containing a, and we report all such events. A sample chain of events is shown in Figure 7.3.

7.6.3 Geolocation

To aid with visualization (Section 7.8), we perform geolocation on the classified news articles, based on the search terms used to retrieve them. To extract latitude and longitude coordinates, we leverage an extensive directory of conservation sites from WWF, and we use the directory to map conservation sites to their corresponding coordinates. If the directory contains no match, we geolocate using the `geopy` package.

7.7 Experiments and Results

Here, we discuss results of our in-lab experiments and ablation studies to verify our hypotheses. Results from real-world deployment are discussed in the succeeding section.

7.7.1 Experiment Settings

Baselines

We compare the performance of our **NewsPanda** model with the following baselines:

1. **Keyword model**: We consider a naive model that checks for the count of certain keywords. We curate two sets of "conservation-related keywords" and "infrastructure-related keywords". If an article contains more than k "conservation-related keywords", then it is considered to be relevant to conservation (likewise for infrastructure).
2. **RNN-based models**: We tokenize each article, then pass the embedding to RNN models, where the hidden state of the last layer is used as input to the final classification layer. We use two types of RNN models, namely GRUs [14] and LSTMs [66].
3. **BERT-based models**: We fine-tune a pretrained BERT-base [40] and RoBERTa-base model [92], where we add a classification head after the final layer to perform relevance classification.

Evaluation Metrics

Since our task is binary classification, we measure the accuracy, precision, recall, and F1-score. For precision, recall, and F1, we consider only the scores of the positive class. All metrics are calculated separately for *Conservation Relevance* and *Infrastructure Relevance*.

Data

For *Conservation Relevance*, we train on the InfraCorp dataset (consisting of both InfraCorp-A and InfraCorp-R), as well as the WHS-Corp dataset. For *Infrastructure Relevance*, since WHS-Corp does not contain infrastructure labels, we only train using InfraCorp. We split the training data into an 80-20 training-validation split. For evaluation, we use the test split of InfraCorp for both *Conservation Relevance* and *Infrastructure Relevance*.

Implementation Settings

For the GRU/LSTM, we use a batch size of 128, hidden size of 128, and dropout of 0.2. We train for 10 epochs with a learning rate of 1e-4. Meanwhile, for BERT, RoBERTa, and **NewsPanda**, we train for 10 epochs with batch size 4 and learning rate 1e-5. We use RoBERTa for the backbone model of **NewsPanda**. Model selection is done by considering the best validation F1-score.

Model	Acc.	P	R	F1
Keyword	0.820	0.317	0.634	0.423
LSTM	0.711	0.495	0.511	0.504
GRU	0.729	0.422	0.505	0.475
BERT	0.860	0.708	0.704	0.706
RoBERTa	0.867	0.705	0.743	0.721
NewsPanda	**0.877**	**0.729**	**0.801**	**0.744**

Table 7.1: Average scores for *Conservation Relevance*, taken over 10 random seeds.

Model	Acc.	P	R	F1
Keyword	**0.947**	0.250	0.455	0.323
LSTM	0.908	0.566	0.537	0.554
GRU	0.895	0.544	0.557	0.553
BERT	0.922	0.840	0.745	0.771
RoBERTa	0.916	0.794	0.809	0.799
NewsPanda	0.941	**0.880**	**0.821**	**0.850**

Table 7.2: Average scores for *Infrastructure Relevance*, taken over 10 random seeds.

7.7.2 Results and Analysis

Experimental results are shown in Tables 7.1 and 7.2. We observe that indeed, adding the sentiment analysis and topic modelling features, as well as the CORES^2 loss for noisy label correction, aids in predictions for both *Conservation Relevance* and *Infrastructure Relevance*, providing an improvement over both BERT-base and RoBERTa-base.

Our data is quite imbalanced: >80% of the articles are not relevant. This manifests itself in the discrepancies between accuracy and F1-score. We observe, for example, that the naive keyword model has very high accuracy scores but very low F1-scores, which indicates that it predicts a lot of zeros (hence the high accuracy), but is not able to predict the relevant articles well. The RNN-based models (LSTM and GRU) seem to perform relatively poorly, achieving an F1-score of around 0.5. This could also be attributed to the data imbalance, since these RNN-based models are generally not as robust to imbalanced datasets. In contrast, the BERT and RoBERTa models perform quite well, with F1-scores >0.7 for conservation and >0.75 for infrastructure, and precision/recall scores also around that range. This indicates that these transformer-based models are able to generalize quite well and successfully capture the notions of *Conservation Relevance* and *Infrastructure Relevance*. Lastly, **NewsPanda** offers significant improvement over the RoBERTa-base model (F1 t-test p-value = 0.018 for conservation and 0.033 for infrastructure), showing the positive effects of incorporating information such as the emotion and topics over simply considering the article text in isolation.

Dataset	Acc.	P	R	F1
WHS-Corp	0.911	0.585	0.585	0.586
WHS+Inf.Corp-A	**0.921**	**0.600**	**0.774**	**0.670**
WHS+Inf.Corp-R	0.916	0.586	0.696	0.637

Table 7.3: Evaluation scores for *Conservation Relevance* for InfraCorp-A compared with InfraCorp-R, averaged over 10 random seeds.

Noisy Label Correction	Acc.	P	R	F1
None	0.907	0.566	0.441	0.497
Peer Loss	**0.911**	**0.591**	0.465	0.509
CORES^2	0.908	0.584	**0.551**	**0.553**

Table 7.4: Evaluation scores for *Conservation Relevance* for two noise correction methods, over 10 random seeds.

7.7.3 Ablation Study

Active Learning

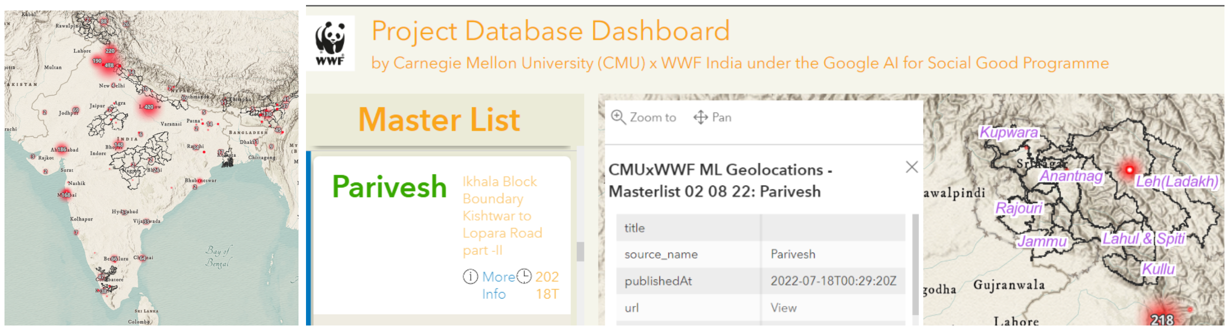
We compare the effect with training on actively-sampled data (InfraCorp-A) and randomly-sampled data (InfraCorp-R). Each of these datasets contain 300 India articles, as detailed in Section 7.5.2 and 7.4.2. We append these articles to the existing WHS-Corp to create the final data for training. We use the RoBERTa model for these experiments. Results are shown in Table 7.3.

For both InfraCorp-A and InfraCorp-R, we see an improvement over just using WHS-Corp. Indeed, training with more data will result in better performance, regardless of how the data is sampled. We also observe that adding actively sampled data results in a larger improvement than adding randomly sampled data across all metrics (F1 t-test p-value = 0.004). This verifies the effectiveness of our hypothesized confidence-based data selection for annotation.

Noisy Label Correction

We examine the effect of the noise correction methods outlined in Section 7.5.3, by comparing the effect of using peer loss, CORES^2 loss, and standard cross entropy loss. Based on InfraCorp, we use the labels supplied by one of the two annotators for the training set, and the final calibrated labels for the test set. Hyperparameter search was done for both peer loss and CORES^2 loss to find the optimal values of $\alpha = 0.05$ and $\beta = 0.05$. We trained for 20 epochs with a learning rate of 2e-5.

From Table 7.4, we observe that for accuracy and precision, all three losses perform very similarly, with peer loss performing the highest by a small margin. For recall and F1, peer loss and the standard loss perform at comparable levels, while CORES^2 loss performs better than both (F1 t-test p-value = 0.001). This is likely because the confidence regularizer used in CORES^2 works better than the random sampling used by peer loss. Both peer and CORES^2 loss might


Project Database Dashboard
by Carnegie Mellon University (CMU) x WWF India under the Google AI for Social Good Programme
Master List
Parivesh
Ikhala Block Boundary Kishtwar to Lopara Road part -II
More Info
Zoom to
Pan
Kupwara
Anantnag
Leh(Ladakh)
Rajouri
Jammu
Lahul & Spiti
Kullu
Gujranwala
Lahore

CMUxWWF ML Geolocations - Masterlist 02 08 22: Parivesh

title	
source_name	Parivesh
publishedAt	2022-07-18T00:29:20Z
url	View

Figure 7.4: Left: The highlighted red areas indicate clusters of articles found by our model. Right: The WWF GIS system, where each relevant article is shown on the map with its corresponding key details.

work even better if we had more training data than the current 600 in InfraCorp. In the end, given the positive results of CORES^2, we used it in our **NewsPanda** model.

7.8 Deployment and Impact

NewsPanda has been used at WWF since February 2022. We describe the deployment, results, and lessons learned.

7.8.1 Pilot Study

The first stage of **NewsPanda** deployment, which is the pilot study, started in February 2022 and ran for around one month. Every week, the CMU team scraped the news articles and ran the entire **NewsPanda** pipeline, forwarding the outputs to the WWF teams to examine and provide feedback. During this pilot phase, the WWF and CMU teams identified a range of operational and technical issues in the initial version of **NewsPanda**.

First, in order for **NewsPanda** to fit into the established workflow of WWF, it needs to be integrated into its GIS system. During the pilot, we realized that it is crucial to add the geolocation of each article (Section 7.6.3) and format the model output according to the specifications of the GIS platform used at WWF. Figure 7.4 shows how **NewsPanda**'s results get integrated into the GIS system, with the red areas being the locations where we identify a relevant article.

We also discovered that while NewsAPI has a good collection of global news sources, it fails to include some relevant sources in the local context. With the suggestions from the WWF team, we incorporated additional sources that often yield relevant local articles. One such site is Parivesh, which contains proposals of infrastructure projects in India.

Finally, we found that some conservation sites' names often lead to 0 results, while other terms were too general and yielded hundreds of results, almost all of which were irrelevant, leading to inefficiencies. We set a lower and upper threshold, and filter out search terms outside the thresholds.

7.8.2 Deployment Results

After we resolved the above issues, we proceeded with the actual deployment. The procedure was similar to the pilot phase, except that at this phase, the focus is to evaluate the performance of **NewsPanda**. The WWF teams closely inspected the model predictions each week and provided ground truth labels for each article. The label feedback allowed the CMU team to retrain the model regularly. This stage ran from March 2022 to July 2022. Table 7.5 shows the aggregated results over 5 months of evaluation results from WWF India, Nepal, and UK. WWF UK labeled the first half of the deployment for all locations and India/Nepal labeled the second half for news articles in their respective countries.

Overall, **NewsPanda** continued to show great performance in *Conservation Relevance* during real-world deployment. Across all evaluations, the precision scores are consistently high, indicating that almost all of the articles reported by **NewsPanda** are indeed relevant. We intentionally tuned the model towards this direction – when almost everything that the model flagged is relevant, it would greatly help with establishing the trust in the model at the early stage of deployment. As we continue developing the model, we aim to improve the model towards achieving higher recall, to be able to capture more relevant articles.

On the other hand, on *Infrastructure Relevance* for India, the model's performance was worse than the offline experiments. Upon further inspection, we discovered that the majority of mistakes were in fact only 2-4 original pieces of news that were paraphrased by various news sources into 20-40 articles. Since there are only a few *Infrastructure Relevance* positive articles to start with, this had a big impact on the model performance. Meanwhile, such phenomenon did not occur in our offline experiments because there we randomly sampled news from a large corpus for labeling.

Aside from overall metrics, we also highlight individual success stories. Figure 7.4(right) shows a concrete example where **NewsPanda** made a difference. In early August, 2022, **NewsPanda** detected a new project of Ikhala Block Boundary Kishtwar to Lopara Road and highlighted it in the WWF GIS system. Upon further investigation by WWF staff, it is found that the project would divert 5.9 hectares of forest land. More importantly, WWF found that the project was still at its pre-proposal stage. This means WWF would be able to take early action and possibly participate in relevant conversations. Such stories are happening frequently since the deployment of **NewsPanda**. Using the tool's outputs integrated into our internal GIS systems, the WWF staff are continuously coordinating with our field teams to examine the status and report on relevant projects and areas.

7.8.3 Qualitative and Quantitative Comparison with Current Practice

Prior to **NewsPanda**, WWF had already been monitoring media for conservation-related articles (Figure 7.1). However, most of these efforts were not very structured or logged. It is thus difficult to draw head-to-head comparisons between **NewsPanda** and WWF's existing approach. That said, we still provide qualitative and quantitative evidence supporting the merit of **NewsPanda** over the current practice.

Two months into the deployment, the CMU team carried out semi-structured interviews with their WWF colleagues who have been using **NewsPanda** outputs in their work. The

	Conservation			Infrastructure		
	P	R	F1	P	R	F1
India	0.849	0.605	0.706	0.462	0.250	0.324
Nepal	0.895	0.917	0.906	0.923	0.308	0.462
UK	0.879	0.823	0.850	1.000	0.455	0.625

Table 7.5: Aggregated scores of NewsPanda on weekly articles from March 2022 to July 2022.

purpose was to understand how WWF teams liked the toolkit and to elicit possible suggestions for improvement. Some quotes from the interviews are as follows.

> "You're giving us a bunch of articles... over 50 articles a week. We had two interns who spend 2-3 days a week on this and would only give us seven to ten articles. So there is a huge bump in efficiency right there in itself."

> "The data that you're sharing give a global perspective. It is very useful to understand the upcoming projects or mitigation measures that are being adopted on a global scale. So it helps us be informed."

This improvement in news collection also helped with the downstream task – infrastructure impact assessment.

> "It took us maybe a month to do analyses of three or four infrastructure projects. With **NewsPanda**, we can send (stakeholders) 20 or 30 reports in a month."

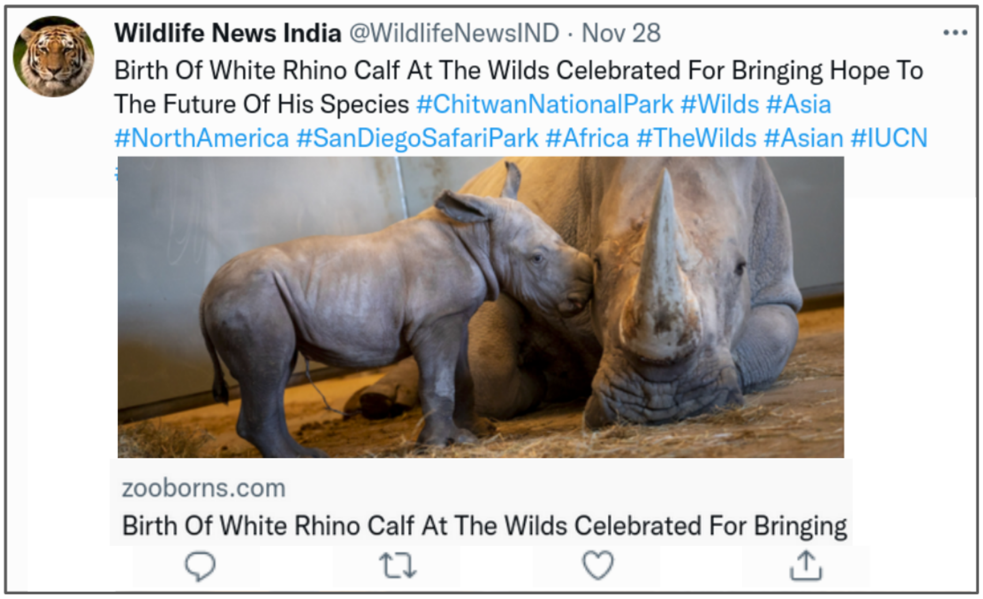
The micro-level improvement in this single task has also resulted in macro-level organizational change:

> "It's also a transition in their (WWF staff) job function. They will not just be doing data hunting. They are qualifying themselves to be data analysts."

The WWF Nepal team has been putting together weekly news digests for conservation sites in Nepal. Although this dataset is small and has no negative labels, this is the only quantitative comparison between **NewsPanda** and current practice we can make. We find that our model is able to identify 62% of the articles in the news digest. This is a relatively good performance as we had extremely limited articles (only 150) about Nepali conservation sites to train the model.

7.8.4 Sustainable Deployment and Broader Impact

Encouraged by the success of **NewsPanda** at the initial stages, we are working to scale it to more sites and permanently deploy **NewsPanda** as part of the WWF computing infrastructure. We have been collecting news articles for over 60,000 sites globally and applying our trained model to classify them on a weekly basis since April 2022. Because the main model has already been trained, we no longer need extensive data labeling for evaluation. Instead, we only need a small subset for model update and fine-tuning purposes. We are currently investigating the ability **NewsPanda** to generalize to new locations and new languages given only a few (or even zero) domain-specific training points. We are also shifting our system to a cloud


Wildlife News India @WildlifeNewsIND · Nov 28

Birth Of White Rhino Calf At The Wilds Celebrated For Bringing Hope To The Future Of His Species #ChitwanNationalPark #Wilds #Asia #NorthAmerica #SanDiegoSafariPark #Africa #TheWilds #Asian #IUCN

zooborns.com

Birth Of White Rhino Calf At The Wilds Celebrated For Bringing

Figure 7.5: Sample tweet of WildlifeNewsIndia

server to be owned and maintained by the WWF team, rather than the CMU team, to ensure sustainable deployment. The CMU team will continue to provide support and tutorials to help WWF eventually grow in-house capability of sustaining the project.

Much as this project was a collaboration between WWF and CMU, **NewsPanda** could also be valuable to the broader civil society. Thus, we also developed a social media module in the form of a Twitter bot called WildlifeNewsIndia. The bot periodically tweets a selected set of the identified relevant articles. In addition to tweeting links to articles, we also use the keywords from **NewsPanda**'s keyword extractor (Section 7.6.1) to generate salient hashtags. A sample tweet is shown in Figure 7.5. Currently, WildlifeNewsIndia is focused on conservation-related articles in India. As we continue working on this project, we hope to scale this to a global level, so that any organization or individual interested in conservation can benefit from the tool.

7.8.5 Lessons Learned

This 1.5 year long and counting collaboration has yielded many valuable lessons for both WWF and CMU. We have already mentioned some of those in earlier sections. We highlight two more generalizable lessons below.

Problem identification is an iterative process and rapid prototyping helps surface unforeseen needs. The event extractor in Section 7.6.2 was not initially part of the agenda: without a prototype of the model readily available, it was difficult for WWF to realize what could be done with it. However, after several iterations of communication and exploring results, the need to track the development related to a single project/location became clear to us. This was made possible by the rapid prototyping, where the CMU team used viable algorithms that may not be optimal but are quick to implement to demonstrate the possibilities of the toolkit.

It is the various "not-so-AI" components that realize the promise of an AI for nonprofit project on the ground. While the classification module in Section 7.5 is the engine of **NewsPanda**, the postprocessing module in Section 7.6 and the visualization module in Figure 7.4 are key in getting the information in a consumable format, and ultimately the buy-in at WWF. Each of the latter two modules requires at least as much engineering effort and careful design as the classification module. We call on future AI for nonprofit projects to pay enough attention to all the infrastructure around the AI part, in order to deliver the real impact that we hoped for.

7.9 Conclusion

In this chapter, we designed and deployed **NewsPanda**, a toolkit for extracting, classifying, and analyzing articles related to conservation and infrastructure. We showed empirically that our **NewsPanda** model classifies better than baseline methods for both *Conservation Relevance* and *Infrastructure Relevance*. We also presented quantitative and qualitative evaluations of our system in the real world as well as its impact on WWF teams in UK, India, and Nepal.

Currently **NewsPanda** mainly focuses on a few countries, and we are expanding it to a global scale. However, incorporating additional conservation sites is just the beginning. To do it right, we also need to cover more languages and more local media sources. This is especially important for the global south, as many high-impact local developments might never reach international news outlets. The ability to capture these local sources, especially if they are not written in English, is something we are currently working on. We are currently starting with articles written in the Nepali language. Initial experiments with a multilingual version of **NewsPanda** have shown good generalization when given only a few Nepali articles for training. With this multilingual model, we hope to further expand to cover a wider array of languages.

Ethical implications **NewsPanda** is intended to help conservation organizations such as WWF streamline their operations and advocate for environmental consideration in economic development. In this process, it is important that the model does not ignore events in certain regions or over-concentrate on events in other regions. While developing **NewsPanda**, we routinely checked with our WWF collaborators whether they thought the model's output was balanced across geographies based on their domain expertise. More quantitative analysis could be carried out in this regard. In the meantime, **NewsPanda** does automate some part of the work at WWF. However, as commented in Section 7.8, the WWF staff who used to collect such news articles are now qualifying themselves to be data analysts. In this sense, **NewsPanda** is facilitating broader organizational change in a positive direction.

Part IV

Learning and Planning Towards AI for Social Good

Chapter 8

Bandit Data-driven Optimization: AI for Social Good and Beyond

AI for social good applications, and many other machine learning (ML)-based systems that are deployed in the field, feature an iterative process which joins prediction, optimization, and data acquisition. We introduce bandit data-driven optimization, the first iterative prediction-prescription framework to formally analyze this practical routine. Bandit data-driven optimization combines the advantages of online bandit learning and offline predictive analytics in an integrated framework. It offers a flexible setup to reason about unmodeled optimization objectives and unforeseen consequences. We propose PROOF, a novel algorithm for this framework and show that it achieves no-regret. Using numerical simulations, we show that PROOF achieves superior performance than existing baseline. We also apply PROOF to a food rescue task with real data, and show that PROOF as a framework works well with the intricacies of ML models in real-world applications.

8.1 Introduction

The success of modern ML largely lies in supervised learning, where one predicts some label c given input feature x. Off-the-shelf predictive models have made their ways into numerous commercial applications. Such tangible progress has motivated the community to address more real-world societal challenges, as evidenced by the growing research theme of AI for social good (AI4SG).

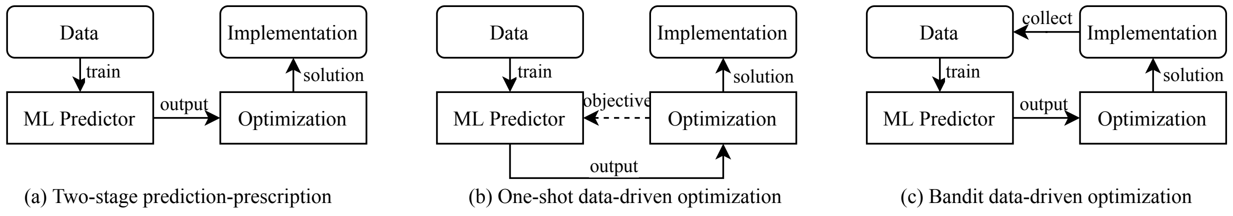


Figure 8.1: Paradigms of how ML systems are used in realistic settings.

Unfortunately, the success of ML often does not translate directly into a satisfactory solution to a real-world AI4SG problem. One obvious reason is supervised learning focuses on prediction, yet real-world problems, by and large, need prescription. For example, rather than predict which households' water pipes are contaminated (labels) using construction data (features), municipal officials need to schedule inspections (interventions) [3]. The common practice is a two-stage procedure, as shown in Figure 8.1a. After training an ML model, the user makes prescriptive decisions based on some optimization problem parameterized by the prediction output. In an emerging line of work on (one-shot) data-driven optimization, the learning problem is made aware of the downstream optimization objective through its loss function, gluing the two stages together [19, 44]. We illustrate this in Figure 8.1b.

However, this is still far from a complete picture. Figure 8.1c shows a typical workflow in many AI4SG projects, such as the food rescue ones we introduced in Part II. After getting data from the collaborating organization, the researcher trains an ML model and then, based on it, recommends an intervention. Using the new data collected under the new intervention, the researcher updates the ML model and recommends a new intervention, so on and so forth, leading to an iterative process. The principles of these steps are often not aligned. Without a rigorous, integrated framework to guide the procedure, this could lead to operation inefficiency, missed expectations, dampened initiatives, and new barriers of mistrust which are not meant to be.

Such an iterative process is necessary due to the following key features of AI4SG applications distilled from existing research under this theme [119]. First, there may not be enough data to begin with. Many of these domains do not have the luxury of millions of training examples. A small dataset at the beginning leads to inaccurate predictions and hence suboptimal decisions, but they will improve as we collect more data, as seen in, e.g., predicting poaching threats from patrol data and designing ranger patrols [52]. Second, too often the initial dataset has some default intervention embedded, while the project's goal is to find the optimal intervention. For example, Shi et al. [142] design a smartphone notification scheme for a volunteer-based platform but existing data are all collected under a default suboptimal scheme. If one expects the data distribution to vary across interventions, one has to try out some interventions and collect data under them. Third, we may not perfectly know the the correct objective function to optimize. This is especially true considering the knowledge and communication gap between AI researchers and domain practitioners. Fourth, the proposed interventions may have unexpected consequences. This hints at the inherent impossibility of fully modeling the problem in one shot.

We propose the first iterative prediction-prescription framework, which we term as *bandit data-driven optimization*. This framework combines the relative advantages of both online bandit learning and offline predictive analytics. We achieve this with our algorithm PRedict-then-Optimize with Optimism in Face of uncertainty (PROOF). PROOF is a modular algorithm which can work with a variety of predictive models and optimization problems. Under specific settings, we formally analyze its performance and show that PROOF achieves no-regret. In addition, we propose a variant of PROOF which handles the scenario where the intervention affects the data distribution and prove that it also enjoys no-regret. Using numerical simulations, we show that PROOF achieves much better performance than a pure bandit baseline. We also apply PROOF in a case study of a real-world AI4SG project on food rescue.

We emphasized AI4SG as the motivation for bandit data-driven optimization. That said, many ML-based systems deployed in the field share similar pain points and can benefit from this framework.

8.2 Related Work

To our knowledge, there is surprisingly no existing work that rigorously studies the procedure as shown in Figure 8.1c. We introduce several lines of work with similar goals below and a summary of the differences can be found in Table 8.1.

First, (one-shot) data-driven optimization is characterized by a dataset consisting of features $x_1, \ldots, x_n$ and labels $c_1, \ldots, c_n$. The task is to find the action w^* that maximizes the expected value of objective function $p(c, w)$ given some feature x, i.e. $w^* = \arg\max_w \mathbb{E}_{c|x}[p(c, w)]$. The first approach comes from the stochastic programming perspective [16, 19]. Another popular approach is referred to as the predict-then-optimize framework [44, 65, 74]. There, one learns an ML predictor f from data and then optimize $p(c, w)$ with the predicted label $c = f(x)$. Compared to our bandit data-driven optimization framework, this entire literature assumes that the optimization objective is known a priori and does not consider multi-period settings.

More generally, our work touches on optimization under uncertainty. Zheng et al. [169] and Chen et al. [25] provide sample complexity bounds for learning the parameters of an optimization problem. However, they do not learn the data distribution. Balkanski et al. [15] consider optimizing a submodular function with samples. Yet, in all these works, there is no clear way to leverage the feature/label dataset that is so common in real-world AI4SG tasks.

Contextual bandit (CB) is a well-studied online decision-making model very relevant to our framework [13, 79]. At each time step t of CB, one receives feature x_t, picks an action w_t, and receives reward whose expectation is some unknown function of x_t and w_t. In fact, our bandit data-driven optimization framework reduces to CB if we skip the training of a predictive model and directly pick an action. However, by doing so, we effectively give up all the valuable information in the historical data. Although CB algorithms achieve no-regret and have been successful in high-frequency decision-making [85], they are often impractical in AI4SG applications. It would hardly be acceptable to any stakeholders that the algorithm only guarantees good results after using it for, say, 10 years, while we choose not to use the dataset already available. That said, bandit provides a proper setting for sequential decision making under uncertainty [162] and algorithms like LinUCB [2, 32, 38] play a central role in designing PROOF.

Also related is offline policy learning [12, 41, 149]. Compared to CB, it does not need any online trials, and hence is much easier to convince the stakeholder to adopt. However, it comes with the assumption that the historical data has many different actions attempted, which often fails to hold in the AI4SG problems. Furthermore, most of this literature focus on the binary action setting and it, similar to CB, does not explicitly use the feature/label dataset.

8.3 Bandit Data-driven Optimization

We describe the formal setup of bandit data-driven optimization in Procedure 7. On Line 1, we receive an initial dataset $\mathcal{D}$ of size n_0, with features x_i^0 and label c_i^0 for data point i, and

Table 8.1: A comparison of different models with respect to the desired properties in AI4SG applications.

Desired properties in AI4SG applications	**Bandit data-driven optimization**	Data-driven optimization	Contextual bandit	Offline reinforcement learning
No diverse past data needed	**Yes**	No	Yes	No
Explicit learning and optimization	**Yes**	Yes	No	No
No assumption on policy objective	**Yes**	No	Yes (but ignores domain knowledge)	Yes (but ignores domain knowledge)
Allows for iterative process	**Yes**	No	Yes	Yes
Finds optimal policy quickly	**Yes (compared to bandit)**	Yes (if diverse data available)	No	Yes (if diverse data available)

intervention in-place w_i^0 when the data point is collected. Each feature vector x_i^0 is drawn i.i.d. from an unknown distribution D_x. Each label $c_i^0 \in C$ is independently drawn from an unknown conditional distribution $D(w_i^0)_{c|x_i^0}$, which is parameterized by the intervention w_i^0, as different interventions could lead to different data distributions. In reality, w_i^0 is often identical across all i. On Line 3, we use all the data collected so far to train an ML model f_t, which is a mapping from features X to labels C. On Line 4, we get a new set of feature samples $\mathbf{x}^t = \{x_i^t\}_{i=1}^n$. Then, we select an intervention $w_i^t \in W$ for each individual i. On Line 5, we commit to interventions $\mathbf{w}^t = \{w_i^t\}$ and receive the labels $\mathbf{c}^t = \{c_i^t\}$. Each label is independently drawn from the distribution $D(w_i^t)_{c|x_i^t}$. Subsequently, on Line 6, we incur a cost u_t.

We assume that the cost u_t is determined by a partially known function $u(\mathbf{c}^t, \mathbf{w}^t)$. The function consists of three terms. The first term $\sum_i p(c_i^t, w_i^t)$ is the known loss. $p(c, w)$ is a fully known function capturing the loss for choosing intervention w and getting label c. It represents our modeling effort and domain knowledge. The second term $\sum_i q(w_i^t)$ is the unknown loss. $q(w)$ is an unknown function representing all the unmodeled objectives and the unintended consequences of using the intervention w. The third term is random noise η. This form of loss – a known part $p(\cdot)$ and an unknown part $q(\cdot)$ – is a realistic compromise of two extremes. AI4SG researchers spend a lot of time communicating with domain practitioners to understand the problem. It would go against this honest effort to eliminate $p(\cdot)$ and model the process as a pure bandit problem. On the other hand, there will be unmodeled objectives, however hard we try. It would be too arrogant to eliminate $q(\cdot)$ and pretend that anything not going according to the plan is noise. The unknown $q(\cdot)$ is our acknowledgement that any intervention recommended by AI4SG projects may have unintended consequences. We leave to future work to consider other interactions between $p(\cdot)$ and $q(\cdot)$.

Procedure 7: Bandit Data-driven Optimization

```
1 Receive initial dataset $\mathcal{D} = \{(x_i^0, c_i^0; w_i^0)_{i=1,\ldots,n_0}\}$ from distribution $D$ on $(X, C)$.
2 for $t = 1, 2, \ldots, T$ do
3     Using all the available data $\mathcal{D}$, train ML prediction model $f_t : X \longrightarrow C$.
4     Given $n$ feature samples $\{x_i^t\} \sim D_x$, choose interventions $\{w_i^t\}$ for each individual $i$.
5     Receive $n$ labels $\{c_i^t\} \sim D(w_i^t)_{c|x_i^t}$. Add $\{(x_i^t, c_i^t; w_i^t)_{i=1,\ldots,n}\}$ to the dataset $\mathcal{D}$.
6     Get cost $u_t = u(\mathbf{c}^t, \mathbf{w}^t) = \sum_i p(c_i^t, w_i^t) + \sum_i q(w_i^t) + \eta$, where $\eta \sim N(0, \sigma^2)$.
```

Given this procedure, the question is how to select the intervention $\mathbf{w}^t$. As is typical in the bandit literature, we define the optimal policy to be that given feature $\mathbf{x}$, pick action $\pi(\mathbf{x})$ such that

$$\pi(\mathbf{x}) = \arg\min_{\mathbf{w}} \mathbb{E}_{\mathbf{c},\eta|\mathbf{x}}[u(\mathbf{c}, \mathbf{w})],$$

where the expectation is taken over labels $\mathbf{c}$ and noise η conditioned on the features $\mathbf{x}$. The goal is to devise an algorithm to select interventions $\mathbf{w}^t$ to minimize the regret

$$R_T = \mathbb{E}_{x,c,\eta}\left[\sum_{t=1}^{T}\left(u(\mathbf{c}^t, \mathbf{w}^t) - u(\mathbf{c}^t, \pi(\mathbf{x}^t))\right)\right].$$

The label c can be a scalar or a vector. For the rest of the chapter, we assume $C \in \mathbb{R}^d$ and $W \in \mathbb{R}^d$. W may be discrete or continuous but it is assumed to be bounded.

Food rescue volunteer recommendation as bandit data-driven optimization We use the food rescue volunteer engagement problem introduced in Chapter 5 to illustrate how bandit data-driven optimization captures real-world AI4SG workflows. The following description is based on the recommender system in Section 5.5. A food rescue (FR) organization receives food donations from restaurants and grocery stores and connects them to low-resource community organizations. FR dispatchers would post the donor and recipient information on their mobile app, and some volunteer would claim the rescue and pick up and deliver the donations. This creates a lot of uncertainty because some rescue trips will get no volunteer to claim it. To prevent this, the dispatcher may recommend each rescue to a subset of volunteers through push notifications, The selection of volunteers to notify is the intervention $w \in \{0, 1\}^d$ (with the j^{th} dimension representing whether to send notification to the j^{th} volunteer). This decision is dependent on how likely a rescue will be claimed by each volunteer. Thus, we develop a ML-based recommender system which uses the features of a rescue and the volunteers, e.g. donor/recipient location, weather, the volunteer's historical activities, etc. (feature x), to predict the probabilities that each volunteer will claim the rescue. After we select w for a rescue, we observe which volunteer actually claim the rescue, that is, the label c (with the j^{th} dimension representing whether the j^{th} volunteer claims the rescue). This data point will be added to our dataset and used for training in the future. The optimization objective $p(c, w)$ is that we want to send notifications to the volunteers who will claim it, while still guaranteeing not a lot of push notifications are sent. Obviously, whether or not the rescue gets claimed after these push notifications matter to the FR. Yet, there is more to the cost to the FR, e.g. how each volunteer reacts to push notifications (will they get annoyed and drop out?). The $q(\cdot)$ cost could capture such factors.

In the US alone, there are already over 50 cities where FRs are providing basic necessities and affecting over a million people. We have been working with a food rescue organization for years. In Section 8.5.2, we include a case study of bandit data-driven optimization in the food rescue setting.

8.4 Algorithms and Regret Analysis

We propose a flexible algorithm for bandit data-driven optimization and establish a formal regret analysis of the algorithm under the specific settings as follows.

The data points are drawn from $X \times C \subseteq \mathbb{R}^m \times \mathbb{R}^d$. We assume all $x \in X$ has l^2-norm bounded by constant K_X, and the label space C has l^1-diameter K_C. The action space W could be either discrete or continuous, but is bounded inside the unit l^2-ball in $\mathbb{R}^d$. We specify the data distribution by an arbitrary marginal distribution D_x on X and a conditional distribution such that $c = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, for some unknown function f. To begin with, we assume $f \in \mathcal{F}$ comes from the class of all linear functions with $f(x) = Fx$, and we use ordinary least squares regression as the learning algorithm. We will relax this assumption towards the end of Section 8.4.2. The known cost $p(c, w) = c^\dagger w$ is the inner product of label c and action w.[1] The unknown cost is $q(w) = \mu^\dagger w$, where μ is an unknown but fixed vector. Furthermore, for exposition purpose we will start by assuming that the intervention w does not affect the data distribution. In Section 8.4.3, we will remove this assumption and present the algorithm for the general case.

8.4.1 With Exactly Known Objectives

As a primer to our main results to be introduced in the following section, we first look into a special case where we know the optimization objective exactly. That is, our cost only consists of $p(\cdot)$, with $q(\cdot) = 0$. This is not a very realistic setting, but by analyzing it we will get some intuition into the general case.

At each iteration, this setting resembles the predict-then-optimize framework studied by Elmachtoub and Grigas [44]. Given a sample feature x, we need to solve the linear program with a known feasible region $W \subseteq \mathbb{R}^d$:

$$\begin{aligned} \min_{w} \quad & \mathbb{E}_{c \sim D_{c|x}}[p(c, w)|x] = \mathbb{E}_{c \sim D_{c|x}}[c|x]^\dagger w \\ s.t. \quad & w \in W \end{aligned}$$

We hope to learn a predictor $\hat{f} : X \longrightarrow C$ from the given dataset, so that we can solve the following problem instead.

$$w^*(\hat{c}) := \arg\min_{w} \quad \hat{c}^\dagger w \qquad \text{where} \quad \hat{c} = \hat{f}(x)$$
$$s.t. \quad w \in W$$

[1]To avoid confusion, in this chapter we use superscript $\dagger$ to denote matrix and vector transpose.

In this chapter we assume that the problem has a unique optimal solution. Since the total cost is the same as the known optimization objective, intuitively we should simply commit to the action $w^*(\hat{c})$. By doing so, the expected regret we incur on this data point is $\mathbb{E}_x[r(x)]$, where

$$r(x) = \mathbb{E}_{c|x}[c]^\dagger(w^*(\hat{c}) - w^*(\mathbb{E}_{c|x}[c])).$$

Theorem 8.4.1 establishes that, indeed, this strategy leads to no-regret. This is not entirely trivial, because the optimization is based on the learned predictor yet the cost is based on the true distribution. The proof of Theorem 8.4.1 is instrumental to the subsequent results.

Theorem 8.4.1. *When the total cost is fully modeled, i.e. $q(\cdot) = 0$, simply following the predict-then-optimize optimal solution leads to regret $O(\sqrt{ndmT})$.*

Proof of Theorem 8.4.1. Let $w_{i*}^t = \arg\min_w \mathbb{E}_{c_i^t|x_i^t}[c_i^t]^\dagger w$ and $w_i^t = \arg\min_w \hat{c}_i^{t\dagger} w$. The expected regret at round t on individual i is $\mathbb{E}[r_i^t]$, where

$$\begin{aligned} r_i^t &= \mathbb{E}\left[\mathbb{E}_{c_i^t|x_i^t}[c_i^t]^\dagger(w_i^t - w_{i*}^t)\right] \\ &\leq \mathbb{E}\left[(\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t)^\dagger(w_i^t - w_{i*}^t)\right] \\ &= O\left(\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right]\right) \end{aligned}$$

The first inequality above used the definition of w_i^t and w_{i*}^t. The second step used Cauchy-Schwartz. Note that what remains to prove is simply an error bound on the OLS regression, which we prove as Lemma 8.4.2. Using that result, we can conclude the total regret is

$$\begin{aligned} R_T &= \mathbb{E}[\sum_{t=1}^T \sum_{i=1}^n r_i^t] \\ &= O\left(\sum_{t=1}^T \sum_{i=1}^n \mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right]\right) \\ &= O\left(\sum_{t=1}^T \sum_{i=1}^n \sqrt{\frac{dm}{nt}}\right) \\ &= O\left(\sqrt{ndmT}\right) \end{aligned}$$

The last step (bounding $\sum_{t=1}^T t^{-1/2}$) is by an upper bound on the generalized harmonic numbers, which can be found in Theorem 3.2 (b) in the text by Apostol [11]. □

Recall that $\mathcal{F}$ is the class of all linear functions mapping X to C and $c = Fx + \epsilon$ where $F \in \mathcal{F}$ and $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$. Assume that $n > m$, that is, assume the number of data points we receive each round is greater than the number of features. Let F_k be the k-th row of F. Fix k, we have a linear regression problem $c_k = F_k^\dagger x + \epsilon_k$, where $\epsilon_k \sim \mathcal{N}(0, \sigma^2)$. At the t-th round, we have nt data points and we need to predict on n new data points. Let X^t be the $n \times m$ matrix whose i-th row is x_i^t. Let $\tilde{X}^t$ be the $nt \times m$ matrix consisting of all the training data points. Suppose we fun an ordinary least wquares regression. Let $\hat{F}_k$ be the OLS estimate of F_k, and $\hat{c}_k = \hat{F}_k x$.

Lemma 8.4.2. *Suppose we use the ordinary least squares regression as the ML algorithm. The prediction error is*

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] = O\left(\sqrt{\frac{dm}{nt}}\right),$$

assuming either (1) $x \sim \mathcal{N}(0,\Lambda)$ follows a normal distribution, or (2) the eigenvalues of $\Sigma = \frac{\tilde{X}^{t\dagger}\tilde{X}^t}{nt}$ are lower bounded by a positive number.

Proof. Consider the first case, since $x \sim \mathcal{N}(0,\Lambda)$, we know $X^{T^\dagger}X^t \sim W(\Lambda, n)$, a Wishart distribution with n degrees of freedom, and $\tilde{X}^{T^\dagger}\tilde{X}^t \sim W(\Lambda^{-1}, nt)$, an inverse Wishart distribution with nt degrees of freedom. Thus, $\mathbb{E}_X[(X^{t^\dagger}X^t)^{-1}] = n\Lambda$ and $\mathbb{E}_X[(\tilde{X}^{t^\dagger}\tilde{X}^t)^{-1}] = \Lambda^{-1}/(nt-m-1)$.

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{n}(\mathbb{E}_{c_{ik}^t|x_i^t}[c_{ik}^t] - \hat{c}_{ik}^t)^2\right] &= \mathbb{E}\left[||X^t(F_k - \hat{F}_k)||_2^2\right] \\
&= \mathbb{E}\left[(F_k - \hat{F}_k)^\dagger X^{t^\dagger} X^t (F_k - \hat{F}_k)\right] \\
&= \mathbb{E}\left[tr((F_k - \hat{F}_k)^\dagger X^{t^\dagger} X^t (F_k - \hat{F}_k))\right] \\
&= tr\left(\mathbb{E}\left[X^{t^\dagger}X^t\right]\mathbb{E}_X\left[(F_k - \hat{F}_k)(F_k - \hat{F}_k)^\dagger\right]\right) \\
&= \sigma^2 tr(\mathbb{E}\left[X^{t^\dagger}X^t\right]\mathbb{E}_X\left[(\tilde{X}^{t^\dagger}\tilde{X}^t)^{-1}\right]) \\
&= \sigma^2 tr\left(\frac{n\Lambda\Lambda^{-1}}{nt-m-1}\right) = \frac{nm\sigma^2}{nt-m-1}
\end{aligned}$$

The above derivation has appeared in previous literature, e.g. the work by Rosset and Tibshirani [130]. The result holds for all k, we get

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2^2\right] = \frac{md\sigma^2}{nt-m-1}.$$

That is,

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] = O\left(\sqrt{\frac{dm}{nt}}\right).$$

In the second case, suppose the eigenvalues of $\Sigma = \frac{\tilde{X}^{t^\dagger}\tilde{X}^t}{nt}$ are lower bounded by a constant $K_\Sigma > 0$.

$$\begin{aligned}
\mathbb{E}_{X,\epsilon}\left[\left|\mathbb{E}_{c_{ik}^t|x_i^t}[c_{ik}^t] - c_{ik}^t\right|\right] &= \mathbb{E}_{X,\epsilon}\left[\left|F_k^\dagger x_{ik}^t - \hat{F}_k^\dagger x_{ik}^t\right|\right] \\
&\leq \mathbb{E}_X\left[\left|F_k^\dagger x_{ik}^t - \hat{F}_k^\dagger x_{ik}^t\right|\right] \leq \mathbb{E}_X\left[\left\|F_k - \hat{F}_k\right\|_2 \|x_{ik}^t\|_2\right] \\
&\leq K_X\mathbb{E}_X\left[\left\|F_k - \hat{F}_k\right\|_2^2\right]^{1/2} = K_X\mathbb{E}_X\left[tr(\sigma^2(\tilde{X}^{t^\dagger}\tilde{X}^t)^{-1})\right]^{1/2} \\
&= \frac{\sigma K_X}{\sqrt{nt}}\mathbb{E}_X\left[tr\left(\Sigma^{-1}\right)\right]^{1/2}
\end{aligned}$$

Algorithm 8: PROOF: Predict-then-optimize with optimism in face of uncertainty

```
1  Initialize:
2      Find a barycentric spanner $b_1, \dots, b_d$ for $W$
3      Set $A_i^1 = \sum_{j=1}^d b_j b_j^\dagger$ and $\hat{\mu}_i^1 = 0$ for $i = 1, 2, \dots, n$.
4  Receive initial dataset $\mathcal{D} = \{(x_i^0, c_i^0; w_i^0)_{i=1,\dots,n_0}\}$ from distribution $D$ on $(X, C)$.
5  for $t = 1, 2, \dots, T$ do
6      Using all data in $\mathcal{D}$, train ML model $f_t : X \longrightarrow C$.
7      Given $n$ feature samples $\{x_i^t\} \sim D_x$, get predictions $\hat{c}_i^t = f_t(x_i^t)$.
8      Set $\beta^t = \max\left(128 d \log t \log \frac{nt^2}{\gamma}, \left(\frac{8}{3} \log \frac{nt^2}{\gamma}\right)^2\right)$
9      for $i = 1, 2, \dots, n$ do
10         Set Confidence ball $B_i^t = \{\nu : \|\nu - \hat{\mu}_i^t\|_{2,A_i^t} \le \sqrt{\beta^t}\}$.
11         Solve optimization problem $w_i^t = \arg\min_{w \in W} \min_{\nu \in B_i^t} (\hat{c}_i^t + \nu)^\dagger w$. Choose
             intervention $w_i^t$.
12         Receive label $c_i^t \sim D_{c|x_i^t}$. Add $(x_i^t, c_i^t; w_i^t)$ to $\mathcal{D}$.
13         Get cost $u_i^t = u(x_i^t, c_i^t, w_i^t) = (c_i^t)^\dagger w_i^t + \mu^\dagger w_i^t + \eta_i$, where $\eta_i \sim N(0, \sigma^2)$. In particular,
             let $u_{oi}^t$ be the first term and let $u_{bi}^t$ be the sum of the second and third term.
14         Update $A_i^{t+1} = A_i^t + w_i^t (w_i^t)^\dagger$
15         Update $\hat{\mu}_i^{t+1} = (A_i^{t+1})^{-1} \sum_{\tau=1}^t u_{bi}^t w_i^t$
```

Then the prediction error can be bounded by

$$\mathbb{E}_{X,\epsilon}\left[\left|\mathbb{E}_{c_{ik}^t|x_i^t}[c_{ik}^t] - \hat{c}_{ik}^t\right|\right] \le O\left(\sqrt{\frac{m}{nt}}\right)$$

This holds for all k. Thus, we have

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] \le O\left(\sqrt{\frac{md}{nt}}\right)$$

□

8.4.2 PROOF: Predict-then-Optimize with Optimism in Face of Uncertainty

When there is no bandit uncertainty, as we showed just now one can simply follow the predict-then-optimize framework and no-regret is guaranteed. However, the unknown bandit cost is crucial to real-world AI4SG applications. We now describe the first algorithm for bandit data-driven optimization, PRedict-then-Optimize with Optimism in Face of uncertainty (PROOF), shown in Algorithm 8.

PROOF is an integration of the celebrated Optimism in Face of Uncertainty (OFU) framework and the predict-then-optimize framework. It is clear that the unknown cost component $q(\cdot) + \eta$ forms a linear bandit. For this bandit component, we run an OFU algorithm for each individual i with the same unknown loss vector μ. The OFU component for each individual i maintains a confidence ball B_i^t which is independent of the predict-optimize framework. The predict-then-optimize framework produces an estimated optimization objective $\hat{c}^t$ independent of OFU. The two components are integrated together on Line 11 of Algorithm 8, where we compute the intervention for the current round taking into consideration the essence of both frameworks.

Below, we justify why this algorithm achieves no-regret. First, we state a theorem by Dani et al. [38], which states that the confidence ball captures the true loss vector μ with high probability. The result was proved for the original OFU algorithm. However, since the result itself does not depend on the way we choose w^t, it still holds in bandit data-driven optimization. We adapt it by adding a union bound so that the result holds for all the n bandits simultaneously.

Lemma 8.4.3 (Adapted from Theorem 5 by Dani et al. [38])**.** *Let $\gamma > 0$, then $\mathbb{P}(\forall t, \forall i, \mu \in B_i^t) \geq 1 - \gamma$.*

The following key lemma decomposes the regret of PROOF into two components: one involving the online bandit loss, the other concerning the offline supervised learning loss.

Lemma 8.4.4. *With probability $1 - \delta$, Algorithm 8 has regret*

$$O\left(\sqrt{8mT\beta^T \log T} + \sum_{t=1}^{T}\sum_{i=1}^{n} \mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right]\right).$$

Proof of Lemma 8.4.4. Let $w_{i*}^t = \arg\min_w (\mathbb{E}_{c_i^t|x_i^t}[c_i^t]+\mu)^\dagger w$. w_{i*}^t is the optimal action for individual i at time t, and is the benchmark in our regret computation.

Fix i, fix t. Let $\tilde{v} = \arg\min_{v\in B_i^t}(\hat{c}_i^t + v)^\dagger w_i^t$. Because of Line 11, we have

$$\begin{aligned}(\hat{c}_i^t + \tilde{v})^\dagger w_i^t &= \min_{v\in B_i^t, w\in W} (\hat{c}_i^t + v)^\dagger w \\ &\leq (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger w_{i*}^t + (\hat{c}_i^t)^\dagger w_{i*}^t - \mathbb{E}_{c_i^t|x_i^t}[c_i^t]^\dagger w_{i*}^t.\end{aligned}$$

The inequality above used the fact that $\mu \in B_i^t$, by Lemma 8.4.3. Thus, we get the per-round regret

$$\begin{aligned}&(\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger (w_i^t - w_{i*}^t) \\ &\leq (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^\dagger w_i^t - (\hat{c}_i^t + \tilde{v})^\dagger w_i^t + (\hat{c}_i^t)^\dagger w_{i*}^t \\ &- \mathbb{E}_{c_i^t|x_i^t}[c_i^t]^\dagger w_{i*}^t \\ &= (\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t)^\dagger (w_i^t - w_{i*}^t) + (\mu - \tilde{v})^\dagger w_i^t\end{aligned}$$

We can view the second term is the per-round regret for the bandit part. By Theorem 6 in [38], we have

$$\sum_{t=1}^{T}((\mu - \tilde{v})^\dagger w_i^t)^2 \leq 8m\beta^T \log T$$

Using the Cauchy-Schwarz, we get

$$\sum_{t=1}^{T}(\mu - \tilde{\nu})^{\dagger} w_i^t \leq \sqrt{8mT\beta^T \log T}$$

Thus, the regret of Algorithm 8 is

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{n}(\mathbb{E}_{c_i^t|x_i^t}[c_i^t] + \mu)^{\dagger}(w_i^t - w_{i*}^t)\right] \\
&\leq \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{n}(\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t)^{\dagger}(w_i^t - w_{i*}^t)\right] \\
&+ n\sqrt{8mT\beta_T \log T} \\
&= O\left(\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] + n\sqrt{8mT\beta_T \log T}\right)
\end{aligned}$$

The last step used Cauchy-Schwartz and the bounded action space assumption. □

Clearly, to characterize the regret, we need to bound $\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right]$. In the case of linear regression, we have the following theorem.

Theorem 8.4.5. *Assuming we use ordinary least squares regression as the ML algorithm, Algorithm 8 has regret* $\tilde{O}\left(n\sqrt{dmT}\right)$ *with probability* $1 - \delta$.

Proof of Theorem 8.4.5. Combine Lemma 8.4.4 and Lemma 8.4.2. □

Theorem 8.4.5 assumes a linear regression problem with a specific learning algorithm – ordinary least squares linear regression. If our intent is for Algorithm 8 to be modular where one can use any learning algorithm, we could resort to sample complexity bounds. In Appendix C.1, we include a derivation of the regret bound from the sample complexity perspective. This approach allows us to extend the result in Theorem 8.4.5 to a more general setting.

8.4.3 When Interventions Affect the Label Distribution

So far in this section, we have had the assumption that the action w does not affect the distribution D from which as sample (X, C). In many real-world scenarios this is not the case. For example, if the wildlife patrollers change their patrol routes, the poachers' poaching location would change accordingly and hence its distribution would be very different. Thus, it is valuable to study this more general setting where the intervention could affect the label distribution.

First, let us make the assumption that there are finitely many possible actions. We will consider the continuous action space later. Since there are finitely many actions, an intuitive idea is to train an ML predictor for each action separately. Because we do not impose any assumption on our initial dataset, which might only have a single action embedded, we clearly need to use exploration in the bandit algorithm and use the data points gathered along the way to train the predictor. It might seem very natural to fit this directly into the framework of

PROOF as shown in Algorithm 8: simply maintain several predictors instead of one, and still choose the best action on Line 11. However, to train the predictor corresponding to each action, we need at least a certain number of data points to bound the prediction error. Yet, PROOF, and UCB-type algorithms in general, do not give a lower bound on how many times each action is tried. For example, Algorithm 8 might never try some action at all, and we would not be able to train a predictor for that action. To resolve this philosophical contradiction, we add a uniform exploration phase of length $\tilde{T}$ at the beginning, where at each round $1, 2, \ldots, \tilde{T}$, each action is taken on some examples. Other than this, we inherit all the setup for the analysis in Section 8.4.2. We describe the detailed procedure as Algorithm 9.

We establish the following lemma which decomposes the regret into 3 parts: regret during uniform exploration, regret in UCB bandit, and regret through supervised learning.

Lemma 8.4.6. *With probability* $1 - \delta$, *Algorithm 9 has regret*

$$O\Bigg(n\tilde{T} + n\sqrt{8mT\beta_T \log T} \\ + \sum_{t=\tilde{T}+1}^{T} \sum_{i=1}^{n} \mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t)\right\|_2\right]\Bigg).$$

Proof of Lemma 8.4.6. In Algorithm 9, at each round $1, 2, \ldots, \tilde{T}$ in the exploration phase, each action is sequentially taken on some examples. This means by the end of step $\tilde{T}$, we have $\tilde{n} = n\tilde{T}/|W|$ data points for training the predictor for each w_i. Although in practice one can keep updating (learning) the predictor during the exploitation phase, in the following theoretical analysis it suffices to ignore this additional learning effect. We assume $\tilde{n}$ is an integer, but this is not an issue in the general case.

Let us first analyze the regret in the exploitation phase.

Let $w_{i*}^t = \arg\min_w(\mathbb{E}_{c_i^t|x_i^t,w}[c_i^t(w)] + \mu)^\dagger w$. w_{i*}^t is the optimal action for individual i at time t, and is the benchmark in our regret computation.

Fix i, fix t. Let $\tilde{\nu} = \arg\min_{\nu\in B_i^t}(\hat{c}_i^t(w_i^t) + \nu)^\dagger w_i^t$. Because of Line 18, we have

$$\begin{aligned}(\hat{c}_i^t(w_i^t) + \tilde{\nu})^\dagger w_i^t &= \min_{\nu\in B_i^t, w\in W} (\hat{c}_i^t(w) + \nu)^\dagger w \\ &\le (\mathbb{E}_{c_i^t|x_i^t,w_{i*}^t}[c_i^t(w_{i*}^t)] + \mu)^\dagger w_{i*}^t + (\hat{c}_i^t(w_{i*}^t))^\dagger w_{i*}^t \\ &- \mathbb{E}_{c_i^t|x_i^t,w_{i*}^t}[c_i^t(w_{i*}^t)]^\dagger w_{i*}^t.\end{aligned}$$

The inequality above used the fact that $\mu \in B_i^t$, by Lemma 8.4.3. Thus, we get the per-round regret

$$\begin{aligned}&(\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] + \mu)^\dagger w_i^t - (\mathbb{E}_{c_i^t|x_i^t,w_{i*}^t}[c_i^t(w_{i*}^t)] + \mu)^\dagger w_{i*}^t \\ &\le (\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] + \mu)^\dagger w_i^t - (\hat{c}_i^t(w_i^t) + \tilde{\nu})^\dagger w_i^t \\ &+ (\hat{c}_i^t(w_{i*}^t))^\dagger w_{i*}^t - \mathbb{E}_{c_i^t|x_i^t,w_{i*}^t}[c_i^t(w_{i*}^t)]^\dagger w_{i*}^t \\ &= (\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t))^\dagger w_i^t \\ &+ (\hat{c}_i^t(w_{i*}^t) - \mathbb{E}_{c_i^t|x_i^t,w_{i*}^t}[c_i^t(w_{i*}^t)])^\dagger w_{i*}^t + (\mu - \tilde{\nu})^\dagger w_i^t\end{aligned}$$

Algorithm 9: PROOF WITH ACTION-SPECIFIC LABEL DISTRIBUTION

```
1  Initialize:
2      Find a barycentric spanner $b_1, \dots, b_n$ for $W$
3      Set $A_i^1 = \sum_{j=1}^d b_j b_j^\dagger$ and $\hat{\mu}_i^1 = 0$ for all $i = 1, 2, \dots, n$
4  Receive initial dataset $\mathcal{D} = \{(x_i^0, c_i^0; w_i^0)_{i=1,\dots,n}\}$ from distribution $D$ on $(X, C)$.
   // Uniform exploration phase
5  for $t = 1, 2, \dots, \tilde{T}$ do
6      for $i = 1, 2, \dots, n$ do
7          Given feature sample $x_i^t$, choose intervention $w_i^t = w_{nt+i \mod |W|}$ where
             $W = \{w_1, \dots, w_{|W|}\}$ is considered as an ordered set.
8          Receive label $c_i^t \sim D(w_i^t)_{c|x_i^t}$. Add $(x_i^t, c_i^t; w_i^t)$ to the dataset $\mathcal{D}$.
9          Get cost $u_i^t = u(x_i^t, c_i^t, w_i^t) = (c_i^t(w_i^t))^\dagger w_i^t + \mu^\dagger w_i^t + \eta_i$, where $\eta_i \sim N(0, \sigma^2)$. In
             particular, let $u_{oi}^t$ be the first term and let $u_{bi}^t$ be the sum of the second and
             third term.
10         Update $A_i^{t+1} = A_i^t + w_i^t (w_i^t)^\dagger$
11         Update $\hat{\mu}_i^{t+1} = (A_i^{t+1})^{-1} \sum_{\tau=1}^t u_{bi}^t w_i^t$
   // UCB exploitation phase
12 for $t = \tilde{T} + 1, \dots, T$ do
13     For each $w$, using all the available data $\mathcal{D}$ that were collected under $w$, train ML
         prediction model $f_w^t : X \longrightarrow C$.
14     Given $n$ feature samples $\{x_i^t\} \sim D_x$, get predictions $\hat{c}_i^t(w) = f_t(x_i^t)$, for each $w$.
15     Set CB radius $\beta^t = \max\left(128d \log t \log(nt^2/\gamma), \left(\frac{8}{3} \log\left(\frac{nt^2}{\gamma}\right)\right)^2\right)$
16     for $i = 1, 2, \dots, n$ do
17         Set confidence ball $B_i^t = \{\nu : \|\nu - \hat{\mu}_i^t\|_{2, A_i^t} \leq \sqrt{\beta^t}\}$.
18         Solve optimization problem $w_i^t = \arg\min_{w \in W} \min_{\nu \in B_i^t} (\hat{c}_i^t(w) + \nu)^\dagger w$. Choose
             intervention $w_i^t$.
19         Receive label $c_i^t \sim D(w_i^t)_{c|x_i^t}$. Add $(x_i^t, c_i^t; w_i^t)$ to the dataset $\mathcal{D}$.
20         Get cost $u_i^t = u(x_i^t, c_i^t, w_i^t) = (c_i^t(w_i^t))^\dagger w_i^t + \mu^\dagger w_i^t + \eta_i$, where $\eta_i \sim N(0, \sigma^2)$. In
             particular, let $u_{oi}^t$ be the first term and let $u_{bi}^t$ be the sum of the second and
             third term.
21         Update $A_i^{t+1} = A_i^t + w_i^t (w_i^t)^\dagger$
22         Update $\hat{\mu}_i^{t+1} = (A_i^{t+1})^{-1} \sum_{\tau=1}^t u_{bi}^t w_i^t$
```

We can view the third term as the per-round regret for the bandit part. By Theorem 6 in Dani et al. [38], we have

$$\sum_{t=1}^{T}((\mu - \tilde{\nu})^\dagger w_i^t)^2 \leq 8m\beta^T \log T$$

Using the Cauchy-Schwarz, we get

$$\sum_{t=1}^{T}(\mu - \tilde{\nu})^\dagger w_i^t \leq \sqrt{8mT\beta^T \log T}$$

Thus, the regret of Algorithm 9 is upper bounded by

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{n}(\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] + \mu)^\dagger w_i^t - (\mathbb{E}_{c_i^t|x_i^t,w_{i*}^t}[c_i^t(w_{i*}^t)] + \mu)^\dagger w_{i*}^t\right] \\
&\leq Kn\tilde{T} + \mathbb{E}\left[\sum_{t=\tilde{T}+1}^{T}\sum_{i=1}^{n}(\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t))^\dagger w_i^t\right. \\
&\left.+ (\hat{c}_i^t(w_{i*}^t) - \mathbb{E}_{c_i^t|x_i^t,w_{i*}^t}[c_i^t(w_{i*}^t)])^\dagger w_{i*}^t\right] + n\sqrt{8mT\beta_T \log T} \\
&= O\left(n\tilde{T} + n\sqrt{8mT\beta_T \log T}\right. \\
&\left.+ \sum_{t=\tilde{T}+1}^{T}\sum_{i=1}^{n}\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t)\right\|_2\right]\right)
\end{aligned}$$

The last step used Cauchy-Schwartz, symmetry, and the bounded action space assumption. □

By combining Lemma 8.4.6 with previous results, we arrive at the regret of PROOF in this more general setting.

Theorem 8.4.7. *With finitely many actions and OLS as the ML algorithm, Algorithm 9 has regret* $\tilde{O}\left(n(d|W|)^{1/3}m^{1/2}T^{2/3}\right)$.

Proof of Theorem 8.4.7. Again, we prove by bounding the linear regret prediction error. The proof follows identically as Theorem 8.4.5. We get, $\forall t > \tilde{T}, \forall w, \forall i$,

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t)\right\|_2\right] = O\left(\sqrt{\frac{dm|W|}{n\tilde{T}}}\right).$$

Using Lemma 8.4.6, and taking $\tilde{T} = T^{2/3}(d|W|)^{1/3}$, we get

$$\begin{aligned}
&O\left(n\tilde{T} + n\sqrt{8mT\beta_T \log T} + \sum_{t=\tilde{T}+1}^{T}\sum_{i=1}^{n}\sqrt{\frac{dm|W|}{n\tilde{T}}}\right) \\
&= O\left(n\tilde{T} + n\sqrt{8mT\beta_T \log T} + T\sqrt{\frac{ndm|W|}{\tilde{T}}}\right) \\
&= \tilde{O}\left((d|W|)^{1/3}m^{1/2}nT^{2/3}\right)
\end{aligned}$$

□

We now move on to the scenario where the action space W is continuous. In this case, we assume the true label of feature x under action w is $c = Fx + Gw + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. A small modification of Algorithm 9 will work in this scenario: instead of rotating over each action in the uniform exploration phase, we simply pick action w uniformly at random for each individual. Then, the regret of the algorithm is as follows.

Theorem 8.4.8. *Suppose the action space is continuous and the label can be modeled as a linear function of the feature and action. Assuming OLS as the ML algorithm, Algorithm 9 has regret* $\tilde{O}\left(m^{1/3} d^{2/3} n T^{2/3}\right)$.

Proof of Theorem 8.4.8. Using Lemma 8.4.2, we know that at the beginning of the exploitation phase, the prediction error is

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t)\right\|_2\right] = O\left(\sqrt{\frac{(m+d)d}{n\tilde{T}}}\right), \qquad \forall t > \tilde{T}, \forall w, \forall i.$$

Thus, using Lemma 8.4.6, we know the regret is $\tilde{O}\left(m^{1/3} d^{2/3} n T^{2/3}\right)$, when we take $\tilde{T} = m^{1/3} d^{2/3} T^{2/3}$. □

8.4.4 PROOF Is a Modular Algorithm

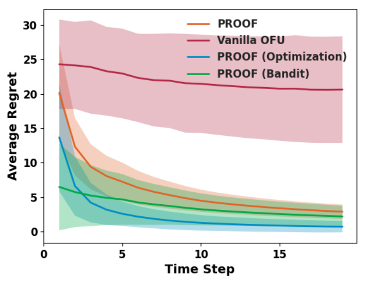
In practice, PROOF can be applied beyond the setting under which we proved the previous results. Rather than a fixed algorithm, it is designed to be modular so that we can plug in different learning algorithms and optimization problems. First, instead of a linear hypothesis class with linear regression algorithms, PROOF can accommodate any predictive model such as tree-based methods and deep neural networks. Second, The nominal optimization problem need not be a linear optimization problem. The optimization problem may be continuous or discrete, convex or non-convex, as we do not concern ourselves with computational complexity in this chapter. In Section 8.5.2, we demonstrate that even when we insert complex algorithms into the PROOF framework, thereby going beyond the setting where we established formal regret guarantees, PROOF still works well.

8.5 Experiment Results

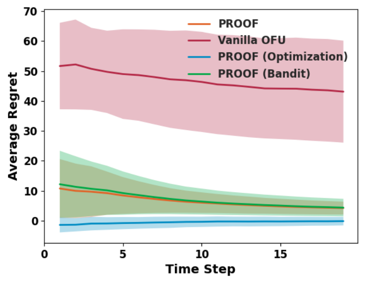
In this section, we evaluate PROOF on both simulated dataset and real dataset. The evaluation metric is the average regret, which is the total regret R_T as introduced in Section 8.3 divided by number of time steps T. We compare PROOF with a vanilla bandit baseline.

8.5.1 Numerical Simulations

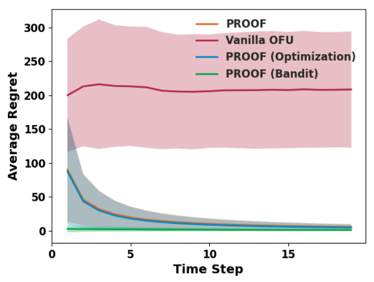
We implement PROOF in the setting described in Section 8.4.2 and show its performance on a simulated dataset.



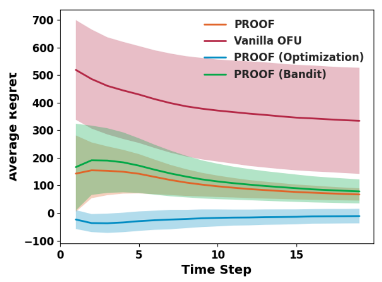
(a) Small scale base case



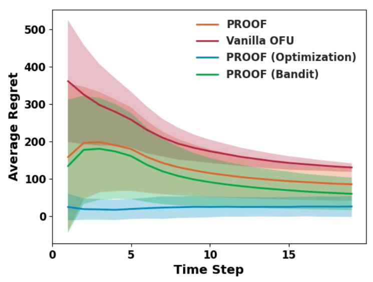
(b) Data per step increased from 20 to 40



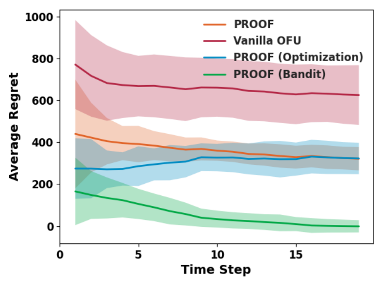
(c) Linear mapping norm multiplied by 10.



(d) Large scale base case.



(e) Linear mapping norm divided by 10.

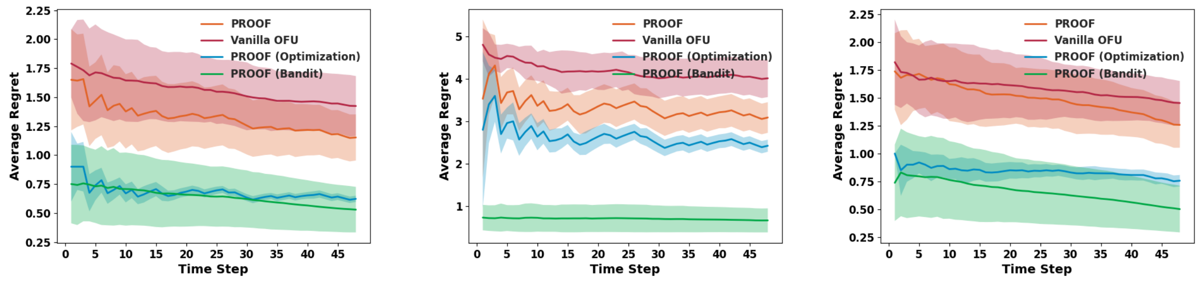


(f) Data noise multiplied by 5.

Figure 8.2: Numerical simulation results of PROOF compared against vanilla linear bandit. All results are averaged over 10 runs with shaded areas representing the standard deviation.

We start with a small-scale experiment. Recall that we train an ML predictor $\hat{f} : X \longrightarrow C$ where $X \subseteq \mathbb{R}^m$ and $C \subseteq \mathbb{R}^d$. We take feature dimension $m = 20$ and label dimension $d = 5$. At every round we get $n = 20$ data points. Following the tradition in the bandit literature, we assume the bandit reward is bounded in $[-1, 1]$ and the feasible region W is the unit l_2-ball, as it is the relative magnitude between the bandit and the optimization rewards that matters. For the true linear map F where $c = Fx + \epsilon$, we upper bound its l_1 matrix norm at 10. We sample the noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$ from a normal distribution where $\sigma^2 = 0.1$. We take the bandit noise $\eta \sim N(0, 10^{-4})$. We use ordinary least squares regression on all the data points collected so far for the learning part at each time step. Line 11 in Algorithm 8 has a non-convex bilinear program and we solve it with the non-convex solver IPOPT. To compute the regret we also need to find the best action given the true reward parameters. This is a convex program and we solve it with Gurobi. Algorithm 8 has an explicit expression for the confidence radius β^t. That is for establishing the regret bound. However, in experiments, the radius would be too large ($\sim 10^4$) for the algorithm to select meaningful actions early on even when the algorithm's reward estimate $\hat{\mu}$ is already quite accurate. We set $\beta^t = 1$ so that the algorithm can quickly concentrate on the correct region of interest. Setting $\beta^t = 10$ leads to similar performance.

The expected cost for a fixed action w is $\mathbb{E}_{c,\eta}[(c+\mu)^\dagger w + \eta] = \mathbb{E}[x^\dagger F^\dagger w] + \mu^\dagger w = \mu^\dagger w$, because when we generated x, the distribution has zero mean. This fits in the setting of [38] and thus this problem in theory might be solved simply as a linear bandit by feeding the total cost in bandit data-driven optimization to their OFU algorithm. Since the regret bound of vanilla OFU



(a) Base case (b) Known cost $p(\cdot)$ multiplied by 4 (c) Initial dataset size decreased to 20

Figure 8.3: The experiment results on the real-world food rescue data of PROOF compared against vanilla linear bandit. All results are averaged over 10 runs with shaded areas representing the standard deviation.

is the same as our PROOF in the order of T, this brings back the question that we have been repeating since the beginning of this chapter: if linear (contextual) bandit is a more general framework whose existing algorithms already solve our problem, why should we care about bandit data-driven optimization at all?

In Sections 8.1 through 8.3, we have answered this question with the nature of real-world AI4SG projects. Here, we can answer this question using numerical experiments. We show the average regret of PROOF as the orange curve in Figure 8.2, and that of OFU in red. We can decompose the average regret of PROOF into the regret of the optimization component and the regret of the bandit component. The former is simply the algorithm's optimization (known) cost minus the best intervention's optimization cost. The latter is defined similarly. Both of them need not be positive, but they sum up to the average regret of PROOF. In this way we show how PROOF makes progress on both ends.

Figure 8.2a shows that PROOF quickly reduces the average regret in both optimization and bandit components. On the other hand, the performance of vanilla OFU is much more underwhelming. A typical bandit regret bound ignores many constant factors. We think this performance difference is primarily due to the large variance in the implicit context x and c, which an offline predictive model captures much better. In fact, PROOF also has much smaller variance in its performance than vanilla OFU consistently.

We now tweak the problem parameters a bit and see how the performance changes. If we increase the number of data points per iteration from $n = 20$ to 40, Figure 8.2b shows that the regret of the optimization component becomes very small even at the beginning, because we have more data to learn from. If we increase the upper bound of the norm of the linear mapping matrix F from 10 to 100, Figure 8.2c shows that the optimization regret dominates the total regret. This is also reasonable as the optimization cost is now much larger than the bandit cost. Here, vanilla OFU suffers even more, because now its cost signal, the total cost for PROOF, has even larger magnitude and variance.

In the second set of experiments, we scale up the experiments and show that PROOF still performs better than OFU even when the problem parameters are not as friendly. Suppose we receive $n = 500$ data points at every time step, and each data point has $m = 50$ features. Keeping all other parameters unchanged, Fig. 8.2d shows that PROOF still outperforms OFU by a lot. Here, we seem to have enough data points for the prediction task, and thus the bandit regret dominates the total regret. In Fig. 8.2e, we change the norm of the linear mapping F from 10 to 1, making the optimization cost less important than the bandit cost. Indeed, this reduces the variance of OFU and thus it is doing much better than in previous experiments. However, our PROOF still outperforms OFU. Finally, in Fig. 8.2f, we increase the noise in the label distribution from $\epsilon \sim \mathcal{N}(0, 0.1I_d)$ to $\epsilon \sim \mathcal{N}(0, 0.5I_d)$. This poses more challenge to PROOF. As the data become more noisy, the optimization regret does not stay close to zero as in the previous two experiments. Nevertheless, PROOF still keeps its regret below OFU.

8.5.2 Food Rescue Volunteer Recommendation

Bandit data-driven optimization is motivated by the practical challenges in AI4SG projects. After abstracting these challenges to a theoretical model, we now return to the real world. We apply PROOF to one AI4SG project: food rescue volunteer recommendation. We have introduced the background of food rescue operations in Section 8.3.

For this experiment, we assume 100 volunteers. [2] At each time step, we get a new rescue and decide a subset of 10 volunteers to whom we send push notifications. We represent this action with a binary vector $w \in \{0,1\}^{100}$ such that $w_i = 1$ if volunteer i is notified and 0 otherwise. Thus, the feasible action space W is $\{0,1\}^{100}$ with the constraint of $\sum_{i=1}^{100} w_i \leq 10$.

The action w we take at each time step is backed by a content-based ML recommender system. The ML model receives a feature vector x which describes a particular rescue-volunteer pair, and outputs a label prediction $\hat{c}$ as the likelihood of this volunteer claiming this rescue.[3] This ML component has been studied in Chapter 5. The actual label c is a one-hot vector in $\{0,1\}^{100}$ indicating which volunteer actually claimed the rescue.

The known cost $p(c, w) = c^{\dagger} w$ is 1 if we notify a volunteer who eventually claimed a rescue and 0 otherwise. To minimize it, we could negate the label c (and its prediction $\hat{c}$). The bandit cost $q(\cdot)$ is the same as before. We solve the optimization at each time step of PROOF with Gurobi after applying a standard linearization trick [88]. We also gradually decrease the confidence radius β over time.

Unlike the case in Section 8.5.1, OFU algorithm does not work here in principle. This is because, working with real-world data, we do not know the data distribution and it is almost certainly not zero-mean. In fact, this experiment has also gone beyond the setting for which we proved formal regret bound for PROOF, yet we would like to see how these two algorithms perform in such a real-world use case.

We assume an initial dataset of 300 rescues and run the algorithms for 50 time steps, each time step corresponding to one new rescue. As shown in Figure 8.3a, PROOF outperforms

[2]The reader might notice that in Chapters 4 and 5 we had more volunteers in the dataset. However, in this chapter, the focus is on PROOF algorithm rather than food rescue recommendation itself. Thus, for each rescue, we take the volunteer who actually claimed it plus 99 other volunteers in this experiment.

[3]In Sections 8.3 and 8.4, the label and action are of the same dimension but here the output label is 1-dimensional while our action space is 100-dimensional. This is easy to resolve. Suppose each rescue-volunteer pair has m' features. While in practice we have an ML model $\hat{f}' : \mathbb{R}^{m'} \longrightarrow \mathbb{R}$ and pass 100 feature vectors to it serially (fixed rescue, iterating over volunteers), one could think of a product model $\hat{f} = \prod_{i=1}^{100} \hat{f}'$ which takes the concatenation of 100 feature vectors and outputs a 100-dimensional vector.

vanilla OFU by roughly 15%. The performance gain by PROOF can be contributed to its effective use of the available data, as the progress on bandit made by PROOF and vanilla OFU are quite similar. In Figure 8.3b, we scale up the known part of the cost by a factor of 4. Because the optimization is more emphasized, it is unsurprising to see that most of PROOF's progress depends on the recommender system itself. In this case, it has a larger performance margin over vanilla OFU. Finally, in Figure 8.3c we decrease the size of the initial dataset from 300 rescues to 20 rescues. We observe that PROOF still has an edge over vanilla OFU. The margin is minimal at the initial time steps, because we have much less initial information here. Yet still, as time goes by PROOF picks up more information in the feature/label dataset to expand its margin. In actual AI4SG projects, the amount of initial data is typically more than this, more resembling Figure 8.3a, but Figure 8.3c assures us that PROOF still works in this more extreme case.

8.6 Conclusion

Non-profit and public sectors have huge potential to benefit from the advancing machine learning research. However, plenty of experience shows that the machine learning model itself is almost always not enough to address the real-world societal challenges. Motivated by four practical pain points in such applications, we proposed bandit data-driven optimization, designed the PROOF algorithm, and showed that it has no-regret. Finally, we show its better performance over bandit algorithm in simulations and the food rescue context. We view bandit data-driven optimization as our first attempt to bridge the last-mile gap between static ML models and their actual deployment in the real-world non-profit and public context. Future work could explore its applications to other problem domains, as well as technical challenges such as confounded costs and high dimensional covariates.

Ethical implications It is always a tricky balance between careful scoping prior to deployment and taking it to the field to uncover hidden objectives and rewards. At either extreme of the spectrum the project would be unlikely to succeed. We proposed bandit data-driven optimization as an initial effort assuming a well-intentioned user who tries to strike this balance. The framework should not be understood by either end of the spectrum as advocating for the other end.

Chapter 9

Conclusion

In this thesis, we studied how to use learning and planning to address concrete real-world challenges in three application domains: cybersecurity, food waste and security, and environmental sustainability.

Our work on learning and planning in cybersecurity is rooted in game-theoretic deception approaches to counter potentially unknown attackers. The main contribution is a learning and planning pipeline with end-to-end optimality guarantee, where we first learn the preferences of an unknown attacker and then compute the optimal configuration to counter such attacker. While this model is relatively general, we also grounded the concept of deception in a specific type of cyber attacks called the watering-hole attacks. There, we implement deception in the network packets sent by the user's browser to mislead the attacker. Hence, besides the problem formulation, our contribution is an algorithm that can scalably compute the optimal deception strategy. We have also implemented this algorithm as part of a browser extension, which is publicly available online.

Our learning and planning work on food waste and security stems from a 4-year collaboration with a nonprofit organization that spans problem scoping, algorithmic research, randomized controlled trial, deployment, and lessons learned. As the first step, we adopted a predictive analysis and a data-driven optimization algorithm to compute the optimal generic intervention and notification scheme applicable to all rescues. This work was easy to implement and built the trust for further collaboration. Subsequently, our main contribution is a rescue-specific recommender system to send push notifications to the most likely volunteers for each given rescue. We leverage a mathematical programming based approach to diversify our recommendations, and propose an online algorithm to dynamically select the volunteers to notify without the knowledge of future rescues. Compared to our learning and planning work on cybersecurity, here we extend the planning to a sequential planning setting. The highlight of this work is the randomized controlled trial that we designed and ran for the recommender system. The trial showed that the algorithm significantly improved the claim rate and hit rate. The deployment of our algorithm inside a real-world production system is, in itself, an achievement.

While our work is grounded in the food rescue context, the techniques have general implications. Recommender systems are widely used in numerous commercial applications. While diversifying recommendations has been studied in the previous literature, it was only studied in a static setting. However, in applications such as food delivery and ridesharing, users (drivers) and items (passengers, delivery routes) often arrive in an online fashion, rendering static diversification approach inapplicable. Our algorithm in Chapter 5 offers a method for diversification in such online problems.

The work on environmental sustainability takes on a different technical angle, but its success is nevertheless grounded from the lessons learned from those previous projects. Also, we paid attention to two common issues in applied learning and planning pipeline: label scarcity and label noise. Working with multiple WWF teams in many corners of the world, we built **NewsPanda**, a toolkit which automatically detects and analyzes online articles related to environmental conservation and infrastructure construction. We fine-tune a BERT-based model using active learning methods and noise correction algorithms to identify articles that are relevant to conservation and infrastructure construction. For the identified articles, we perform further analysis, extracting keywords and finding potentially related sources. **NewsPanda** has been successfully deployed by the World Wide Fund for Nature teams in the UK, India, and Nepal since February 2022. We have now scaled it up to cover 60,000 conservation sites globally.

As exciting as these applications are, they only make up half of the picture. Another half of our effort is on technical AI research that goes beyond specific application domains, that takes a step back from immediate applications and then attempts to propose methods to address the pain points shared by all these applications.

Our work on bandit data-driven optimization is one such example. All of our previous works are about one-shot learning and planning. However, in practice we do it in an iterative fashion. Bandit data-driven optimization is the first iterative prediction-prescription framework to address the four pain points in ML for nonprofit applications: small data, data collected only under the default intervention, unmodeled policy objectives, and unforeseen consequences of the intervention. It combines the advantages of online bandit learning and offline predictive analytics in an integrated framework. We propose a novel algorithm for this framework and formally prove that it has no-regret. We demonstrate its great performance on both simulated data and real-world food rescue volunteer recommendation problem studied in previous chapters.

9.1 Future Directions

We proposed our game-theoretical model on cyber deception as a general framework in order to handle various security scenarios. However, there are a few limitations which we believe deserves future work. First, we assumed a single attacker, or a homogeneous attacker population. Future work could investigate the case when we face heterogeneous attackers, how to learn their diverse preferences and then compute optimal strategies against them. Second, our framework is inherently single-shot. Advanced attackers could update their belief about the system through repeated interactions. And hence, future work could consider the deception in a sequential setting. Third, incidental to the sequential setting, attacker's preferences could shift over time. It would be an interesting direction to study how to detect attacker's changing preferences and plan optimally against them.

In our work on volunteer engagement in crowdsourcing food rescue, we took a data-centric approach. That is, we used data to guide us to which volunteers we should send notifications.

This is an indirect approach as it made multiple implicit assumptions about volunteer behaviors. This approach was partly justified by our RCT results, where our algorithm improved the claim rate by a significant margin. However, a lot more research is needed to understand volunteers' motivation in participating in such platform, and to understand how push notifications are translated into volunteers' behavioral change. Only by understanding these questions can we design better algorithms for volunteer engagement in the future.

As for our work on **NewsPanda**, a key challenge when we scale it up to incorporate more conservation sites worldwide is the need to support more languages and more local media sources. This is especially important for the global south, as many high-impact local developments might never reach international news outlets, and commercial translation tools do not yet have satisfactory performance on these low-resource languages. In order for this direction to be practical, we need to train the model in few-shot, or even zero-shot setting, because obtaining labels from each WWF country team is very expensive, and hardly scalable. And hence, this is one of the main future directions coming out of this work.

The bandit data-driven optimization framework has many future directions from both application and technical perspectives. On the application side, one immediate next step is to apply it to the feature deception problem introduced in Chapter 2. In this way, we will tie together the materials in Chapter 2, 5, and 8 in a unified framework. And more generally, this application will show that bandit data-driven optimization is applicable to other security game-like settings as being used in anti-poaching and public safety. On the technical side, there are several challenges. One example is about confounded cost, where instead of receiving the optimization cost u_o and the bandit cost u_b separately, only a total cost is provided. High dimensional covariates were also a challenge as we observed in the practical performance of PROOF. Finally, correlated bandits imposed by constraints covering all individuals are another challenge that has very intuitive real-world motivation.

9.2 Discussion

In the end, let us take a step back to ponder what AI4SG really means. AI4SG is an ambiguous concept, given that neither AI nor social good has a widely accepted definition. Some of the recent efforts try to define AI4SG based on the realized or potential social impact and non-profit nature. However, there is no consensus on such a definition. One can argue that "social impact" is both too exclusive and inclusive: lots of AI4SG research has not (yet) achieved any tangible social impact, and AI research that does have some social impact may not be *doing good*. The "non-profit nature" is similarly debatable since for-profit industry companies arguably contribute a lot to AI4SG, and the majority of research on AI and transportation, which most people would count in AI4SG, is not always intended for non-profit applications. Currently, the most popular set of efforts is to describe AI4SG by listing the application domains of AI technologies, or even simpler, referring to "societal challenges, which have not yet received significant attention by the AI community".[1] As Berendt points out, social good (common good) is referred to as a goal, but not defined [18].

[1]https://aaai.org/Symposia/Spring/sss17symposia.php

We did not attempt to propose any new definition of AI4SG. For most part of this thesis, we implicitly adopted the widely used approach by categorizing by application domains and technical areas of AI. In fact, we believe that the lack of precise definition might be more of a feature than a bug. Just like the lack of common definition for AI helped the field to grow and innovate beyond its boundary, we think an inclusive boundary of AI4SG could encourage more researchers to contribute to this area.

Of course, this is not to pretend that the "bug" does not exist. There are undesirable consequences of this approach. It inevitably leads to an awkward situation where AI4SG research has become an umbrella term that is only a loose connection of disjoint research communities. Yet, in fact, these seemingly disjoint communities have a lot in common beyond the general desire to "do good". The lack of systematic research on working on AI4SG has already led to extreme inefficiency in the numerous AI4SG projects. AI4SG, as a research field about 15 years of age, desperately calls for systematic research of its own to guide the community. To do this, we need to go beyond any particular application domain or any particular technical area, but focus on the entire workflow of working on AI in the social good setting, from problem scoping to deployment. This would give us a chance to uncover the challenges in different stages and aspects of AI4SG projects.

The work presented in this thesis represents our view of AI4SG. However, as much as a thesis is often construed as a final deliverable, it is also a log of a journey. Throughout this journey, we repeatedly questioned ourselves what is AI4SG research, what kind of AI4SG research is valuable, what is the right way to do AI4SG research, and what are the limits of AI4SG research. These are not easy questions. Not only are there no simple answers, but also it could be difficult to face the answers because sometimes the answer means we have to dismiss our own past work, and also it could be convenient to ignore the answers because following the answers could mean sacrificing many short-term benefits. Nevertheless, this thesis is a log of us trying to figure out these questions, and executing our answers through the work presented. If the reader, while reading this thesis, could feel the struggle, the debate, and the choices we made, please do not question your feelings, they are real.

Appendix

Appendix A

Appendix to Chapter 2

A.1 Deferred Algorithms

We show the MILP formulation for the mathematical program $\mathcal{MP}1$. We use $M_c \subseteq M$ to denote the set of continuous features, and $M_d = M - M_c$ denotes the set of discrete features. For discrete feature $k \in M_d$, we assume that η_{ik} and budget B have been processed such that Constraint (2.3) has been modified to

$$\sum_{i \in N} \left(\sum_{k \in M_c} \eta_{ik} |x_{ik} - \hat{x}_{ik}| + \sum_{k \in M_d} \eta_{ik} x_{ik} \right) \leq B.$$

This transformation based on $\hat{x}_{ik} \in \{0,1\}$ simplifies our presentation below.

$$\max_{b,d,g,h,q,s,t,v,y} \quad \sum_{i\in N} t_i \tag{A.1}$$

$$s.t. \quad t_i = ve^{-2W} + \sum_l \gamma_l(v\epsilon - s_{il}) \tag{A.2}$$

$$\sum_{k\in M_c} w_k q_{ik} + \sum_{k\in M_d} w_k b_{ik} - Wv = -\sum_l s_{il} \tag{A.3}$$

$$h_{ik} \geq q_{ik} - \hat{x}_{ik}v, h_{ik} \geq \hat{x}_{ik}v - q_{ik} \qquad \forall k \in M_c \tag{A.4}$$

$$\sum_{i\in N}\left(\sum_{k\in M_d} \eta_{ik}b_{ik} + \sum_{k\in M_c} \eta_{ik}h_{ik}\right) \leq Bv \tag{A.5}$$

$$\epsilon g_{il} \leq s_{il}, s_{i(l+1)} \leq \epsilon g_{il} \qquad \forall l \tag{A.6}$$

$$s_{il} \leq v\epsilon \qquad \forall l \tag{A.7}$$

$$g_{il} \leq v, g_{il} \leq Zy_{il}, g_{il} \geq v - Z(1-y_{il}) \qquad \forall l \tag{A.8}$$

$$b_{ik} \leq v, b_{ik} \leq Zd_{ik}, b_{il} \geq v - Z(1-d_{ik}) \qquad \forall k \in M_d \tag{A.9}$$

$$q_{ik} \in [(\hat{x}_{ik} - \tau_{ik})v, (\hat{x}_{ik} + \tau_{ik})v] \cap [0,1] \qquad \forall k \in M_c \tag{A.10}$$

$$\sum_{i\in N} u_i t_i = 1 \tag{A.11}$$

$$\text{Categorical constraints} \tag{A.12}$$

$$t_i, v, s_{il}, q_{ik}, h_{ik}, g_{il} \geq 0, y_{il} \in \{0,1\} \qquad \forall k \in M_c, \forall l \tag{A.13}$$

$$b_{ik} \geq 0, d_{ik} \in \{0,1\} \qquad \forall k \in M_d \tag{A.14}$$

We establish the variables in the MILP above with the FDP variables as below.

$$t_i = \frac{f_i}{\sum_{i\in N} f_i u_i}, \qquad v = \frac{1}{\sum_{i\in N} f_i u_i} \tag{A.15}$$

$$h_{ik} = \frac{|x_{ik} - \hat{x}_{ik}|}{\sum_{i\in N} f_i u_i}, \qquad q_{ik} = \frac{x_{ik}}{\sum_{i\in N} f_i u_i}, \qquad \forall k \in M_c \tag{A.16}$$

$$d_{ik} = x_{ik}, \qquad b_{ik} = \frac{x_{ik}}{\sum_{i\in N} f_i u_i}, \qquad \forall k \in M_d \tag{A.17}$$

$$s_{il} = \frac{z_{il}}{\sum_{i\in N} f_i u_i}, \qquad g_{il} = \frac{y_{il}}{\sum_{i\in N} f_i u_i}, \qquad \forall l \tag{A.18}$$

All equations above involving index i without summation should be interpreted as applying to all $i \in N$.

A.2 Exact Algorithms for Special Cases

A.2.1 Deception cost on discrete features

In our first attempt at exact algorithms, we assume the cost of deception is only associated with discrete features, i.e. $\eta_{ik} = 0$ if k is a continuous feature.

Algorithm 10: Greedy

```
1  Use gradient-based method to find $x^{max} \approx \arg\max_x f(x)$ and $x^{min} \approx \arg\min_x f(x)$.
2  Sort the targets such that $u_1 \le u_2 \le \cdots \le u_n$.
3  Initialize $i = 1, j = n$.
4  while $i < j$ and budget $> 0$ do
5      Let $x_i \longleftarrow x^{max}$ if
6      if $Cost(x_i \longleftarrow x^{max}) \le$ remaining budget then
7          $x_i \longleftarrow x^{max}$, decrease the budget, $i = i + 1$.
8      if $Cost(x_j \longleftarrow x^{min}) \le$ remaining budget then
9          $x_j \longleftarrow x^{min}$, decrease the budget, $j = j - 1$.
10 return feature configuration $x$
```

Recall that we use $M_c \subseteq M$ to denote the set of continuous features, and $M_d = M - M_c$ denotes the set of discrete features. Suppose $x_i^d = (x_{ik})_{k\in M_d}$ and $x_i^c = (x_{ik})_{k\in M_c}$, and let $x_i = (x_i^d, x_i^c)$ be the observable features decomposed into discrete features and continuous features. Our score function $f(x_i) = \exp\{\sum_{k\in M} w_k x_{ik}\}$ can be factorized into $f(x_i) = f_d(x_i^d) f_c(x_i^c)$, where f_d, f_c are scores considering discrete and continuous features only, respectively.

Let $A_i^d = \{x_i^{d_j} : j = 1, \dots, k\}$ be the finite set of possible discrete observable feature combinations at target i. Each $x_i^{d_j} \in \{0,1\}^{m_d}$ is a m_d-dimensional vector, where $m_d = |M_d|$ is the number of discrete features in FDP. Based on the hidden discrete features $\hat{x}_i^d$ and the feasible regions $A(\hat{x}_{ik})$, we can compute both A_i^d and all possible scores $f_{ij}^d = f_d(x_i^{d_j})$. Similarly, we can compute the interval $[\alpha_i, \beta_i]$ in which the continuous score f_i^c could possibly lie, since each continuous feature x_{ik} can take value in an interval $A(\hat{x}_{ik})$.

Subsequently, we formulate the following mathematical program. The binary variable $y_{ij} = 1$ if target i's discrete observable features are the j-th combination in A_i^d, that is, $x_i^d = x_i^{d_j} \in A_i^d$. The cost c_{ij} for setting the discrete observable features to $x_i^{d_j}$ could be computed accordingly.

$$
\begin{aligned}
\min_{y, f^c} \quad & \frac{\sum_{i\in N}\sum_{j=1}^{k} u_i f_{ij}^d y_{ij} f_i^c}{\sum_{i\in N}\sum_{j=1}^{k} f_{ij}^d y_{ij} f_i^c} & \\
s.t. \quad & \sum_{j=1}^{k} y_{ij} = 1 & \forall i \in N \\
& \sum_{i\in N}\sum_{j=1}^{k} c_{ij} y_{ij} \le B & \\
& f_i^c \in [\alpha_i, \beta_i] & \forall i \in N \\
& y_{ij} \in \{0,1\} & \forall i \in N, j \in [k]
\end{aligned}
$$

We may apply the same linearization method as before to obtain a MILP.

Solving this MILP yields the optimal discrete feature configuration, as well as the optimal scores f_i^c's of the continuous features. One may then solve the system of linear equations $\ln f_i^c =$

$\sum_{k \in M} w_k x_{ik}$ for $i \in N$ for the optimal continuous features. Since the feasible regions of the continuous features are connected, there exists at least one solution to the system of equations. We remark that when all features are continuous, the above approach finds the optimal feature configuration in polynomial time.

A.2.2 No budget and feasibility constraints

We present an efficient algorithm when the defender has no budget and feasibility constraints. [135] proposed a greedy algorithm in a similar setting (with feasibility constraints), whose complexity is polynomial in the size of "type space", which is still exponential in the representation of FDP, not to mention that with a single continuous feature the size of our "type space" becomes infinite. Furthermore, the probabilistic behavior of the attacker in FDP makes their key strategy inapplicable.

Since the defender aims at minimizing her expected loss, one intuitive idea is to give the lowest score to the target with the highest loss u_i. In fact, we show below that the defender should configure the features at each target in only two possible ways: the ones which maximizes or minimizes the score. First, we assume that the defender's losses have been sorted in ascending order $u_1 \leq u_2 \leq \cdots \leq u_n$.

Lemma A.2.1. *Let $x = (x_1, \dots, x_n)$ be an optimal observable feature configuration. There exists a permutation σ on N where $x^\sigma = (x_{\sigma(1)}, \dots, x_{\sigma(n)})$ is also an optimal observable feature configuration, and*

$$f(x_{\sigma(1)}) \geq f(x_{\sigma(2)}) \geq \cdots \geq f(x_{\sigma(n)}).$$

In particular, if $u_1 < u_2 < \cdots < u_n$, σ can be the identity permutation.

Proof. We prove by contradiction. Suppose $i < j$ and $f(x_i) < f(x_j)$. We have

$$\begin{aligned}&(f(x_j)u_i + f(x_i)u_j) - (f(x_i)u_i + f(x_j)u_j)\\ &= (f(x_j) - f(x_i))(u_i - u_j) \leq 0\end{aligned}$$

and the inequality is strict if $u_i < u_j$. Thus, when $u_i < u_j$, if we swap the features on target i and j, we would strictly improve the solution, which contradicts x being optimal. When $u_i = u_j$, we could swap the observed features on node i and j, and strictly decrease the number of score inversions. □

Lemma A.2.2. *There exists an optimal observable feature configuration $x = (x_1, \dots, x_n)$ such that, for some $j \in N - \{n\}$, if $i \leq j$, $f(x_i) = \max_{x_i'} f(x_i')$; otherwise, $f(x_i) = \min_{x_i'} f(x_i')$.*

Proof. Let x be an optimal observable feature configuration. Fix a target $i \in N$. Consider an alternative configuration $\tilde{x}_i$ for target i, while keeping features of other targets unchanged. We have

$$\begin{aligned}&\frac{f(x_i)u_i + \sum_{j \neq i} f(x_j)u_j}{f(x_i) + \sum_{j \neq i} f(x_j)} - \frac{f(\tilde{x}_i)u_i + \sum_{j \neq i} f(x_j)u_j}{f(\tilde{x}_i) + \sum_{j \neq i} f(x_j)}\\ &\propto \left((f(x_i) - f(\tilde{x}_i)) \left(\sum_{j \neq i} f(x_j)(u_i - u_j) \right) \right)\end{aligned}$$

Depending on its sign, we could improve the solution x by making $f(\tilde{x}_i)$ greater or smaller than $f(x_i)$, and obviously we should take it to extreme by setting $f(\tilde{x}_i) = \max_{x'_i} f(x'_i)$ or $f(\tilde{x}_i) = \min_{x'_i} f(x'_i)$. Since we assumed x is optimal, then we know $\tilde{x} = (\tilde{x}_i, x_{-i})$ is also optimal, with $f(\hat{x}_i)$ at an extreme value. By Lemma A.2.1, we could permute the features in $\tilde{x}$ so that the scores are in decreasing order. After applying the above argument repeatedly, the score of each target achieves either the maximum or minimum score possible. Therefore, there exists some $j \in N - \{n\}$, such that

$$\begin{aligned}&\max_{x'_i} f(x'_i) = f(x_1) = \cdots = f(x_j)\\ &\geq f(x_{j+1}) = \cdots = f(x_n) = \min_{x'_i} f(x'_i)\end{aligned}$$

□

Theorem A.2.3. *The optimal feature configuration can be found in $O(n \log n + m)$ time.*

Proof. We can do an exhaustive search on the "cut-off" node j in Lemma A.2.2. With bookkeeping, the search can be done in $O(n)$ time. Since f is monotone in each observable feature, the maximum and minimum score can be found in $O(m)$ time. Sorting the targets' losses takes $O(n \log n)$ time. □

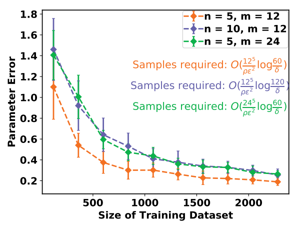
A.3 Additional Experiments

A.3.1 Experiments in the main text

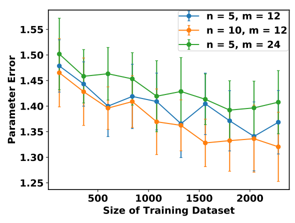
Learning In addition to the mean total variation distance reported in the main text, we present another metric to measure the performance of learning. We consider $|\hat{\theta} - \theta|$, the L_1 error in the score function parameter θ, which directly relates to the sample complexity bound in Theorem 2.4.1. Since the dimension of θ depends on the number of features k and other factors, we consider the L_1 error divided by the number of parameters and report this metric in Fig. A.1a and Fig. A.1b.

In Fig A.1a, the L_1 error of CF decreases as the sample size increases. The complexity bounds in Theorem 2.4.1 are marked in Fig A.1a. We need much fewer samples in practice to achieve a small learning error. For NN-3 score function, the learning error is larger as shown in Fig. A.1b, even though Fig. 2.1b in the main text shows the total variation distance is small. This suggests that the loss surface for the NN-3 score function is more complex. The solution gap in Fig. 2.1h is much larger than that in Fig. 2.1f, which can partly be explained by the fact that at the same level of TV error, the learned classical score function score function is closer to the ground truth than the NN-3 score function, and thus performs better in the learning and planning pipeline.

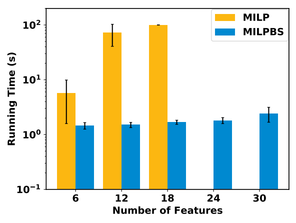
Planning In the main text, we showed how the number of targets effect the running time and solution gap. In In Fig. A.1c- A.1e, we show how the number of features affect these two metrics. For the classical score function, Fig. A.1c shows that the running time for MILP increases with the number of features, and MILPBS is much more scalable. Fig. A.1d and A.1e show the running



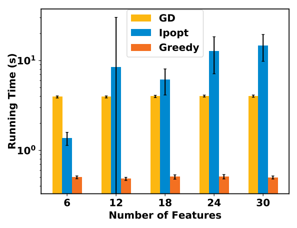
(a) Learning, classical score function



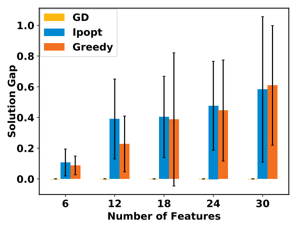
(b) Learning, NN-3 score function



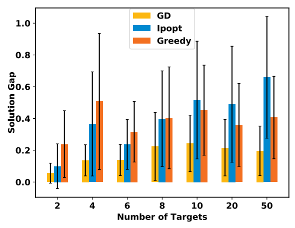
(c) Planning with classical score function, $n = 10$



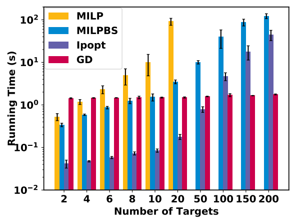
(d) Planning with NN-3 score function, $n = 10$



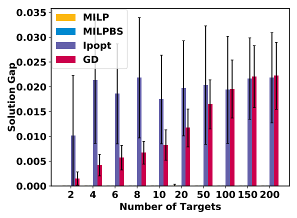
(e) Planning with NN-3 score function, $n = 10$



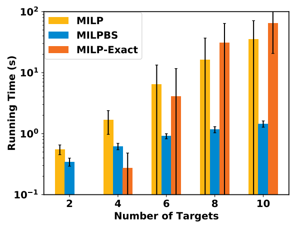
(f) Learning + Planning, NN-3 score function, $m = 12$



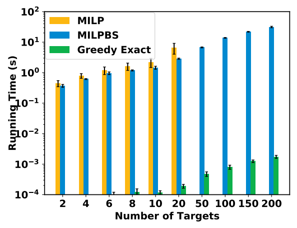
(g) Continuous features



(h) Continuous features



(i) Cost on discrete features, classical score function, $m = 12$



(j) No budget and feasibility constraints, classical score function, $m = 12$

Figure A.1: Additional experimental results

time and solution gap fixing $n = 10$. The running time of GD and Greedy does not change much across different problem sizes, yet Ipopt runs slower than the former two on most problem instances. GD also has smaller solution gap than Ipopt and Greedy on most instances.

Combining learning and planning Given enough data, Fig. A.1f shows that GD can achieve a solution gap below 0.2 with as many as 50 targets.

A.3.2 Experiments for the special cases

We present the performance of our algorithms on some special cases of FDP, as studied in Appendix A.2.

When all features are continuous, in addition to our MILP, we may use non-convex solver or GD as a heuristic to find optimal feature configurations. Fig. A.1g shows that these two heuristics scale better than the approximation algorithms. In Fig. 2.1e and Fig. A.1e, we showed

Discrete feature $k \in M_d$		Continuous feature $k \in M_c$	
Variable	Distribution	Variable	Distribution
$\lvert M_d \rvert$	$2m/3$	$\lvert M_c \rvert$	$m/3$
η_{ik}	$U(-3, 3)$	η_{ik}	$U(0, 3)$
τ_{ik}	N/A	τ_{ik}	$U(0, 0.25)$
$\hat{x}_{ik}$	$U\{0, 1\}$	$\hat{x}_{ik}$	$U(0, 1)$
u_i		$U(0, 1)$	
Variable		Distribution	
B		$U(0, 0.2C_{\max})$	
$C_{\max}$		$\sum_{i\in N}\sum_{k\in M_c} \eta_{ik} \min(\hat{x}_{ik}, 1 - \hat{x}_{ik}, \tau_{ik}) + \sum_{k\in M_d} \eta_{ik}$	

Table A.1: FDP parameter distributions for experiments on classical attacker score function. Used in Fig. 2.1c, A.1c, 2.1f, 2.1g

that GD demonstrates the best solution quality among the heuristics on NN-3 score functions. A natural question to ask is if GD is in practice close to exact. In Fig. A.1h, we show that at least when we have a single-layer score function, GD solution is not far from optimal, though its solution deteriorates as the problem size grows. Non-convex solver yields a small solution gap as well.

When deception cost is only associated with discrete features, we presented an exact MILP formulation in Appendix A.2.1. Fig. A.1i shows that it is especially efficient on very small problems. Yet as the problem size grows its efficiency decreases quickly.

Finally, in Appendix A.2.2, we proposed a $O(n \log n + m)$ time algorithm for FDP without budget and feasibility constraints. Indeed, as shown in Fig. A.1j, the algorithm's running time is several magnitudes less than the MILP-based approaches.

A.4 Experiment Parameters and Hyper-parameters

NN-3 score function architecture The NN-3 score function has input layer of size $m \times 24$, second layer 24×12, and third layer 12×1. The first and second layers are followed by a tanh activation, and the last layer is followed by an exponential function. The neural network parameters are initialized uniformly at random in $[-0.5, 0.5]$. We use this network architecture for all of our experiments.

FDP parameters for classical score function We detail in Table A.1 the parameter distributions used in the planning and combined learning and planning experiments, when the adversary assumes the single-layer score function. These distributions apply to the results shown in Fig. 2.1c, A.1c, 2.1f, 2.1g.

FDP parameters for NN-3 score function We detail in Table A.2 the parameter distributions used in the planning and combined learning and planning experiments, when the adversary assumes the NN-3 score function. These distributions apply to the results shown in

Variable	Distribution
η_{ik}	$U(0,1)$
τ_{ik}	1
$\hat{x}_{ik}$	$U(0,1)$
u_i	$U(0,1)$
B	$U(0, 0.2nm)$

Table A.2: FDP parameter distributions for experiments on NN-3 attacker score function. Used in Fig. 2.1d,2.1e, A.1d, A.1e,2.1h, A.1f

Parameter	Fig 2.1h ($\lvert D_{train} \rvert$ > 10000), A.1f	Fig. 2.1g	All other experiments
Learning rate	{1e-3, 1e-2, 1e-1} ⟶ 1e-1	{1e-3, 1e-2, 1e-1} ⟶ 1e-1	{1e-3, 1e-2, 1e-1} ⟶ 1e-1
Number of epochs	{20, 30, 60} ⟶ 30	{20, 30, 60} ⟶ 30	{10, 20, 40} ⟶ 20
Steps per epoch	{20, 30, 40} ⟶ 30	12	{10, 20} ⟶ 10
Batch size	{120, 600, 5000} ⟶ 5000	{120, 600, 5000} ⟶ 5000	$\lvert D_{train} \rvert$/Number of epochs

Table A.3: Hyper-parameters for the experiments. The values between the braces are the ones we tested. The values after the arrows are the ones we used in generating the results.

Fig. 2.1d,2.1e, A.1d, A.1e,2.1h, A.1f.

Hyper-parameters for learning Table A.3 shows the hyper-parameters we used in learning the attacker's score function f.

Appendix B

Appendix to Chapter 3

B.1 Deferred Algorithms

B.1.1 Attacker's Better Response Heuristic

In light of the hardness of finding the adversary's best response, we consider a greedy heuristic. Leveraging Theorem 3.5.2, Greedy (Alg. 11) allocates the adversary's budget to websites in decreasing order of the ratio $r_w = \frac{t_w(1-x_w)/t_w^{all}}{\alpha_w}$, where α_w is a tuning parameter. We replace the MILP for $\mathcal{P}_2(x)$ in CyberTWEAK with Alg. 11 to find an adversary's better response. If it does not yield a new effort vector, the MILP is called. The column generation process terminates if the MILP again does not find a new effort vector. We refer to this entire procedure as GreedyTWEAK. Note that GreedyTWEAK also terminates with the optimal solution. Although Greedy (Alg. 11) does not provide an approximation guarantee, it performs well in practice. As we show in the experiment section, in practice the accuracy of its solution improves as the size of the problem grows. We also considered a dynamic programming algorithm which is exact and runs in pseudo-polynomial time. However, its practical performance is unsatisfactory.

Algorithm 11: Greedy

```
1 Sort the websites in decreasing order of $r_w = \frac{t_w(1-x_w)/t_w^{all}}{\alpha_w}$.
2 foreach website w in the sorted order do
3     if remaining attack budget ≥ attack cost $\pi_w$ then
4         Attack this website w with maximum effort allowed
5     if running out of budget then break ;
```

B.1.2 Baseline Algorithm for $\mathcal{P}_1$

We show the details of one of our baseline algorithms, All Actions, in Alg. 12. Let $\mathcal{A}$ denote the set of actions available to the adversary such that the budget constraint is satisfied. Each

Algorithm 12: All Actions

```
1  foreach $(a^*, w^*) \in \mathcal{A} \times W$ where $w^* \in a^*$ do
2      foreach website $w \in W$ do
3          if $w = w^*$ then
4              Define $z_w = \min\{B_e - \sum_{w \in a^*, w \neq w^*} t_w^{all}, t_{w^*}^{all}\}$
5          else if $w \in a^*$ then
6              Define $z_w = t_w^{all}$
7          else
8              Define $z_w = 0$
9          Define $k_w = \frac{t_w}{t_w^{all}} z_w$
10         foreach $(\hat{a}, \hat{w}) \in \mathcal{A} \times W$ where $\hat{w} \in \hat{a}$ do
11             Define $\hat{k}_w$ similarly as above, for each $w \in W$.
12             Add to $BR(a^*, w^*)$ the following linear constraint
               $\sum_{w \in a^*} k_w(1 - x_w) \geq \sum_{w \in \hat{a}} \hat{k}_w(1 - x_w)$
13     Solve the following LP
```

$$\min_{x,v} \quad v \tag{B.1}$$

$$\text{s.t.} \quad v \geq \sum_{w \in W} k_w(1 - x_w) \tag{B.2}$$

$$\text{linear constraints in } BR(a^*, w^*) \tag{B.3}$$

$$\sum_{w \in W} c_w t_w x_w \leq B_d \tag{B.4}$$

$$x_w \in [0, 1], \qquad \forall w \in W \tag{B.5}$$

```
14 Select the best solution out of all the LPs.
```

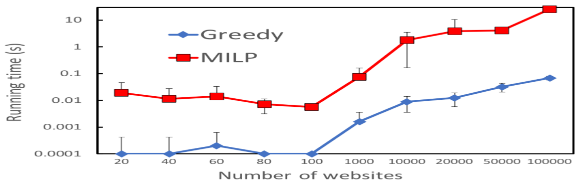
action $a^* \in \mathcal{A}$ is a set of websites being compromised. According to Theorem 3.5.2, among all the websites w compromised in a^*, the adversary puts "partial" effort $e_w \in (0, t_w^{all}$ on at most one website w^*. Therefore, the action-website pairs (a^*, w^*) fully characterize the adversary's strategies. Alg. 12 works by finding the optimal defender strategy, assuming each action-website pair is the optimal strategy for the adversary.

B.2 Deferred Experiments

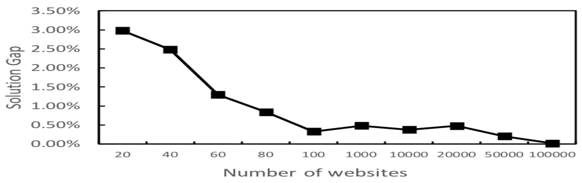
We present additional experiments on the adversary's best response problem. In the greedy algorithm (Alg. 11), the adversary selects websites based on a decreasing order of $r_w = \frac{t_w(1-x_w)/t_w^{all}}{\alpha_w}$. Here, α_w is the tuning parameter. With different choices of α_w, we compare the output value $\text{OPT}_{\text{Greedy}}$ of Greedy with the optimal value OPT obtained by solving the MILP $\mathcal{P}_2(x)$. Table B.1 shows the solution gap $\frac{\text{OPT}-\text{OPT}_{\text{Greedy}}}{\text{OPT}}$. We observe that $\alpha_w = \pi_w$ yields the smallest solution gap.

α_w	$\frac{\text{OPT}-\text{OPT}_{\text{Greedy}}}{\text{OPT}}$
π_w	0.0079
$\pi_w/B_a + 1/B_e$	0.0285
1	0.0082

Table B.1: Solution gaps of different greedy heuristics for the adversary best response problem. Results are averaged over 5 runs on different problem sizes $|W| = 100, 200, \ldots, 500$.



(a) Greedy running time



(b) Greedy solution gap

We also tested other choices for α_w such as $(\pi_w/B_a)^p + (1/B_e)^q$ for different powers p and q, yet they do not yield better optimization gaps. Hence we fix $r_w = \frac{t_w(1-x_w)/t_w^{all}}{\pi_w}$ in subsequent experiments.

Fig. B.1b shows Greedy's solution gap decreases to near zero as the problem size grows. In addition, Greedy typically runs within 1% of the time of the MILP.

B.3 Experiment Parameters

Table B.2 shows the distribution from which the parameters are generated in most of our experiments. In Table B.3, we detail the parameters used in the experiment in Fig. 3.2e.

Variable	Distribution
t_w^{all}	$U(350, 750)$
t_w	$U(50, 100)$
c_w	$U(1, 4)$
π_w	$U(30, 54)$
B_d	$U(0.11 \sum_{w \in W} c_w t_w, 0.71 \sum_{w \in W} c_w t_w)$
B_a	$U(0.1 \sum_{w \in W} \pi_w, 0.8 \sum_{w \in W} \pi_w)$
B_e	$U(0.2 \sum_{w \in W} t_w^{all}, 0.8 \sum_{w \in W} t_w^{all})$

Table B.2: Parameter distribution

In addition, in the case of small effort budget, B_e is generated uniformly between 1 and $\min_{w \in W} t_{all}^w$.

For large scale instances, we set different websites to have different importance, motivated by the fact that people do not visit all websites with equal frequency. We split W into W_1, W_2 with $|W_1| : |W_2| = 1 : 9$. Websites in W_1 have a large portion of traffic from the organization and

For $w \in W_1$		For $w \in W_2$	
Variable	Distribution	Variable	Distribution
t_w^{all}	$U(60, 110)$	t_w^{all}	$U(20, 70)$
t_w	$U(45, 55)$	t_w	$U(3, 8)$
c_w	$U(2, 6)$	c_w	$U(1, 3)$
π_w	3	π_w	3
B_d	$U(0, 10 \sum_{w \in W} c_w t_w / \lvert W \rvert)$		
B_a	$U(0.1 \sum_{w \in W} \pi_w, 0.8 \sum_{w \in W} \pi_w)$		
B_e	$U(0, 3 \sum_{w \in W} t_w^{all} / \lvert W \rvert)$		

Table B.3: Parameter distributions for the experiment on large instances.

those in W_2 have a smaller portion. Thus, W_1 and W_2 follow different distributions (Table B.3). The attacker has a uniform cost of attack. In less than 4 of the 20 instances DWE did not reduce the problem size by much. We report in Fig. 3.2e the majority group where DWE eliminated a significant number of websites. $|W_1|/|W|$ could be a lot smaller in reality, and our algorithms with DWE would run even faster.

B.4 Discussion

Assumptions and generality We assumed that the attack will succeed if and only if the network packet is unaltered. If the attacker can obtain the true system information with probability p_w even if the packet is altered, we may modify the objective in Eq. (3.1) to $\sum_w t_w(1 - x_w(1 - p_w))e_w/t_w^{all}$. If the organization has other countermeasures (e.g. Bromium browser VMs), the attack may fail with probability q_w even if the packet is unaltered, the objective then becomes $\sum_w t_w(1 - x_w)(1 - q_w)e_w/t_w^{all}$. Thus, our algorithm can account for different levels of adversary and defender sophistication.

We do not attempt to claim that altering the network packets is a panacea to all watering hole attacks. Cyber attackers have many tools to circumvent existing deception techniques. Nonetheless, the proposed deception technique increases their uncertainty about the true nature of the environment, which leads to more cost on them, e.g. technical complexity and increased exposure. This uncertainty ties into our consideration of the attacker's scanning effort e_w and budget B_e, as the attacker cannot easily obtain or trust the basic information in the network packets.

Limitations The generality notwithstanding, We acknowledge a few limitations of our work and potential problems in large-scale deployment. First, if an organization is the sole user of our method and if the attacker has (possibly imperfect) clue about the source of traffic from the start, randomizing network packet information might serve as an unintended signal to the attacker, reducing the effort needed e_w to identify traffic from the targeted organization. Second, by manipulating the web traffic, the organization is effectively monitoring its employees' internet activities. Although in many jurisdictions this is allowed when doing properly, the potential ethical issues must be carefully addressed.

Appendix C

Appendix to Chapter 8

C.1 Regret Bounds Using Sample Complexity Characterization

Let us first introduce the multivariate Rademacher complexity and its associated generalization bounds, which were introduced by Bertsimas and Kallus [19].

Definition C.1.1. *Given a sample $S_n = \{s_1, \dots, s_n\}$, the empirical multivariate Rademacher complexity of a class of functions $\mathcal{F}$ taking values in $\mathbb{R}^d$ is defined as*

$$\hat{\mathfrak{R}}_n(\mathcal{F}; S_n) = \mathbb{E}_\sigma \left[\sup_{g \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} \sigma_{ik} g_k(s_i) \right]$$

where σ_{ik}'s are independent Rademacher random variables. The multivariate Rademacher complexity is defined as $\mathfrak{R}_n(\mathcal{F}) = \mathbb{E}_{S_n}[\hat{\mathfrak{R}}_n(\mathcal{F}; S_n)]$

Theorem C.1.1. *[19] Suppose function $c(z; y)$ is bounded and equi-Lipschitz in z:*

$$\sup_{z \in Z, y \in Y} c(z; y) \leq \bar{c}, \text{ and } \sup_{z \neq z' \in Z, y \in Y} \frac{c(z; y) - c(z'; y)}{\|z - z'\|_\infty} \leq L < \infty$$

For any $\delta > 0$, each of the following events occurs with probability at least $1 - \delta$,

$$\mathbb{E}[c(z(X); Y)] \leq \frac{1}{n} \sum_{i=1}^{n} c(z(x^i); y^i) + L\mathfrak{R}_n(\mathcal{F}) + \bar{c}\sqrt{\frac{\log(1/\delta)}{2n}}$$

$$\mathbb{E}[c(z(X); Y)] \leq \frac{1}{n} \sum_{i=1}^{n} c(z(x^i); y^i) + L\hat{\mathfrak{R}}_n(\mathcal{F}; S_n) + 3\bar{c}\sqrt{\frac{\log(2/\delta)}{2n}}.$$

When the function $c(z; y)$ is nonnegative, we have

$$\mathbb{E}[c(z(X); Y)] \leq \frac{1}{1 - \delta} \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^{n} c(z(x^i); y^i) \right] + L\mathfrak{R}_n(\mathcal{F}) + \bar{c}\sqrt{\frac{\log(1/\delta)}{2n}} \tag{C.1}$$

Before we proceed, we make the following assumption.

Assumption C.1.1. *With n training data points $x_1, \dots x_n$, the learning algorithm learns a predictor $\hat{f}$ such that $\mathbb{E}\left[\sum_{i=1}^n \left\|f(x_i) - \hat{f}(x_i)\right\|_2^2\right]$ is constant with respect to n.*

Although this assumption might appear somewhat unintuitive, it is actually satisfied when, for example, $f \in \mathcal{F}$ comes from the class of all linear functions, ordinary least squares regression used as the learning algorithm satisfies this assumption. In that case, we have $\mathbb{E}\left[\sum_{i=1}^n \left\|f(x_i) - \hat{f}(x_i)\right\|_2^2\right] = O(md)$.

Theorem C.1.2. *Suppose we use any learning algorithm that satisfies Assumption C.1.1, including but not limited to OLS regression. The regret of Algorithm 8 is $\tilde{O}\left(md\sqrt{nT}\right)$, with probability $1-\delta$.*

Proof. First, let's compute the Rademacher complexity of the linear hypothesis class, for completeness and for our specific setting. Let F_k be the k-th row of matrix F. We have

$$\begin{aligned}
\hat{\mathfrak{R}}_n(\mathcal{F}; X_n) &= \mathbb{E}_\sigma\left[\sup_{F\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n\sum_{k=1}^d \sigma_{ik}F_k^\dagger x_i\right] = \mathbb{E}_\sigma\left[\sup_{F\in\mathcal{F}} \frac{1}{n}\sum_{k=1}^d F_k^\dagger\left(\sum_{i=1}^n \sigma_{ik}x_i\right)\right] \\
&\le \mathbb{E}_\sigma\left[\sup_{F\in\mathcal{F}} \frac{1}{n}\sum_{k=1}^d \|F_k\|_1\left\|\sum_{i=1}^n \sigma_{ik}x_i\right\|_2\right] \qquad \text{(Cauchy-Schwartz)} \\
&\le \mathbb{E}_\sigma\left[\sup_{F\in\mathcal{F}} \frac{1}{n}\sum_{k=1}^d \|F\|_\infty\left\|\sum_{i=1}^n \sigma_{ik}x_i\right\|_2\right] \\
&\le \sum_{k=1}^d mK_F\mathbb{E}_\sigma\left[\left\|\frac{1}{n}\sum_{i=1}^n \sigma_{ik}x_i\right\|_2\right] \le \sum_{k=1}^d mK_F\left(\mathbb{E}_\sigma\left[\left\|\frac{1}{n}\sum_{i=1}^n \sigma_{ik}x_i\right\|_2^2\right]\right)^{1/2} \qquad \text{(Jensen)} \\
&\le \sum_{k=1}^d mK_F\left(\mathbb{E}_\sigma\left[\frac{1}{n^2}\sum_{i=1}^n \|x_i\|_2^2 + \frac{2}{n^2}\sum_{i<j}\sigma_{ik}\sigma_{jk}x_i^\dagger x_j\right]\right)^{1/2} \\
&= \sum_{k=1}^d mK_F\left(\mathbb{E}_\sigma\left[\frac{1}{n^2}\sum_{i=1}^n \|x_i\|_2^2\right]\right)^{1/2} = \frac{dmK_FK_X}{\sqrt{n}}
\end{aligned}$$

Using Equation C.1, we have

$$\begin{aligned}
\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] &\le \frac{1}{(1-\delta)nt}\mathbb{E}\left[\sum_{i=1}^n\sum_{\tau=1}^t \left\|f(x_i^\tau) - \hat{f}(x_i^\tau)\right\|_2\right] + L\mathfrak{R}_{nt}(\mathcal{F}) + \bar{c}\sqrt{\frac{\log(T/\delta)}{2nt}} \\
&\le \frac{1}{(1-\delta)nt}\mathbb{E}\left[\sqrt{nt\sum_{i=1}^n\sum_{\tau=1}^t \left\|f(x_i^\tau) - \hat{f}(x_i^\tau)\right\|_2^2}\right] + L\mathfrak{R}_{nt}(\mathcal{F}) + \bar{c}\sqrt{\frac{\log(T/\delta)}{2nt}} \\
&\le \frac{1}{(1-\delta)\sqrt{nt}}\sqrt{\mathbb{E}\left[\sum_{i=1}^n\sum_{\tau=1}^t \left\|f(x_i^\tau) - \hat{f}(x_i^\tau)\right\|_2^2\right]} + L\mathfrak{R}_{nt}(\mathcal{F}) + \bar{c}\sqrt{\frac{\log(T/\delta)}{2nt}}
\end{aligned}$$

By Assumption C.1.1, the term under square root is constant w.r.t. nt, under regularity conditions. Thus, we have

$$\mathbb{E}_{X,\epsilon}\left[\left\|\mathbb{E}_{c_i^t|x_i^t}[c_i^t] - \hat{c}_i^t\right\|_2\right] = \tilde{O}\left(md(nt)^{-1/2}\right)$$

The rest of the proof follows from Lemma 8.4.4. □

This result is almost identical as Theorem 8.4.5. However, the intent to work with Rademacher complexity is that we hope to at least get some bound when we move beyond the linear regression scenario. Let us consider a feed-forward neural network with ReLU activation. There are existing results which shows that the Rademacher complexity is $O(1/\sqrt{n})$ [56]. Thus, if we accept Assumption C.1.1, we would have the following result.

Theorem C.1.3. *Supose the learning problem is fitting a neural network and we use any learning algorithm that satisfies Assumption C.1.1. The regret of Algorithm 8 is $\tilde{O}(nT^{1/2})$, with probability $1-\delta-\lambda$, ignoring the dependency on d and m.*

Finally, we extend the previous results to the more general case where interventions affect the label distribution. Please refer to the setting described in Section 8.4.3 and Algorithm 9.

Theorem C.1.4. *Suppose there are finitely many actions. Assuming we use any learning algorithm that satisfies Assumption C.1.1, including but not limited to OLS regression, the regret of Algorithm 8 is $\tilde{O}\left(|W|^{1/3}(md)^{2/3}nT^{2/3}\right)$, with probability $1-\delta$.*

Proof. After the exploration phase, we have had $\tilde{n} = n\tilde{T}/|W|$ data points for training the predictor for each w_i. Similar to our approach in Theorem C.1.2, we have

$$\begin{aligned}
&\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t)\right\|_2\right] \\
&\leq \frac{1}{(1-\delta)\tilde{n}}\mathbb{E}\left[\sum_{i=1}^{\tilde{n}}\left\|f(x_i)-\hat{f}(x_i)\right\|_2^2\right] \\
&+ L\mathfrak{R}_{\tilde{n}}(\mathcal{F}) + \bar{c}\sqrt{\frac{\log(T/\delta)}{2\tilde{n}}} \\
&= \tilde{O}\left(|W|^{1/2}md(n\tilde{T})^{-1/2}\right), \qquad \forall t > \tilde{T}
\end{aligned}$$

Thus, the regret is

$$\begin{aligned}
&O\Bigg(n\tilde{T} + n\sqrt{8mT\beta_T\log T} \\
&+ \sum_{t=\tilde{T}+1}^{T}\sum_{i=1}^{n}\mathbb{E}\left[\left\|\mathbb{E}_{c_i^t|x_i^t,w_i^t}[c_i^t(w_i^t)] - \hat{c}_i^t(w_i^t)\right\|_2\right]\Bigg) \\
&= \tilde{O}\left(n\tilde{T} + n\sqrt{8mT\beta_T\log T} + nT|W|^{1/2}dm(n\tilde{T})^{-1/2}\right) \\
&= \tilde{O}\left(|W|^{1/3}(md)^{2/3}nT^{2/3}\right)
\end{aligned}$$

where we let $\tilde{T} = T^{2/3}|W|^{1/3}(md)^{2/3}$. □

C.2 Details of the Food Rescue Experiment

C.2.1 Recommender System Model

We build our recommender system using a neural network. We show the neural network architecture in Table C.1. The input to the neural network is the feature vector of a rescue-volunteer

Table C.1: Neural network architecture

Layer	Operation	Hidden Units
1	Dense (ReLU)	192
2	Dense (ReLU)	512
3	Dense (Logistic)	16

pair. The feature vector passes through three dense layers. Each layer is followed by a ReLU activation function, except for the last layer where we output a single number which is then converted to a number between 0 and 1 by the logistic function. This output represents the likelihood that this volunteer will claim this rescue trip. We use the cross entropy loss to train the neural network. At prediction time, for a given rescue, we pass the feature vectors of the rescue-volunteer pairs for all volunteers on a fixed rescue through the network and obtain a likelihood estimate for each volunteer.

C.2.2 Training

We performed all the experiments in this chapter on an Intel Core i5-7600K CPU and 32GB RAM.

We use the data from March 2018 to October 2019 for feature preparation. Recall that some of the features we use are related to the volunteer's historical number of rescues. We use the data from this period to generate such features. Then, we use the 556 rescues from November 2019 to March 2020 for learning and prediction in the actual experiment. In this way we avoid the potential data leakage.

In these 556 rescues, we select the first 300 of them to be the initial dataset for the bandit data-driven optimization (refer to Line 1 in Procedure 7, Section 8.3). From the remaining 256 rescues, we randomly sample and set aside 150 rescues as the validation dataset. Finally we take the 50 earliest rescues from the remaining 106 rescues to run the PROOF algorithm for 50 iterations, each iteration corresponding to one rescue. At time step t, our training set consists of the 300 rescues in the initial dataset and all the rescues we have seen from time step 1 up to time step $t-1$. When training the recommender system at each time step, we use the Adam optimizer with learning rate 1×10^{-3}. We stop the training when the 3-episode moving average loss on the validation set stops decreasing. In the following paragraph, we discuss our way to address a key challenge in the training dataset in more detail.

Negative Sampling As mentioned earlier, there is an extremely high label imbalance in our dataset. Each rescue typically has only one volunteer who claimed it, which means, theoretically, the ratio between negative and positive examples is about $100 : 1$. Using the method introduced in Section 5.4.2, we can obtain a selected set of negative examples D_n derived from push notifications and another set of negative examples D_c derived from dispatcher calls. The set D_c is about the same size as the positive examples D_p, while $|D_n| : |D_p| \approx 11 : 1$. When training the neural network, we always use all the examples from D_p and D_c. However, we randomly sample a subset of examples from D_n at each episode of the training. By doing this,

Table C.2: Hyperparameters tuning

Hyperparameter	Values Attempted	Value Chosen
Adam learning rate	$10^{-5}, 10^{-4}, 10^{-3}$	10^{-3}
L2 regularization coefficient	$10^{-6}, 10^{-4}, 10^{-3}$	10^{-4}
Batch size	1024, 256	256

we ensure that the negative examples from D_n do not dominate the training set, and at the same time the "more certain" negative examples from D_c gets emphasized more than D_n. This whole procedure leads to an overall ratio between negative and positive samples around $3.5 : 1$ in each single batch.

Hyperparameters We ran a grid search over the hyperparameters of the ML model on an offline recommendation task. In the search process, we used the data from March 2018 to October 2019 (i.e. not including the data we test PROOF on), where the first 7/8 of the selected data are used as the training set and the last 1/8 are used as the validation set. We show the search result in Table C.2.

Bibliography

[1] Yasaman Abbasi, Debarun Kar, Nicole Sintov, Milind Tambe, Noam Ben-Asher, Don Morrison, and Cleotilde Gonzalez. Know your adversary: Insights for a better adversarial behavioral model. In *CogSci*, 2016. 2.3

[2] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011. 8.2

[3] Jacob Abernethy, Alex Chojnacki, Arya Farahi, Eric Schwartz, and Jared Webb. Activeremediation: The search for lead pipes in flint, michigan. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 5–14. ACM, 2018. 1, 8.1

[4] Gediminas Adomavicius and YoungOk Kwon. Optimization-based approaches for maximizing aggregate recommendation diversity. *INFORMS Journal on Computing*, 26(2): 351–369, 2014. 5.2, 5.5.2

[5] Agari. *Email Security: Social Engineering Report*, 2016. 3.1

[6] Deepak Agarwal, Souvik Ghosh, Kai Wei, and Siyu You. Budget pacing for targeted online advertisements at linkedin. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1613–1619, 2014. 5.2

[7] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 1–7, 2006. 5.2

[8] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014. 5.2

[9] Massimiliano Albanese, Ermanno Battista, and Sushil Jajodia. Deceiving attackers by creating a virtual attack surface. In *Cyber Deception*. 2016. 1.1, 2.1, 2.2, 3.2

[10] Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online fair division: analysing a food bank problem. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2540–2546, 2015. 5.2

[11] Tom M Apostol. Introduction to analytic number theory. 1966. 8.4.1

[12] Susan Athey and Stefan Wager. Efficient policy learning. *arXiv preprint arXiv:1702.02896*, 2017. 8.2

[13] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of*

Machine Learning Research, 3(Nov):397–422, 2002. 8.2

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[15] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The power of optimization from samples. In *NIPS*, pages 4017–4025, 2016. 8.2

[16] Gah-Yi Ban and Cynthia Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019. 8.2

[17] Guillaume W Basse, Hossein Azari Soufiani, and Diane Lambert. Randomization and the pernicious effects of limited budgets on auction experiments. In *Artificial Intelligence and Statistics*, pages 1412–1420. PMLR, 2016. 6.2.3

[18] Bettina Berendt. Ai for the common good?! pitfalls, challenges, and ethics pen-testing. *Paladyn, Journal of Behavioral Robotics*, 10(1):44–65, 2019. 9.2

[19] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020. 5.7, 8.1, 8.2, C.1, C.1.1

[20] Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. In *NIPS*, 2014. 2.2

[21] Robert Boutilier and Kyle Bahr. A natural language processing approach to social license management. *Sustainability*, 12, 10 2020. doi: 10.3390/su12208441. 7.2

[22] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018. 1

[23] Erik Brynjolfsson, Yu Hu, and Duncan Simester. Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Science*, 57 (8):1373–1386, 2011. 5.2

[24] Alberto Caprara, Margarida Carvalho, Andrea Lodi, and Gerhard J Woeginger. Bilevel knapsack with interdiction constraints. *INFORMS Journal on Computing*, 2016. 7

[25] Lijie Chen, Anupam Gupta, Jian Li, Mingda Qiao, and Ruosong Wang. Nearly optimal sampling algorithms for combinatorial pure exploration. In *Conference on Learning Theory*, pages 482–534. PMLR, 2017. 8.2

[26] Mia Xu Chen, Benjamin N. Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, and et al. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2287–2295, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330723. URL `https://doi.org/10.1145/3292500.3330723`. 1

[27] T. Chen, Linpeng Tang, Q. Liu, Diyi Yang, Saining Xie, Xuezhi Cao, C. Wu, E. Yao, Zhengyang Liu, Z. Jiang, C. Chen, Weihao Kong, and Yingrui Yu. Combining factorization model and additive forest for collaborative followee recommendation. 2012. 5.2

[28] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *The Journal of Machine Learning*

Research, 13(1):3619–3622, 2012. 5.2

[29] Hao Cheng, Zhaowei Zhu, Xingyu Li, Yifei Gong, Xing Sun, and Yang Liu. Learning with instance-dependent label noise: A sample sieve approach. In *ICLR*, 2021. URL https://openreview.net/forum?id=2VXyy9mIyU3. 7.1, 7.5.3

[30] Cho-Yu J Chiang, Yitzchak M Gottlieb, Shridatt James Sugrim, Ritu Chadha, Constantin Serban, Alex Poylisher, Lisa M Marvel, and Jonathan Santos. Acyds: An adaptive cyber deception system. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pages 800–805. IEEE, 2016. 2.3

[31] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 815–824, 2016. 5.2

[32] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011. 8.2

[33] Cisco. *Cisco Umbrella Popularity List*, 2019. 3.7

[34] Alisha Coleman-Jensen, Matthew P Rabbitt, Christian A Gregory, and Anita Singh. Household food security in the united states in 2017. *USDA-ERS Economic Research Report*, 2018. 4.1

[35] Alisha Coleman-Jensen, Matthew P Rabbitt, Christian A Gregory, and Anita Singh. Household food security in the united states in 2019. *USDA-ERS Economic Research Report*, 2020. 5.1

[36] Zach Conrad, Meredith T Niles, Deborah A Neher, Eric D Roy, Nicole E Tichenor, and Lisa Jahns. Relationship between food waste, diet quality, and environmental sustainability. *PloS one*, 13(4):e0195405, 2018. 4.1

[37] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198. ACM, 2016. 1

[38] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory*, 2008. 8.2, 15, 8.4.3, 15, 22, 8.5.1

[39] Rajeev H Dehejia and Sadek Wahba. Propensity score-matching methods for nonexperimental causal studies. *Review of Economics and statistics*, 2002. 4.2

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423. 7.5.1, 3

[41] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011. 8.2

[42] Karel Durkota, Viliam Lisỳ, Branislav Bosanskỳ, and Christopher Kiekintveld. Optimal network security hardening using attack graph games. In *IJCAI*, 2015. 3.2

[43] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288, 2015. 5.1

[44] Adam N Elmachtoub and Paul Grigas. Smart" predict, then optimize". *arXiv preprint arXiv:1710.08005*, 2017. 5.7, 8.1, 8.2, 8.4.1

[45] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, 2015. 2.2

[46] Fei Fang, Thanh H Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, and Andrew Lemieux. Deploying paws: Field optimization of the protection assistant for wildlife security. In *Twenty-Eighth IAAI Conference*, 2016. 1

[47] David Farquhar. *Watering hole attack prevention*, 2017. 3.1

[48] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 33–44. ACM, 2012. 4.1, 5.1

[49] Daniel Fleder and Kartik Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5):697–712, 2009. 5.2

[50] Jiarui Gan, Haifeng Xu, Qingyu Guo, Long Tran-Thanh, Zinovi Rabinovich, and Michael Wooldridge. Imitative follower deception in stackelberg games. In *EC*, 2019. 2.2

[51] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *2010 IEEE International Conference on Data Mining*, pages 176–185. IEEE, 2010. 5.2

[52] Shahrzad Gholami, Amulya Yadav, Long Tran-Thanh, Bistra Dilkina, and Milind Tambe. Don't put all your strategies in one basket: Playing green security games with imperfect prior knowledge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 395–403. International Foundation for Autonomous Agents and Multiagent Systems, 2019. 8.1

[53] Robert Gibb. How consumers perceive push notification in 2018, 2018. 4.6

[54] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961. 3.5.2

[55] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, volume 8, pages 982–991, 2008. 5.2

[56] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pages 297–299. PMLR, 2018. C.1

[57] Mark P Graus and Martijn C Willemsen. Improving the user experience during cold start through choice-based preference elicitation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 273–276, 2015. 5.2

[58] Dana Gunders and Jonathan Bloom. Wasted: How america is losing up to 40 percent of its food from farm to fork to landfill. 2017. 5.1

[59] Canan Gunes, Willem-Jan van Hoeve, and Sridhar Tayur. Vehicle routing for food rescue programs: A comparison of different approaches. In *CPAIOR*, 2010. 4.2, 5.2

[60] Qingyu Guo, Bo An, Branislav Bosanskỳ, and Christopher Kiekintveld. Comparing strategic secrecy and stackelberg commitment in security games. In *IJCAI*, pages 3691–3699, 2017. 2.2

[61] Nika Haghtalab, Fei Fang, Thanh H Nguyen, Arunesh Sinha, Ariel D Procaccia, and Milind Tambe. Three strategies to success: Learning adversary models in security games. In *IJCAI*, 2016. 2.2, 2.4, 2.4.2, 8

[62] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017. 5.1, 5.2

[63] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. 2012. 2.4

[64] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, 2012. 4.2

[65] Nam Ho-Nguyen and Fatma Kılınç-Karzan. Risk guarantees for end-to-end prediction and optimization processes. 2019. 8.2

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2

[67] Karel Horák, Quanyan Zhu, and Branislav Bošanskỳ. Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *GameSec*, 2017. 1.1, 2.1, 2.2

[68] Kasra Hosseini and Mariona Coll Ardanuy. Data study group final report: Wwf, June 2020. URL `https://doi.org/10.5281/zenodo.3878457`. 7.2, 7.4

[69] George Hurlburt. "good enough" security: The best we'll ever have. *Computer*, 2016. 1.1, 2.1

[70] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012. 2.3

[71] Sushil Jajodia, VS Subrahmanian, Vipin Swarup, and Cliff Wang. *Cyber deception*. Springer, 2016. 1.1, 2.1

[72] Sushil Jajodia, Noseong Park, Fabio Pierazzi, Andrea Pugliese, Edoardo Serra, Gerardo I Simari, and VS Subrahmanian. A probabilistic logic of cyber deception. *IEEE Trans. Inf. Forensics Secur.*, 12(11), 2017. 1.1, 2.1, 2.2, 3.2

[73] Kalyani Joshi, Bharathi N, and Jyothi Rao. Stock trend prediction using news sentiment analysis. *International Journal of Computer Science and Information Technology*, 8:67–76, 06 2016. doi: 10.5121/ijcsit.2016.8306. 7.2

[74] Yi-hao Kao, Benjamin V Roy, and Xiang Yan. Directed regression. In *Advances in Neural Information Processing Systems*, pages 889–897, 2009. 8.2

[75] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, 1990. 4.2

[76] Sedrick Scott Keh, Zheyuan Ryan Shi, David J Patterson, Nirmal Bhagabati, Karun Dewan, Areendran Gopala, Pablo Izquierdo, Debojyoti Mallick, Ambika Sharma, Pooja Shrestha, et al. Newspanda: Media monitoring for timely conservation action. In *IAAI*, 2023. 1.2

[77] H. Kellerer, H.K.U.P.D. Pisinger, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Nature Book Archives Millennium. Springer, 2004. ISBN 9783540402862. URL `https://books.google.com/books?id=u5DB7gck08YC`. 8

[78] David Laborde, Will Martin, Johan Swinnen, and Rob Vos. Covid-19 risks to global food security. *Science*, 369(6503):500–502, 2020. 5.1

[79] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985. 8.2

[80] Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Optimal personalized filtering against spear-phishing attacks. In *AAAI*, 2015. 3.2

[81] Jon Latimer. *Deception in War*. John Murray, 2001. 2.1

[82] Kuang-Chih Lee, Ali Jalali, and Ali Dasdan. Real time bid optimization with smooth budget delivery in online advertising. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, pages 1–9, 2013. 5.2

[83] Min Kyung Lee, Daniel Kusbit, Anson Kahng, Ji Tae Kim, Xinran Yuan, Allissa Chan, Daniel See, Ritesh Noothigattu, Siheon Lee, Alexandros Psomas, and Ariel D. Procaccia. Webuildai: Participatory framework for algorithmic governance. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019. ISSN 2573-0142. doi: 10.1145/3359283. URL `http://doi.acm.org/10.1145/3359283`. 4.2, 5.2

[84] Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *SAGT*, 2009. 2.2

[85] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010. 8.2

[86] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 539–548, 2016. 5.2

[87] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pages 689–698, 2018. 5.1

[88] Leo Liberti, Sonia Cafieri, and Fabien Tarissan. Reformulations in mathematical programming: A computational approach. In *Foundations of Computational Intelligence Volume 3*, pages 153–234. Springer, 2009. 8.5.2

[89] Matt Lisivick. Newsapi python library. https://github.com/mattlisiv/newsapi-python, 2018. Accessed: 2022-12-12. 1

[90] Min Liu, Jialiang Mao, and Kang Kang. Trustworthy and powerful online marketplace experimentation with budget-split design. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 3319–3329, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467193. URL https://doi.org/10.1145/3447548.3467193. 6.2.3

[91] Yang Liu and Hongyi Guo. Peer loss functions: Learning from noisy labels without knowing noise rates. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 7.5.3

[92] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. 3

[93] Steven Loria. textblob documentation. *Release 0.15*, 2, 2018. 7.5.1

[94] Sasha Luccioni, Emi Baylor, and Nicolas Duchene. Analyzing sustainability reports using natural language processing. In *NeurIPS 2020 Workshop on Tackling Climate Change with Machine Learning*, 2020. URL https://www.climatechange.ai/papers/neurips2020/31. 7.2

[95] Taylor Lundy, Alexander Wei, Hu Fu, Scott Duke Kominers, and Kevin Leyton-Brown. Allocation for social good: auditing mechanisms for utility maximization. In *ACM EC*, 2019. 4.2, 5.2

[96] Shana Lynch. Andrew ng: Why ai is the new electricity, 2017. https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity. 1

[97] Gordon Fyodor Lyon. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009. 2.1

[98] Vahideh Manshadi and Scott Rodilitz. Online policies for efficient volunteer crowdsourcing. *arXiv preprint arXiv:2002.08474*, 2020. 5.2

[99] Janusz Marecki, Gerry Tesauro, and Richard Segal. Playing repeated stackelberg games with unknown opponents. In *AAMAS*, 2012. 2.2

[100] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007. 5.2

[101] Aranyak Mehta, Bo Waggoner, and Morteza Zadimoghaddam. Online stochastic matching with unequal probabilities. In *Proceedings of the twenty-sixth ACM-SIAM symposium on Discrete algorithms*, 2015. 4.2

[102] Kevin D Mitnick and William L Simon. *The art of deception*. John Wiley & Sons, 2011. 3.1

[103] Laura Clark Murray, Nikhel Gupta, Joanne Burke, Rishika Rupam, and Zaheeda Tshankie. Matching land conflict events to government policies via machine learning models. https://omdena.com/wp-content/uploads/2019/12/Omdena-Land-Conflicts-Challenge-1.pdf, 2020. Accessed: 2022-12-12. 7.2

[104] Ibrahim Muter and Tevfik Aytekin. Incorporating aggregate diversity in recommender systems using scalable optimization approaches. *INFORMS Journal on Computing*, 29(3): 405–421, 2017. 5.2

[105] Divya J Nair, Taha Hossein Rashidi, and Vinayak V Dixit. Estimating surplus food supply for food rescue and delivery operations. 2017. 5.2

[106] Divya Jayakumar Nair, Hanna Grzybowska, David Rey, and Vinayak Dixit. Food rescue and delivery: Heuristic algorithm for periodic unpaired pickup and delivery vehicle routing problem. *Transportation Research Record*, 2548(1):81–89, 2016. 5.2

[107] DJ Nair, H Grzybowska, Y Fu, and VV Dixit. Scheduling and routing models for food rescue and delivery operations. *Socio-Economic Planning Sciences*, 63:18–32, 2018. 4.1, 4.2

[108] Ellen Nakashima. *To thwart hackers, firms salting their servers with fake data*, 2013. http://articles.washingtonpost.com/2013-01-02/world/362116541hackers-servers-contract-negotiations. 2.1

[109] Preetam Nandy, Divya Venugopalan, Chun Lo, and Shaunak Chatterjee. A/b testing for recommender systems in a two-sided marketplace. *Advances in Neural Information Processing Systems*, 34:6466–6477, 2021. 6.2.3

[110] Thanh Hong Nguyen, Yongzhao Wang, Arunesh Sinha, and Michael P. Wellman. Deception in finitely repeated security games. In *AAAI*, 2019. 2.2

[111] Katja Niemann and Martin Wolpers. A new collaborative filtering approach for increasing the aggregate diversity of recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 955–963, 2013. 5.2

[112] Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003. 3.5.1

[113] Bolanle Ojokoh. Automated online news content extraction. *International Journal of Computer Science Research and Application*, 2:2–12, 01 2012. 7.2

[114] Parliament. *Watering Hole Attacks*, 2018. 3.3

[115] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *AAMAS*, 2008. 2.7

[116] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 7.5.1

[117] Binghui Peng, Weiran Shen, Pingzhong Tang, and Song Zuo. Learning optimal strategies

to commit to. In *AAAI*, 2019. 2.2

[118] Andrew Perrault, Bryan Wilder, Eric Ewing, Aditya Mate, Bistra Dilkina, and Milind Tambe. Decision-focused learning of adversary behavior in security games. *arXiv preprint arXiv:1903.00958*, 2019. 2.2

[119] Andrew Perrault, Fei Fang, Arunesh Sinha, and Milind Tambe. Ai for social impact: Learning and planning in the data-to-deployment pipeline. *AI Magazine*, 41(4):3–16, 2020. 8.1

[120] Caleb Phillips, Rhonda Hoenigman, and Becky Higbee. Food redistribution as optimization. *arXiv preprint arXiv:1108.5768*, 2011. 4.2, 5.2

[121] Radek Píbil, Viliam Lisỳ, Christopher Kiekintveld, Branislav Bošanskỳ, and Michal Pěchouček. Game theoretic model of strategic honeypot selection in computer networks. In *GameSec*. Springer, 2012. 3.2

[122] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008. 1

[123] Bruce Potter and Greg Day. The effectiveness of anti-malware tools. *Computer Fraud & Security*, 2009. 1.1, 2.1

[124] Jean Pouget-Abadie, Kevin Aydin, Warren Schudy, Kay Brodersen, and Vahab Mirrokni. Variance reduction in bipartite experiments through correlation clustering. *Advances in Neural Information Processing Systems*, 32, 2019. 6.2.3

[125] Clara C Pratt, William M McGuigan, and Aphra R Katzev. Measuring program outcomes: Using retrospective pretest methodology. *American Journal of Evaluation*, 21(3):341–349, 2000. 4.2

[126] Canice Prendergast. The allocation of food to food banks. *EAI Endorsed Trans. Serious Games*, 3(10):e4, 2016. 4.1, 4.2, 5.2

[127] Niels Provos et al. A virtual honeypot framework. In *USENIX Security Symposium*, 2004. 2.3

[128] PwC. *Operation Cloud Hopper Technical Annex*, 2017. https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-annex-b-final.pdf. 2.3

[129] D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, page 502–511, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 158113844X. doi: 10.1145/988672.988740. URL https://doi.org/10.1145/988672.988740. 7.2

[130] Saharon Rosset and Ryan J Tibshirani. From fixed-x to random-x regression: Bias-variance decompositions, covariance penalties, and prediction error estimation. *Journal of the American Statistical Association*, 115(529):138–151, 2020. 8.4.1

[131] Neil C Rowe. Deception in defense of computer systems from cyber attack. In *Cyber Warfare and Cyber Terrorism*. IGI Global, 2007. 2.2

[132] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. Active learning in recommender systems. In *Recommender systems handbook*, pages 809–846. Springer, 2015. 5.2

[133] Bianca S Santos and Larry B Crowder. Online News Media Coverage of Sea Turtles and Their Conservation. *BioScience*, 71(3):305–313, 02 2021. ISSN 0006-3568. doi: 10.1093/biosci/biaa175. URL `https://doi.org/10.1093/biosci/biaa175`. 7.2

[134] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, 2002. 5.2

[135] Aaron Schlenker, Omkar Thakoor, Haifeng Xu, Fei Fang, Milind Tambe, Long Tran-Thanh, Phebe Vayanos, and Yevgeniy Vorobeychik. Deceiving cyber adversaries: A game theoretic approach. In *AAMAS*, 2018. 2.2, 3.2, A.2.2

[136] Zain Shamsi, Ankur Nandwani, Derek Leonard, and Dmitri Loguinov. Hershel: single-packet os fingerprinting. In *ACM SIGMETRICS Performance Evaluation Review*, 2014. **??**, 2.3

[137] Zheyuan Ryan Shi, Ziye Tang, Long Tran-Thanh, Rohit Singh, and Fei Fang. Designing the game to play: optimizing payoff structure in security games. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 512–518. AAAI Press, 2018. 7

[138] Zheyuan Ryan Shi, Ariel D Procaccia, Kevin S Chan, Sridhar Venkatesan, Noam Ben-Asher, Nandi O Leslie, Charles Kamhoua, and Fei Fang. Learning and planning in the feature deception problem. In *International Conference on Decision and Game Theory for Security*, pages 23–44. Springer, 2020. 1.2

[139] Zheyuan Ryan Shi, Aaron Schlenker, Brian Hay, Daniel Bittleston, Siyu Gao, Emily Peterson, John Trezza, and Fei Fang. Draining the water hole: Mitigating social engineering attacks with cybertweak. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13363–13368, 2020. 1.2

[140] Zheyuan Ryan Shi, Claire Wang, and Fei Fang. Artificial intelligence for social good: A survey. *arXiv preprint arXiv:2001.01818*, 2020. 1

[141] Zheyuan Ryan Shi, Zhiwei Steven Wu, Rayid Ghani, and Fei Fang. Bandit data-driven optimization: Ai for social good and beyond. *arXiv preprint arXiv:2008.11707*, 2020. 1.2

[142] Zheyuan Ryan Shi, Yiwen Yuan, Kimberly Lo, Leah Lizarondo, and Fei Fang. Improving efficiency of volunteer-based food rescue operations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(8):13369–13375, 2020. 1.2, 5.2, 5.3, 8.1

[143] Zheyuan Ryan Shi, Leah Lizarondo, and Fei Fang. A recommender system for crowdsourcing food rescue platforms. In *Proceedings of The Web Conference*, 2021. 1.2

[144] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai,

Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. 1

[145] Arunesh Sinha, Debarun Kar, and Milind Tambe. Learning adversary behavior in security games: A pac model perspective. In *AAMAS*, 2016. 2.2

[146] Lance Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 2003. 2.1, 2.4

[147] Ioan M Stancu-Minasian. *Fractional programming: theory, methods and applications*, volume 409. Springer Science & Business Media, 2012. 2.5

[148] Michael Sutton. *How to protect against watering hole attacks*, 2014. 3.1

[149] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research*, 16 (1):1731–1755, 2015. 8.2

[150] Symantec. *Attackers target dozens of global banks with new malware*, 2017. 3.3

[151] Liang Tang, Yexi Jiang, Lei Li, and Tao Li. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 73–80, 2014. 5.2

[152] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL https://aclanthology.org/W03-0419. 7.6.1

[153] Kentaro Toyama. *Geek heresy: Rescuing social change from the cult of technology*. PublicAffairs, 2015. 5.7, 6.4

[154] Johan Ugander, Brian Karrer, Lars Backstrom, and Jon Kleinberg. Graph cluster randomization: Network exposure to multiple universes. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–337, 2013. 6.2.3

[155] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015. 1

[156] Wei Wang and Bo Zeng. A two-stage deception game for network defense. In *GameSec*, 2018. 2.2

[157] Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1401–1408, 2019. 7

[158] Megan D Wolfson and Catherine Greeno. Savoring surplus: effects of food rescue on recipients. *Journal of Hunger & Environmental Nutrition*, 2018. 5.1

[159] D.H. Wolpert. Stacked generalization. *Neural networks*, 1992. 4.3.2

[160] Yinglin Wu, Ling Xie, Shiang-Lin Huang, Ping Li, Zengwei Yuan, and Wenhua Liu. Using

social media to strengthen public awareness of wildlife conservation. *Ocean & Coastal Management*, 153:76–83, 2018. 7.2

[161] Jian Xu, Kuang-chih Lee, Wentong Li, Hang Qi, and Quan Lu. Smart pacing for effective online ad campaign optimization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2217–2226, 2015. 5.2

[162] Lily Xu, Elizabeth Bondi, Fei Fang, Andrew Perrault, Kai Wang, and Milind Tambe. Dual-mandate patrols: Multi-armed bandits for green security. *arXiv preprint arXiv:2009.06560*, 2020. 8.2

[163] Amulya Yadav, Bryan Wilder, Eric Rice, Robin Petering, Jaih Craddock, Amanda Yoshioka-Maxwell, Mary Hemler, Laura Onasch-Vera, Milind Tambe, and Darlene Woo. Influence maximization in the field: The arduous journey from emerging to deployed application. In *Proceedings of the 16th conference on autonomous agents and multiagent systems*, pages 150–158. International Foundation for Autonomous Agents and Multiagent Systems, 2017. 1

[164] Rong Yang, Fernando Ordonez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *AAMAS*, 2012. 2.2

[165] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *AAMAS*, 2014. 2.2

[166] Yue Yin, Bo An, Yevgeniy Vorobeychik, and Jun Zhuang. Optimal deceptive strategies in security games: A preliminary study. In *AAAI Symposium on Applied Computational Game Theory*, 2014. 2.2

[167] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 123–130, 2008. 5.2

[168] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1411–1420, 2013. 5.2

[169] Shuran Zheng, Bo Waggoner, Yang Liu, and Yiling Chen. Active information acquisition for linear optimization. In *Uncertainty in artificial intelligence*, 2018. 8.2

[170] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32, 2005. 5.1, 5.2