Limitations of Data-Based Decision Making: An Multiparadigmic Investigation of Challenges Faced by Artificial Intelligence

by

Sarah Kinsey

A dissertation accepted and approved in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in Computer and Information Science

Dissertation Committee:

Thanh H. Nguyen, Chair

Daniel Lowd, Core Member

Thien Nguyen, Core Member

Brice Kuhl, Institutional Representative

University of Oregon

Fall 2024

DISSERTATION ABSTRACT

Sarah Kinsey

Doctor of Philosophy in Computer and Information Science

Title: Limitations of Data-Based Decision Making: An Multiparadigmic Investigation of Challenges Faced by Artificial Intelligence

As big data and computing capability continue to grow, an ever-increasing amount of artificial intelligence approaches are being deployed in the real world, across various domains. With real world deployments come additional complexities, challenges, and vulnerabilities, particularly concerning data reliance. Among others, these data related vulnerabilities include the potential for intelligently sabotaged data, and incomplete data. This work explores vulnerabilities from three perspectives: adversarial learning, game theory, and online reinforcement learning. First, we investigate whether a directly targeted end-to-end poisoning attack on a data-based decision making learning-planning model is feasible. Next, through the lens of security games, we investigate how a data-based decision maker can form a useful behavioral model, despite the observable data being maliciously manipulated. Lastly, we examine how, in a real world online RL setting, limited and sparse data can be overcome to build an effective data-based decision making model.

This dissertation includes previously published and coauthored material.

CURRICULUM VITAE

NAME OF AUTHOR: Sarah Kinsey

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
University of Idaho Moscow, ID, USA

DEGREES AWARDED:

(Anticipated) Doctor of Philosophy, Computer and Information Science, 2024, University of Oregon
Bachelor of Science, Computer Science, 2019, University of Idaho

AREAS OF SPECIAL INTEREST:

Data Science
Data-Based Decision Making
Artificial Intelligence

PROFESSIONAL EXPERIENCE:

Graduate Teaching Assistant, Foundations of Data Science, 2024
Graduate Research Assistant, University of Oregon, Advised by Prof. Thanh Nguyen, 2020-2024
Graduate Teaching Assistant, Introduction to Programming and Problem Solving, 2022
Graduate Teaching Assistant, Fluency with Information Technology, 2021-2022
Software Development Contractor, Vynyl LLC, 2015-2020

GRANTS, AWARDS AND HONORS:

National Merit Scholar, University of Idaho, 2015
First Year Fellowship, University of Oregon, 2019

PUBLICATIONS:

Sarah Kinsey, Jack Wolf, others. Building a Personalized Messaging System for Health Intervention in Underprivileged Regions Using Reinforcement Learning, *In IJCAI-23, Special Track on AI for Good*. August 2023.

Wong, Wai Tuck, **Sarah Kinsey**, R. Karunasena, Thanh H. Nguyen, Arunesh Sinha. Beyond NaN: Resiliency of Optimization Layers in the Face of Infeasibility, *in Proceedings of the AAAI Conference on Artificial Intelligence*. June 2023.

Sarah Kinsey, Wong Wai Tuck, Arunesh Sinha, Thanh H. Nguyen. An Exploration of Poisoning Attacks on Data-based Decision Making, *In GameSec-22: Proceedings of the 13th Conference on Decision and Game Theory for Security*. October 2022.

(as Ryan Butler) **Ryan Butler**, Wong Wai Tuck, Arunesh Sinha, Thanh H. Nguyen. Poisoning Attacks on Data-based Decision Making: A Preliminary Study, *In AASG-22: 3rd Autonomous Agents for Social Good held at the 21st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. May 2022.

(as Ryan Butler) **Ryan Butler**, Arunesh Sinha, Thanh H. Nguyen. Countering Attacker Data Manipulation in Security Games, *In GameSec-21: Proceedings of the 12th Conference on Decision and Game Theory for Security (GameSec)*. October 2021.

(as Andrew Butler) Thanh H. Nguyen, **Andrew Butler**, Haifeng Xu. Countering Attacker Data Manipulation in Security Games, *In ECAI-20: Proc. of the 24th European Conference on Artificial Intelligence (ECAI)*. September 2020.

ACKNOWLEDGEMENTS

Many thanks to my supportive advisor, Dr. Thanh H. Nguyen, for her patience and understanding throughout my graduate studies. Without good advisorship I would not have made it through this program, and I am very grateful to be where I am today. I also extend my gratitude to my other committee members: Dr. Daniel Lowd, Dr. Thien Nguyen, and Dr. Brice Kuhl on my dissertation committee, and Dr. Lei Jiao for his service on my committee in previous years.

I would also like to thank the undergraduate students that I've had the opportunity to teach. Teaching was a surprising source of joy and motivation for me, and has helped me greatly in developing both technical and personal skills.

My wonderful mother who encouraged me to prioritize my education, and sacrificed much of her time and energy to get me an early headstart on math and science. My close friends who have had the patience to stick with me through so many rough patches. Finally, I thank my beautiful partner, Kali, for their support throughout the final phase of my degree.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

As computing capability grows, and data reigns supreme, artificial intelligence has been applied to solve problems in a variety of domains. These include security, conservation, public health, and city planning. In all of these domains, additional challenges (such as imperfect data and deceptive behavior) arise when considering real-world deployments. In this work, we examine a paradigm that we refer to as data-based decision making. At its core, this paradigm involves taking in a set of data, performing some intermediate work on it to get *another* set of data that is, crucially, unavailable at test or deployment time. Then comes the ultimate goal of the system: making decisions based on this processed, unobservable at test time, data.

While this approach is powerful, and generalizable, it is far from bulletproof. The very data that makes a model valuable can also be its greatest vulnerability. In this work, we study two key weaknesses of data-based decision makers. First, we look into the possibility of a bad actor intentionally manipulating a model's available data, altering its learning outcome. We investigate this both through adversarial learning, and through a more game theoretic approach. The second weakness we explore is more organic, namely, missing data. More specifically, we look into a case where an online reinforcement learning model must make decisions based on human responses, which may be missing entirely.

1.1 Research Questions

Our first vector of investigation is to explore an attack based on adversarial learning research in the field of *decision focused* learning. This field studies approaches to data-based decision making models, making it of particular interest

to us. The traditional approach is simple: train a model to maximize prediction accuracy. However, in real world settings, it's inevitable that predictions will not be perfect. Thus, previous work [146, 28, 147] has investigated incorporating the end goal (high quality decisions) directly into the model's training process. This approach is what we refer to as decision focused learning. As this field involves using AI to make decisions based on real-world data, it's natural to examine the data itself as a source of vulnerability [62] but adversarial research in this area is underexplored, especially adversarial research that explicitly targets the resulting decisions of such models.

The first research question asked in this work seeks to address that gap in the literature. Namely, **is it feasible to attack data-based decision making models directly?** More specifically, we investigate whether an attacker can devise a poisoning attack, directly targeting a downstream goal (i.e. altering decisions made at test time to further its own ends). We explore multiple methods of creating this attack. First, we seek to directly attack a decision focused model. Next, we attempt an attack on the more traditional method for solving these problems, which we call a two-stage model. This approach trains a model solely to maximize predictive power, then incorporates the decision making component separately. We then test if such an attack can be transferred effectively to the decision focused model. Lastly, we devise an attack against a simple end-to-end proxy model (which seeks to directly make decisions from the observable data without explicitly including a solver for optimization problems) and investigate the transfer potential of *this* attack.

As security games model interactions between adversaries, it is natural to consider the vulnerabilities of data based decision making models from this

perspective. Deceptive behavior on both the attacker and defender sides have been explored in this field. Most well established is deception from the defender side [110, 175, 45], for example, concealing the allocation of defensive resources. More recently, a lot of work has been done investigating deception from the attacker side [90, 86, 33, 166], such as acting deceptively during a data collection phase so that the defender will create an exploitable strategy. Naturally, defenses against such deception (which must rely on the limited data available to the defender) have also been studied, albeit in a more limited capacity.

In the security game setting we consider [88], there is a single defender and two forms of attacker present. First is a population of non-deceptive attackers who are not considering future refinement to the defender's strategy and are, in effect, playing honestly. Secondly, there is a deceptive attacker present who is aware that the defender collects information to refine its strategy, and is thus seeking to manipulate the learning result.

Furthermore, we consider this to be a game with two phases: a (relatively) short learning phase, followed by a planning phase. The learning phase is when the defender is actively collecting data to form a model of attacker behavior, which it will then use to refine its defense in the planning phase. After which, it will commit to its strategy for an arbitrary long period of time. This motivates the deceptive attacker to alter its play during the learning phase in order to gain advantage against the strategy resulting from the planning phase. However, it has limitations on its power: it can only make a limited number of attacks, and cannot alter the attacks of the non-deceptive attackers. The defender, then, knowing that a deceptive attacker exists, must still deduce the behavior of the non-deceptive attackers to form its defense.

This brings us to the second research question to be explored in this work. **Given such a manipulated data set and some knowledge of the deceptive attacker's capability, how could a savvy defender model attacker behavior and formulate an effective defense?** Answering this question requires addressing its two components. First, the defender must understand the behavior of the *non*-deceptive attacker, despite the presence of deception. We accomplish this by finding the possible range of behavior that the *non*-deceptive attacker could have demonstrated in the learning phase, given the deceptive attacker's capability. The next component is simpler, merely requiring the defending to create a defense balanced against both the deceptive attacker (who will begin playing rationally to exploit the strategy it induced) and the non-deceptive attacker behavior, which was characterized previously.

Lastly, we turn to online reinforcement learning applications to investigate another perspective data-based decision making models. Namely, that the data relied upon is inherently limited, can be difficult to collect, and varies highly based on individual behavior. This makes it difficult to deploy AI applications that will be useful to a variety of users, and necessitates careful strategies to make effective use of the data that is available. Specifically, this kind of setting tends towards sparse rewards, as well as partially observable states. One approach for handling sparse rewards is return decomposition [113, 8], where rewards obtained over a trajectory are redistributed to the actions that lead to said rewards, even if that action and the eventual reward are separated by many timesteps. For addressing the challenge of partial observability, the standard approach is to include historical state/action data as input to the model that will decide on actions.

The combination of these challenges motivates our third and final research question. **How can we improve a reinforcement learning data-based decision making model to effectively utilize highly limited and incomplete real-world data?** To answer this, we use a combination of methods. Firstly, we explore creating simulated data using a generative model trained on the real user data. This allows us to test future models by simulating responses. Next, seek to devise a reinforcement learning approach that can deal with both sparse rewards and partial observability. We explore a variety of approaches here, including a predictive model to handle the partial observability hurdle, and a reward redistribution model to address the sparse rewards present in our setting.

1.2 Dissertation Statement

In this work, we explore concerns surrounding data-based decision making models, namely vulnerability to adversarial manipulation and the challenge of effectively utilizing incomplete data. In the field of decision focused learning, we explore poisoning attacks against the traditional two-stage models, the direct end-to-end approach, and the more recently proposed decision focused models. Next, we investigate a game theoretic perspective through the field of security games, which models interactions between adversaries. Defenders often rely on attacker's past behavior to build defenses against them, meaning that savvy attackers could manipulate this data nefariously. Our work, then, is to determine how a wary defender could address this deception. Lastly, we explore a slightly different concern surrounding data-based decision making through the lens of online reinforcement learning in a public health setting. Here, we devise a new RL model to address the combination of challenges (namely reward sparsity and partial observability) inherent to this setting.

Co-authored and Published Work Disclaimer Parts of this dissertation (Chapters 3-5) are adapted from published papers and are a result of collaboration with co-authors: Jack Wolf, Wong Wai Tuck, Professor Arunesh Sinha, Professor Thanh H. Nguyen, Nalini Saligram, Varun Ramesan, Meeta Walavalkar, and Nidhi Jaswal. More detail on the breakdown of work for those publications will be given at the start of each chapter.

CHAPTER II

BACKGROUND KNOWLEDGE

Before investigating the diverse areas of artificial intelligence, we provide some background knowledge surrounding each of the fields we are concerned with. Primarily, we investigate data-based decision making, security games, and adversarial learning, as these are the paradigms we study in this work. Additionally, we briefly discuss the areas of reinforcement learning, meta-learning, and knowledge tracing, to provide context for methods we utilize in later chapters.

Adversarial learning, which investigates attacks on machine learning models as well as defenses, is a valuable area for us to study. Through this field, we gain insight into the ways that nefarious actors may target data-based decision makers. Of particular interest to us is graph adversarial learning, as graph learning problems often fall into the data-based decision making paradigm (e.g. predicting links based on node information and then solving a problem such as bipartite matching on the predicted graph). However, deep learning in general and computer vision are more well researched from an adversarial perspective. Understanding existing adversarial learning research is key to applying these methods to data-based decision making applications, and being able to create robust approaches in this field.

This chapter will start by describing data-based decision making and detail existing applications, paying particular attention to the relatively new decision-focused approach as well as some social good applications that are adjacent to this area of research. Next, we will provide an overview of security games and discussing the current state of deception research in this field. Then, we investigate adversarial learning. We give special attention to poisoning attacks and graph

based adversarial learning, as those are of particular relevance to both security game deception and attacks on data-based decision making. Finally, we cover the miscellaneous knowledge required to understand the context of each chapter.

2.1 Data-Based Decision Making

As the name implies, data-based decision making is an approach to problem solving that is reliant on real world data, with the ultimate goal of producing a decision. In this section, we first give an overview of the field, and then detail the two most common approaches (two-stage and decision focused) for solving the problem of interest. Afterwards, we discuss applications using these approaches, as well as various social good applications that don't fall neatly into either category.

Data-based decision making refers to a common paradigm in real world artificial intelligence applications in which we are concerned with three related pieces of information: directly observable data (denoted by u), data that will be unobservable at test time (denoted by θ), and a *decision* that must be made (denoted by x). The decision, x, depends directly on θ, which in turn can be predicted based on u. The ultimate goal in a data-based decision making problem is to find an optimal decision to maximize a utility function, abstractly represented as follows:

$$\max_{x \in X} f(x, \theta)$$

where x is the decision variable and $X \subseteq \mathbf{R}^K$ is the set of all feasible decisions. Note that the objective, f, depends directly on the *unobservable* parameter θ, which must be inferred from the correlated observable data, u.

There are two common ways of solving data based decision making problems. First, and most well established, is the two-stage approach. Here, the task is split into two separate steps. The predictive component (i.e. a neural

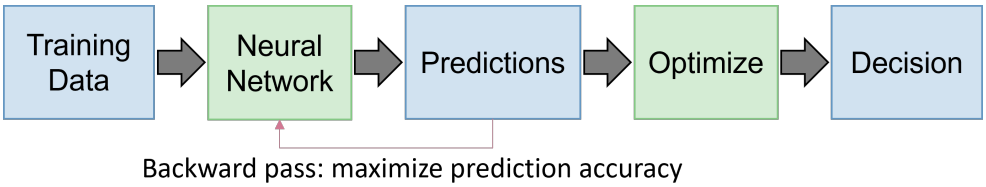


Figure 1. Depiction of a two-stage learner

network) is first trained directly to learn the relationship between θ and u, taking u as the input and outputting a prediction for θ, denoted $\hat{\theta}$. Next, we have the planning or optimization step, in which $\hat{\theta}$ is used to optimize the final decision, x.

While the two-stage approach would be optimal if we could perfectly predict θ from u, in realistic settings, errors are inevitable. Using imperfect predictions to optimize our decision may result in compounding errors, and notably worse decision quality. To address this shortcoming, an approach called *decision focused* learning seeks to bridge the disconnect between the end goal of the system and the learning result. That is, rather than training a model for predictive accuracy, it is directly trained to maximize decision quality.

The naive approach to this would be to have the network directly output x, and bypass prediction of θ entirely. However, in practice, training a neural network to solve optimization problems is a difficult task. Instead, decision focused learning approach still uses the model to predict θ. The innovation here, then, is to differentiate *through* the solution to an optimization problem, allowing the model to be trained based on the solution quality, while still incorporating a convex optimization solver [147].

2.1.1 Two-Stage Formulation. For the two-stage approach, the first stage is predicting the unknown parameter θ from the observed feature vector u. The second stage, then, is to compute the optimal x given the predicted θ

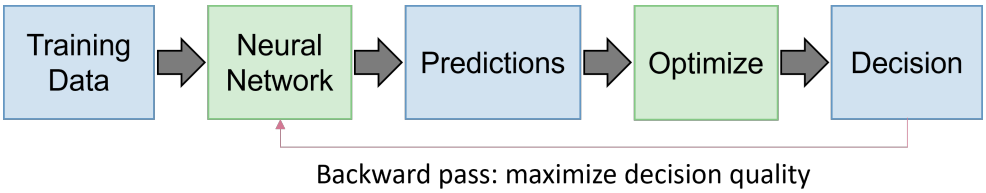


Figure 2. Depiction of a decision focused learner

(Figure 3). Predicting the *unknown* parameter θ can be done using a parametric model trained for the task, denoted by $\hat{\theta} = g(u, w)$. Here, w represents the model's parameters where the learner seeks an optimal set of model parameters, w^*, that minimize the training loss, abstractly formulated as follows:

$$\min_w \mathcal{L}(\mathcal{D}, w)$$

For example, using mean squared error as the training loss:

$$\mathcal{L}(\mathcal{D}, w) = \frac{1}{n} \sum_i (\theta_i - g(u_i, w))^2$$

Once the model has been trained (yielding w^*), the decision maker can use observed u values to predict a θ value ($\hat{\theta} = g(u, w^*)$), then use that prediction to find an optimal decision by solving the following optimization problem:

$$\max_{x \in X} f(x, g(u, w^*))$$

2.1.2 Decision Focused Formulation. We focus on the problem setting in which the decision optimization is convex, meaning that the objective is convex with respect to the decision variable, x. This convexity setting has been widely considered in previous studies on data-based decision making [147, 146, 28, 1].

Based on this convexity characteristic, we can leverage the implicit function theorem [65] to differentiate through the decision-optimization component (computing $\frac{dx}{d\hat{\theta}}$). The decision-optimization is formulated as a convex optimization

problem:

$$\max_{x} \; f(x, \hat{\theta}) \;\; \text{s.t.} \;\; Ax \leq b$$

Since this is a convex optimization problem, any solution that satisfies the following KKT conditions is optimal:

$$-\nabla_x f(x, \hat{\theta}) + \lambda \cdot \nabla_x (Ax - b) = 0$$

$$\lambda \cdot (Ax - b) = 0$$

$$Ax \leq b, \lambda \geq 0$$

where λ is the dual variable. Observe that the first equation indicates that x and λ are functions of $\hat{\theta}$. Based on the implicit function theorem, we can differentiate through the first two equations to obtain the following gradient formulation:

$$\begin{bmatrix} \frac{dx}{d\hat{\theta}} \\ \frac{d\lambda}{d\hat{\theta}} \end{bmatrix} = \begin{bmatrix} \nabla_x^2 f(x, \hat{\theta}) & A^T \\ diag(\lambda)A & diag(Ax - b) \end{bmatrix}^{-1} \begin{bmatrix} \frac{d\nabla_x f(x, \hat{\theta})}{d\hat{\theta}} \\ 0 \end{bmatrix} \tag{2.1}$$

Solving this system gives us $\frac{dx}{d\theta}$ which then allows us to directly optimize the predictive component of the model for decision quality using standard gradient descent based methods. The primary disadvantage of decision focused approaches is the increased computational cost. Every training instance requires both solving and backpropagating through this optimization, rather than computing the training loss based directly on the model output as in two-stage methods.

2.1.3 Two-Stage Applications. Data-based decision making encapsulates a wide range of applications and approaches. In this section, we'll begin by covering some works following the two-stage approach.

COPE [140] uses observed data to construct a convex hull containing expected future traffic demands. Then, they solve a linear program in order to

optimize traffic routing. Additionally, their method provides a worst-case guarantee for unprecedented or unpredictable future scenarios.

One application [154] uses a two-stage approach for reducing bias in citizen science (i.e. models built on crowd sourced data). More specifically, they consider a bird observation collection app (eBird) and seek to gamify observation collections so that contributors will provide more balanced data. The learning task here is to model the user's preferences (which determine how rewards scale, i.e. less preferred tasks require higher reward) based on observations of past behavior. Next, the decision-optimization component incorporates both the user's and the organizer's goals by transforming the user's goals into constraints on the organizer's objective. Finally, solving this optimization problem yields rewards that were shown to effectively incentivizes the users to explore under-observed areas, resulting in more balanced observations and lower bias.

In wildlife conservation, PAWS [30] uses a two-stage approach to protect wildlife and combat poachers. The predictive portion of the task is using past observations of poacher and animal activity to model the poachers' behavior using SUQR. Then, the decision-optimization task is to optimize ranger patrols using that model. Notably, this was the first deployed security game application considering imperfectly rational attackers as previous deployments assumed full rationality.

Another application [82] seeks to optimize allocation of emergency responders. For the learning task, they use features such as weather, season, and transportation network details to predict both the timing and severity of potential incidents requiring emergency services. Then, ask the decision-optimization component, they use a greedy approach to solve a non-linear, non-

convex optimization problem to yield desirable placements for emergency responder facilities.

Two-stage approaches also find application in graph based problems. One such work [157] considers community detection in the case where edge information is unknown. First, for the predictive task, they utilize edge prediction based on vertex similarity to learn the weights in a previously unweighted graph. Then, as the decision-optimization component, the authors use several standard community detection algorithms and compare results between them. Overall, this work demonstrates the value of making predictions in graph optimization tasks, rather than trying to make do with only the directly observable information.

2.1.4 Decision Focused Applications. While newer than the two-stage approach, a variety of works have considered decision focused approaches to solving data-based decision making problems. One [146] introduces a general formulation for decision focused combinatorial optimization problems, using linear programming and submodular maximization as examples. To bridge the gap between the optimization component and the model, the authors leverage the KKT conditions on the implicit function theorem, as described previously. Their experimental results across three different problems (budget allocation, bipartite matching, and diverse recommendations) demonstrate overall better solution quality than the two-stage approach, despite less accurate predictions from the predictive model. These results suggest that maximizing predictive accuracy is often a poor proxy for maximizing the final decision quality.

Similarly, another work investigates a decision focused approach for stochastic optimization [28]. Once again, they utilize the KKT conditions and the implicit function theorem to differentiate through the solution to an optimization

problem, using the derived gradient to train the predictive component. The author's experiments consider three different applications. These are a synthetic data inventory stock problem, and two real world applications: energy scheduling and battery load arbitrage. Their results demonstrate both higher utility and lower variance than the corresponding two-stage approaches.

Another paper [147] applies decision focused learning to graph optimization problems. The approach the authors consider starts with a graph embedding network that encodes the graph's adjacency matrix along with any available node information. Then, as the decision-optimization component, they incorporate a differentiable optimization layer that performs K-means clustering. This generalized approach can be seen as an analogous to many common graph problems, including maximum coverage and community detection. Solving the backwards pass uses the implicit function theorem to compute gradients. However, instead of using the KKT conditions of the optimization problem's solution, they directly characterize the optimization update process and compute gradients accordingly. Another contribution of this work is introducing a heuristic for this computation, significantly reducing the computational complexity of the backwards pass. Essentially, the authors find that, in practice, the K-means cluster assignments change little in each optimization step. When that holds, the gradient of the objective with respect to the cluster assignments can be approximated as the identity matrix.

Observing the computational complexity of decision-focused learning approaches, researchers are motivated to examine heuristics. One such work [141] investigates learning surrogates for decision-focused optimization problems, seeking to preserve the advantages of the decision-focused approach while

addressing the discouraging compute requirements. The authors utilize a *learnable* reparameterization matrix and incorporate it into the model. This allows for dramatic (but lossy) simplification of the decision-optimization problem, and allows loss based on the final solution quality to train both the predictive component and the reparameterization component. Another advantage of this surrogate approach is that it's less prone to getting stuck in local minima than both the decision-focused and two-stage approaches due to the gradient sparsity alleviating effects of the reparameterization. Experimentally, their results demonstrate the value of the surrogate approach, showing significantly lower runtime and/or significantly better solution quality than the decision-focused approach, and strictly better solution quality than the two-stage approach.

In the security game domain, researchers [103] leverage a decision focused approach to optimize defender utility. The predictive component in this setting is designed to learn the attacker's behavior (e.g. the target weights in SUQR). The decision-optimization component, then, is to optimize the defender's strategy accordingly. While the optimization problem here is generally non-convex, the local region is generally convex for boundedly rational attackers. This allows them to utilize the KKT conditions of the implicit function theorem to compute the gradient, enabling direct optimization of the predictive component based on solution quality. Lastly, their experiments show higher quality solutions across a variety of settings (including real-world human attacker data) than two-stage approaches.

In wildlife conservation, researchers [153] were motivated to investigate improving on PAWS by using a decision focused approach rather than the original two-stage approach. Furthermore, they account for uncertainty in observations of

poacher behavior by incorporating Gaussian processes into an ensemble learner, which allows them to quantify the uncertainty of observations in each section of a park. Leveraging this knowledge allows them to create more robust strategies, minimizing the harm done by imprecise observations. Experimentally, this end-to-end method increased detection of poaching by 30%, showing the value of decision focused approaches when observed data isn't fully reliable.

2.1.5 Social Good Applications. Though they may not perfectly fit into the "predict-then-optimize" framework, a variety of social good applications follow the general philosophy of data-based decision making. One such work [168] uses a large language model, RoBERTa [69] to automate triage of pregnant people with health concerns in Kenya. The model takes in questions sent over text message and attempts to classify their problem based on a set of pre-defined common concerns among pregnant people. If the predicted problem isn't severe, and the classification confidence is high, an automated response is sent. If either of these things are not true, the problem is referred to human health desk staff. The main challenge in this work lies in the text messages - they contain natural language including slang, and, to further complicate things, mixed English-Swahili text. Their final system shows high classification performance on problems of interest and is able to reduce the workload of the health desk workers.

Another work in maternal healthcare [74] also considers an automated messaging system. However, in this work, the goal is to optimize limited intervention resources to prevent dropouts. They use restless multi-armed bandits (RMABs), a reinforcement learning technique, to model the problem. The goal is to predict which participants will benefit most from an intervention, given their behavioral history. To deal with scaling issues and lack of data for new

participants, they cluster participants into groups and use a single RMAB for each group. Their results show that the selected interventions were significantly better than randomized interventions, highlighting the benefit of optimizing resources in similar health applications.

Also in public health, research [59] has investigated a decision focused approach for targeting interventions for improving adherence to tuberculosis treatment plans. This is accomplished via using various features (such as recent call data and demographic information) of the patients to predict which ones are likely to stop adhering to the treatment plan. The paper considers a random forest as well as an LSTM based model (which proves more effective), as the data is comprised of time series information for each participant. Furthermore, the authors investigate using the same models to predict the *effectiveness* of the interventions, which is where decision focused learning comes in (i.e. training a model directly to maximize the effectiveness of interventions selected, rather than just predicting what participants may need intervention). The decision-focused learning approach yields less accurate predictions, but results in 15% higher total intervention utility than the two-stage counterpart.

Similarly, research [156] has investigated using AI to optimize interventions among homeless youth to raise HIV awareness. This is done by forming a model of the community structure within the population of homeless youth, and selecting individuals who will most effectively spread information to others. Notably, the models formed of the community are never fully accurate, and uncertainty in some information (such as exactly how likely individuals are to pass on information to each other) has to be accounted for. The authors here perform a pilot study in the real world comparing two methods. First is HEALER [155] which leverages

Partially Observable Markov Decision Processes (POMDPs) to select optimal interventions. Notably, using a single POMPD for each possible combination of interventions results in an intractable problem. Thus, HEALER breaks the graph down, and uses multiple nested levels of POMDPs to overcome this challenge. The next method under evaluation is DOSIM [148], which uses a game theoretic approach with robust optimization. To make the problem tractable, the authors use the double oracle approach to find an approximate equilibrium. Notably, this method yields *mixed* strategies (which allows for more robust policy selection), while HEALER only gives pure strategies. In practice, both methods gave similar results, giving 160% more information spread compared to the baseline (degree centrality).

In AI for education, research often involves predicting student performance. One such work [127] uses a recurrent neural network (a modified LSTM) to perform this task, incorporating *both* the performance history of students as well as the text of the exercises in question. One of the key challenges in this area is known as the "cold start" problem, referring to the difficulty of predicting performance of new students or on new exercises. Their methods outperform baselines, particularly in the cold start setting, by incorporating correlations between exercises. While they don't consider any task *after* this predictive stage, their predictions could be used for objectives such as recommending tutoring, sending automated informative messages, and deciding what subjects should be covered in more depth.

2.2 Security Games

Our second area of interest lies in security games, which is primarily an area of study using game theory to optimize defensive resources in real-world security problems. We start by describing some real-world applications motivation further

interest in this field. Then, we discuss two key models (Stackelberg Security Games and their variant Green Security Games) to provide some context. Next, we discuss some common models of human behavior in these games, which are important when considering how deception can function. Lastly, we discuss the current state of deception research in this domain.

Game theoretic AI approaches have been shown effective in a variety of real world applications, particularly in security and wildlife conservation. One security application, ARMOR [106], was deployed to protect the Los Angeles International Airport. This is accomplished via modeling the interaction between adversaries such as terrorists or drug smugglers and airport security as a Stackelberg game, considering terminals and checkpoints as targets top be attacked. Solving this game using their method, DOBSS, yielded strategies that outperformed the existing human devised schedules, while still allowing for manual overrides within the scheduling system.

PROTECT [121] has been deployed by the US Coast Guard to optimize patrols and protect the ports of the United States. Their method models interactions between the Coast Guard and terrorists as a Stackelberg Security Game [132] with the Coast Guard as the defender and the terrorists as attackers. The targets considered are areas of interest in a port (e.g. critical infrastructure) which must be protected via Coast Guard patrols. To model the attacker's behavior, they utilize the Quantal Response model [75, 159] which allows for modeling of sub-optimal attacker behavior.

One challenge faced in this domain is the sheer number of targets, which results in an exponential number of potential patrols. To account for this, PROTECT first divides the port into patrol areas, and restricts patrols to covering

targets within a single area. Further, they reduce the final number of strategies considered (each strategy corresponding to a patrol allocation) via removing equivalent strategies as well as dominated strategies from the list, resulting in a more compact representation of the strategy space. Patrols created by their system resulted in more consistent coverage of targets as well as more proportional coverage (i.e. more valuable targets are patrolled more frequently).

In the conservation domain, PAWS [30] has been deployed in parks in both Uganda and Malaysia to combat poaching. Its approach divides the parks into grids and then models the rangers vs poachers dynamic as a Green Security Game [31], with rangers as the defender and poachers as the attackers. Each grid cell is considered a target, and the value of that target is determined by the animal density in that area. To model the attacker/poacher's behavior, the authors use Subjective Utility Quantal Response [91] with parameters learned from historical data. Solving the game yields patrol strategies that were proven effective.

Improving on this work, researchers [153] investigate an end-to-end approach for creating patrol strategies based on observed poacher data. To do so, they integrate Gaussian processes into an ensemble learner, quantifying the various levels of uncertainty in predictions across different sections of the park. Then, they use this uncertainty in order to build more robust patrol strategies. Experimentally, this method increased detection of poaching by 30%.

2.2.1 Stackelberg Security Games (SSGs). SSGs [132] consist of at least two players: a defender (the leader in traditional Stackelberg games) and one or more attackers (the followers). The defender's goal is to protect a set of T targets from these attackers, given a limited number of *resources* (K, where $K < T$) that each can be allocated to protect a single target. A defender's pure

strategy consists of a one-to-one allocation of resources to targets. A mixed defense strategy, $\mathbf{x}$, is a probability distribution over these pure strategies. This mixed strategy can be represented as a coverage probability vector: $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$, where $x_i \in [0, 1]$ represents the probability that target i is protected by the defender and $\sum_i x_i \leq K$. In SSGs, the attacker is fully aware of the defender's *mixed* strategy and chooses a target to attack based on this knowledge.

Suppose the attacker decides to attack target i. This action gives each player a reward or a penalty, depending on whether the defender is currently protecting target i. If i is unprotected, the attacker gains reward R_i^a and the defender receives penalty P_i^d. Conversely, if target i is protected, the attacker takes penalty $P_i^a < R_i^a$ and the defender gains reward $R_i^d > P_i^d$. Given coverage probability x_i, the *expected* utilities for the defender and the attacker resulting from an attack on target i can be formulated as follows:

$$U_i^d(x_i) = x_i R_i^d + (1 - x_i) P_i^d$$

$$U_i^a(x_i) = x_i P_i^a + (1 - x_i) R_i^a$$

2.2.1.1 Solution Concepts and Equilibrium. The standard solution concept for Stackelberg games is the Strong Stackelberg Equilibium (SSE). Note that this may be different from the Nash equilibrium as Stackelberg games are non-simultaneous. To be considered a SSE, a pair of attacker/defender strategies must satisfy three conditions:

– The defender's strategy, x, is the best response to the attacker's strategy, g:

$$U_d(x, g(x)) \geq U_d(x', g(x')) \forall x'$$

– The attacker's strategy, g, is the best response to the defender's strategy, x:

$$U_a(x, g(x)) \geq U_a(x, g'(x)) \forall g'(x)$$

– If any ties exist (strategies with equal expected utility for the attacker), the attacker breaks ties in favor of the defender:

$$U_d(x, g(x)) \geq U_d(x, g'(x)) \forall g'(x) \in T$$

Where T is the set of attacker strategies with equal expected attacker utility.

This equilibrium is guaranteed to exist [138]. The Weak Stackelberg Equilibrium (WSE), on the other hand, is not. This equilibrium concept is the same as the SSE with the third condition inverted. That is, the attacker breaks ties by choosing the *worst* strategy for the defender. As the WSE is not guaranteed to exist, the SSE is used as the standard solution, with the justification that the attacker can be induced to choose the best strategy for the defender by trivial adjustments to the defender's strategy.

2.2.2 Green Security Games. `GSG`s [31] define a specialized form of `SSG`s designed to be applicable to conservation problems. This model has two key differences: firstly, `GSG`s specifically focus on repeated game settings where there are multiple *rounds* of the game being played. In each round, there are multiple episodes. For the duration of a round, the defender commits to a mixed strategy, while each episode considers a single pure strategy drawn from this mixed strategy. As in `SSG`s, the attacker(s) commit to an attack based on their knowledge of the defender. The second key difference from `SSG`s in general is that the `GSG` attacker does not have a perfect knowledge of the defender's mixed strategy. Instead, each attacker is modeled with a memory length parameter, as well as a parameter controlling how much consideration is given to each historical timestep.

2.2.3 Human Behavior Modeling. While modeling attackers as perfectly rational is simple, real world adversaries don't conform to this assumption. Due to many factors including imperfect knowledge of the world

and human emotions, attackers are unlikely to choose their targets optimally. To address this, multiple approaches modeling human behavior have been used, including MATCH [107] and Quantal Response [75, 159].

MATCH uses robust optimization techniques to create defenses that can perform well against attackers of various behavior. Furthermore, it doesn't rely on any sort of attacker behavior modeling. Instead, it's singularly controlled by a parameter which dictates the tradeoff between defender utility when the attacker plays according to best response and robustness to less predictable attackers.

For our work, we focus on Quantal Response and its variants:

*2.2.3.1 Quantal Response (*QR*)..*

QR is an well-known model describing attacker behavior in SSGs [75, 159]. Intuitively, QR provides a mechanism for partially rational behavior where higher expected utility targets are attacked more frequently.

Essentially, the probability of attacking target i is given as follows:

$$q_i(\mathbf{x};\lambda) = \left(e^{\lambda U_i^a(x_i)}\right) \Big/ \left(\sum\nolimits_j e^{\lambda U_j^a(x_j)}\right) \tag{2.2}$$

This model describes the attacker with a single parameter, λ, governing its rationality. As λ approaches 0, the attacker becomes completely random. As λ approaches ∞, the attacker becomes perfectly rational.

Maximum Likelihood Estimation For computing λ, the traditional method is to use maximum likelihood estimation over the historical data. This yields the most likely λ matching the observed attack pattern:

$$\lambda^{learnt} = \texttt{argmax}_\lambda \sum_m \sum_i z_i^m \log q_i(x_i^m, \lambda)$$

where x_i^m is the defender's coverage probability of target i at timestep m and z_i^m is the observed number of attacks at that timestep. This allows the defender to learn

a single parameter, λ, using historical data. Then, the defender can predict future attacker behavior using that λ, and optimize its defense accordingly.

Defense Against `QR` Attacker To defend against such an attacker, `BRQR` (Best Response to Quantal Response) was proposed [159]. The defender's optimization problem is defined as follows:

$$\max_x \sum_i q_i U_i^d(x_i)$$
$$\text{s.t.} \sum_i x_i \leq T$$
$$0 \leq x_i \leq 1, \forall i$$

As this objective is generally non-convex, finding the global optimum isn't feasible. Instead, the general approach used is to find multiple local optima from different starting points, and to take the best one found [159].

2.2.3.2 Subjective Utility Quantal Response (`SUQR`)..

In `SUQR`, the attacker's perceived utility of attacking each target is calculated differently. Rather than computing the actual expected utility, `SUQR` uses a linear combination of some information available to the attacker. The attacker considers for each target the coverage probability, the reward for a successful attack, and the penalty for a defended attack:

$$\hat{U}_i^a = w_1 x_i + w_2 R_i^a + w_3 P_i^a$$

The attack probabilities, then, are given by:

$$q_i(\mathbf{x}; \lambda) = \left(e^{\lambda \hat{U}_i^a(x_i)}\right) \Big/ \left(\sum\nolimits_j e^{\lambda \hat{U}_j^a(x_j)}\right) \tag{2.3}$$

Intuitively, this allows attackers to give different weights to the defender coverage, the reward, and the penalty than the objective expected utility calculation does. This model was shown to outperform both regular `QR` as well as MATCH [91].

2.2.3.3 Other Models. Another model called CAPTURE [87] aims to improve upon the shortcomings of `SUQR`. Firstly, this model considers attacker behavior at each time step to be related to behavior at prior time steps, rather than independent as in `QR` models. Next, the model incorporates a larger range of domain features (e.g. slope and habitat) than `SUQR` does. Third, CAPTURE incorporates observational uncertainty on the part of the defender, which is modeled as depending on the domain features, the underlying behavior of the attackers, and the defense strategies during the observation. Lastly, attack probabilities are calculated independently per-target. Experimentally (using real world data), this model was shown to significantly out-perform `SUQR`.

Noting the complexity and poor interbretability of CAPTURE, researchers were motivated to create a simpler model, INTERCEPT [57]. Their underlying approach uses decision trees to produce effective and interpretable models. To handle the spatial challenges of the space (e.g. addressing the continuous nature of real world terrain), they draw on criminology's theory of "hot spots" which are points where crime (in this case poaching) is likely to be common. Then, they utilize the distance from expected hot spots as another input to the decision trees. Lastly, they utilized ensemble learning by creating different expert models (limited to 5 for interpretability reasons) that will then vote on the attack likelihood of each target. Experimentally, their model was shown to significantly outperform CAPTURE, despite being far simpler and computationally cheaper.

2.2.4 Game Theoretic Deception.

While modeling attacker behavior allows for a better defense, it does present a vulnerability. Namely, that the defender must utilize historical attack data to form a model of the attacker. If a particularly clever attacker were to change its attack pattern, knowing that data collection was in progress, it could alter the learning results and find advantage in the resulting strategy. Recently, research has investigated this kind of deception from the attacker side [34, 90, 166] in `SSGs`, and the follower side in general Stackelberg games [33]. This type of attack is analogous to a poisoning attack in adversarial learning.

One such work considers multiple *types* of attackers, corresponding to different rewards and penalties for each target. To deceive the defender, an attacker could then pretend to be a different type and play accordingly during the learning phase [90]. Then, after the defender has created its strategy, that attacker can play optimally, gaining advantage from the earlier deception's influence on the resulting strategy.

Addressing this imitative deception has also been studied [85]. This work introduces an exact equilibrium formulation for repeated SSGs, as well as using this formulation to devise an optimal counter to the aforementioned deception. However, the authors note that, given the repeated game setting, considering both historical data and future expected utility exponentially compounds this optimization problem. To address this, they introduce limited memory and limited lookahead heuristics. Their experimental results show that addressing the deception, with or without heuristics, yields significantly better utility for the defender, and worse utility for the attacker, than naively ignoring the deceptive behavior.

Another approach [88] considers a realistic scenario in which the defender must contend with multiple attackers of *unknown* behavior. These attackers are then modeled by the defender with `QR`, using a single λ to describe the attacker population. The deception, then, takes the form of an attacker playing according to some λ to skew the learning result for the entire population, altering the defender's resulting strategy. Again, after the learning phase, the attacker can play optimally to take advantage of the altered strategy.

Noticing the advantages of this form of deception, researchers were motivated to study counterstrategies [18]. Their approach relies on characterizing the possible deceptive space of the attacker and then using a maximin optimization to form an effective strategy against it. Using binary search, a defender can find both the minimum and the maximum possible λ parameter for the *non* deceptive attacker population (which was concealed by the deceptive attacker polluting the collected historical data). Then, the defender can optimize its strategy against both attackers (the deceptive, fully rational one and the boundedly rational population) using a maximin over the range found by the binary search.

One limitation of the two previously discussed deception approaches is that they only consider a one-shot game, where the attacker has no incentive to play dishonestly after the initial learning phase. A newer paper [86] explores a repeated game setting where the attacker must consider the longer term. The authors use projected gradient descent to solve the attacker's nested optimization problem and find its deception strategy. Their experimental results (on repeated games of 4 and 8 timesteps) show significantly higher utility for the attacker, and lower utility for the defender, compared to the case where the attacker plays honestly.

While studying attacker deception is relatively new, deception from the defender side has been more well considered [175]. One such work [175] described information concealment by the defender. Their setting was a multiple timestep, general sum game in which the defender could invest more resources between timesteps, and the attacker could learn more information based on observing signals and on results of attacks. Their findings showed that it can be in the best interest of the defender to conceal information (e.g. leading attackers to believe that targets are better defended than they actually are).

Similarly, a study shows selectively revealing information can improve outcomes for the defender [110]. The authors consider a Stackelberg security game setting in which allocation of defender resources to targets may not be visible to the attacker. They then investigate what resource assignments the defender should reveal to the attacker, finding that this selective disclosure can be a powerful deterrent, improving outcomes for the defender. Furthermore, they note that acting on this information (updating their strategy) will still be in the attacker's best interest, even if they know that it was intentionally revealed as a deterrent.

Another work [45] compared the utility of *signaling* (openly flaunting defense resources) to concealment, showing that there they can both be advantageous depending on the payoff structure of the game itself. The authors are able to formalize the tradeoff between concealment and signalling/commitment. Furthermore, their results show that the boundary of this tradeoff is close to zero sum.

2.3 Adversarial Learning

Our work has considered attacks in both game theoretic and data-based decision making settings. Specifically, we've investigated attack scenarios where

an adversary can manipulate some portion of the training data. In the field of adversarial learning, this would be considered a *poisoning attack* (or backdoor attack). By way of contrast, an *evasion attack* (or adversarial example) occurs at test time, seeking to manipulate the model's output for specific samples. *Exploratory attacks* work in another direction entirely, using their attack capabilities to learn more details about the system. Here, we pay extra attention to poisoning attacks as they are the most relevant to our work.

As graph learning problems often follow the data based decision making paradigm (e.g. using node features to predict edges and then performing bipartite matching on the edge predictions) we spend more time on this domain than others. After discussing poisoning attacks and evasion attacks in graph learning and deep learning, we detail the current research into defense and robustness.

Direct attacks to data-based decision making models are relatively unexplored. To the best of our knowledge, our work [62] is the only paper in this domain. We utilize the metagradient method to optimize poisoning attacks against data-based decision making models, investigating both the two-stage approach and the decision focused approach as targets. Furthermore, we evaluate the effectiveness of using a simpler model (i.e. one trained to directly output the decision) as a vector for generating attacks that will then be transferred. Experimentally, our results are mixed. Directly attacking the decision-focused learner is infeasible due to the computational requirements of solving the attacker's optimization problem. Attacks from the two-stage learner do transfer effectively to the decision focused learner, though generating these attacks is still difficult. Due to the complexity and non-convexity of the attack space, obtaining the global optima is implausible, and even finding a good local optima isn't guaranteed. Attacking the simpler model,

on the other hand, is entirely ineffective. Future work in this area should consider approximate metagradients or non metagradient based methods for attacking data-based decision making model.

2.3.1 Attacking Graph Learning. Problems across many domains including social networks, city planning, network security, and biology can be modeled as graphs. This has led to significant study of graph learning in recent years, particularly using deep learning on graphs [63, 13, 64, 79]. Anomaly detection in this field is well studied [3] based on the observation that learning results on the entire graph can be compromised via anomalous individual nodes. However, until more recently, intentional attacks on graph learning problems was an unexplored area of research.

2.3.1.1 Evasion Attacks. One early work in this area investigates adversarial example generation for the link prediction task [76]. In their setting, an adversary generates examples maximizing the *inconsistency loss*. This loss is calculated by first identifying constraints on non-adversarial inputs, and then measuring how much a given example violates those constraints. The learner (or discriminator) then makes use of this inconsistency loss as a regularization term when training on generated adversarial examples. Surprisingly, they are able to find efficient closed-form solutions for the adversarial generation task against several popular link prediction models. Their experimental results show that incorporating adversarial examples in this manner improves the performance of these link prediction models, particularly when limited training data exists.

Primarily motivated by network security problems, another work [23] investigates attacks on community detection tasks by an adversary without perfect knowledge. The authors introduce two different attacks: *targeted noise injection*

and *small community*. As in the name, targeted noise injection adds some noise to the graph structure, creating new edges in a way that imitates the structure of the true graph. The small community attack, on the other hand, aims to create smaller clusters in the graph by removing edges and/or nodes. For defences, the authors recommend re-training on adversarial examples (altered by the noise injection) and specifically tuning hyperparameters based on performance against the small community attack. Experimentally, they found the attack to dramatically reduce model performance when unaddressed, but that their suggested defences are effective.

In the domain of social networks, researchers have studied attacks on community detection problems. One work [142] considers an attack with the primary goal of obscuring the importance of a single individual (denoted $v^\dagger$) in a community (e.g. concealing the leader of a terrorist cell). The secondary goal of this attacker is to hide the community entirely. They also present simple (such that they could be used by attackers without mathematical or technical requirements) heuristics for both of these goals. For hiding individuals, ROAM (remove one, add many) removes the link between $v^\dagger$ and some v_0, then connects v_0 to up to $budget - 1$ other neighbors of $v^\dagger$. This reduces the closeness centrality of $v^\dagger$ and its degree, while increasing the closeness centrality and the degree of its chosen neighbor, v_0. For hiding communities, their DICE (disconnect internally, connect externally) heuristic first disconnects d (where $d \leq budget$) links within the community, and then creates $budget - d$ links from within the community to outside nodes. Note that this method is concerned with a single community/individual.

For a more global attack on community detection, work [20] utilize a genetic algorithm to generate adversarial examples. Their results show that their method

outperforms simpler heuristics they propose, Community Detection Attack (CDA) and Degree Based Attack (DBA). CDA randomly selects a node in each community to remove random inter-community links, and add intra-community links. DBA is identical, except instead of randomly selected nodes, it targets the highest degree node in each community. Furthermore, their results show significant transferability of their GA generated adversarial examples across different types of target models.

Many graph learning approaches use some kind of lower dimensional representation of nodes, done via some machine learning node embedding process such as DeepWalk [102], LINE [134], or node2vec [44]. Then, these node embeddings can be used for a variety of downstream tasks. Researchers are thus motivated to investigate attacks on the node embeddings as general purpose adversarial examples. One work [129] targets node embeddings and uses link prediction as the downstream task of interest. Their approach makes use of the KKT conditions to differentiate through the node embedding process and then optimizes adversarial graph modifications via projected gradient descent. The authors consider two specific attacks: *integrity attack* which targets specific links and *availability attack* which seeks to maximize overall prediction errors. Both are accomplished by adding or moving edges. Their results show the effectiveness of their technique, even with a budget of relatively few edges. Once again, attacks generated by this method are shown to transfer effectively between different node embedding techniques.

2.3.1.2 Poisoning Attacks.

The first work to consider training time attacks on deep learning for graphs [176] targeted the node classification task. Their approach allows for both structural attacks (modifying edges) and feature attacks (modifying node features), and seeks to create unnoticeable perturbations

by preserving degree distribution and feature co-occurrence statistics (e.g. ensuring that features never seen together in the original graph don't appear together in the modified graph). To make the computations tractable, the authors target a *surrogate* model to produce their attack, and then transfer it to the final model. Experimental results demonstrate the effectiveness of this attack, transferring successfully to other semi-supervised graph learning methods, and, notably, to the unsupervised method DeepWalk [102].

By way of contrast, another work [14] directly targets unsupervised methods for node embedding. This setting presents additional challenges: no labels exist to exploit, and many unsupervised node embedding methods (such as those based on random walks) prevent direct gradient calculations. Instead, they utilize matrix perturbation theory [125] to efficiently approximate the loss function of DeepWalk [102] and compute their attack, which takes the form of added and removed edges. They consider three general attack types: first, the general attack seeking to maximize the node embedding loss; second, a targeted attack seeking to change the classification of a specific node; third, a targeted attack seeking to prevent link prediction between a set of node pairs. Experimentally, they show that their attacks are effective even when the allowed number of edges flipped is low. Furthermore, they once again demonstrate transferability of their attacks between a variety of models.

Investigating poisoning attacks on the node classification task, another work [177] leverages meta learning to *directly* solve the bilevel optimization underlying the poisoning attack. Essentially, this requires unrolling the training process of the classifier (each step of training itself being differentiable) and computing the gradient of the resulting weights with respect to the training

data. Rather than considering specific nodes, their goal is to decrease the overall accuracy of the classifier. Additionally, they provide memory efficient heuristics for the metagradient calculation. The experimental results provided demonstrate the effectiveness of the main method, as well as the heuristics, decreasing overall classification performance of the target models even with small perturbations in the training data.

A more recent paper [167] investigates attacks and defences for graph neural networks under the *label flipping* setting. Here, the attacker's power is limited to changing the labels of nodes (considered to be binary) in the training set. To solve this attack, the authors come up with a closed-form approximation for the classifier (a GCN here) as well as transforming the discrete components of the attack objective into continuous surrogates. This allows them to avoid directly computing the metagradient, as [177] did. For defence, they propose a self-supervised community labelling task as a regularization method during the training process. Their experiments on several real world datasets demonstrate the value of the attack as well as the effectiveness of their proposed defence.

Targeting classical methods for graph learning (rather than the relatively new methods considered in the previously mentioned works) researchers [68] seek to create a unified framework for poisoning attacks on semi-supervised graph learning problems, particularly focusing on the label propagation method. Their framework considers both classification tasks and regression tasks, and presents novel approaches for solving both. Experimental results demonstrate that, even with very few perturbations, their methods can significantly decrease classification accuracy or increase regression loss.

Two simultaneous works [170, 150] first considered *backdoor* attacks on graph neural networks. These are a special case of poisoning attack where the attacker seeks to influence the model to classify test time examples with some *trigger* present as a specific class. Furthermore, the attack is designed to not impact performance on clean test examples (those without the trigger present).

The first of these works [170] seeks to directly produce a graph neural network that is susceptible to these triggers, given a pre-trained clean GNN and the data that will be used for downstream classification (using the node embeddings produced by the GNN). Interestingly, they tailor the triggers (which take the form of subgraphs) to each graph in question, rather than using a one-size-fits-all approach. Their results show how effective such an attack can be, and they provide analysis of the threat model and its limitations.

The other work [150] takes a different approach to this backdoor attack. Rather than trying to produce an altered GNN, this method seeks to alter training data by injecting a trigger (again taking the form of a subgraph) as well as arbitrarily changing the label. For this trigger, they randomly (using various methods to ensure similarities to the real data) generate a subgraph to insert. Interestingly, their results show that fixing this subgraph (one randomized trigger shared across every poisoned training and test instance) barely performs better than each subgraph being individually randomized. In addition, the authors provide a certified defence against this treat model. Their experimental results show the effectiveness of the attack, however, their certified defence is ineffective in some settings, necessitating further study.

In a more recent paper, researchers [172] propose a new approach to backdoor attacks on GNNs, based on *motifs* which are recurrent and statistically

significant subgraphs. To select the trigger, then, they analyze the motifs in available graphs, and construct an appropriate trigger. Their experimental results demonstrate more effective attacks than existing methods, as well as ensuring the target model's performance on clean test instances isn't compromised.

2.3.2 Attacking Deep Learning. Attacks to deep learning systems in general are much more well-researched, especially in computer vision, than those targeted against graph learning models.

2.3.2.1 Evasion Attacks. Adversarial examples targeted against deep learning models were initially introduced by researchers [131] who noticed that imperceptible modifications could cause an image to be misclassified by image classification models [164]. Furthermore, they found that adversarial examples generated against one network transferred effectively to other models with different architectures or even different training data sets.

While effective, the method introduced by the previous paper was inefficient, and relied on a linear search to find the best imperceptible perturbation. To address this flaw, the Fast Gradient Sign Method [42] was introduced. Intuitively, this method computes the gradient of the classification loss exactly once. Then, each pixel value is modified with the same magnitude, based on the sign of the gradient with respect to that pixel. Similarly, the Fast Gradient Value method [115] also computes the gradient exactly once. However, they modify each pixel with the raw gradient value, rather than making modifications of the same magnitude to each pixel. Note that this allows larger per-pixel modifications than the previous method.

Seeking to improve on the weaknesses of single step adversarial example generation (imprecision and relatively easy defense primarily) as well as the

weaknesses of traditional iterative methods (getting stuck in local optima, unstable optimization) researchers [27] applied momentum to the gradient descent method of optimizing adversarial examples. Additionally, they formulated their attacks against an ensemble of models (via averaging their logit outputs) to generate broadly applicable attacks. Their experimental results demonstrate better attack performance than the single step or iterative (without momentum) methods.

Working in a different direction, DeepFool [81] seeks to understand adversarial examples and improve model robustness by efficiently and precisely computing adversarial perturbations. Specifically, their method iteratively linearizes the classification model, and then computes a minimal step to take, repeating until the class of the target instance changes. Intuitively, this method seeks to find the minimal possible perturbation that produces the desired misclassification. Experimentally, they demonstrate that they are able to produce adversarial examples more reliably than previous methods, and training models on their examples significantly improves robustness.

Another creative work [126] explored the viability of attacking a single pixel. Their method uses differential evolution (a genetic algorithm that does not require computing gradients) to produce this extremely limited attack. Ultimately, they are able to change the classification of 67% of images in the CIFAR-10 dataset, and 16% of the images in the ImageNet dataset, despite only modifying one pixel per image.

2.3.2.2 Poisoning Attacks.

One early work investigating poisoning attacks on deep learning models [83] uses the metagradient method to optimize its attack. The intuition here is that, when a model is being trained, each update is itself a differentiable operation. By unrolling these updates, an attacker can

compute the gradient of the final weights with respect to the training data, enabling poison attack optimization via gradient descent. To make the gradient calculation tractable, they consider only a few steps of updates to the model while training. Additionally, they find that attacks generated can be transferred effectively to different training algorithms.

Improving on the previous work, MetaPoison [53] employs the same metagradient method except on an ensemble of target models, each at different stages of their training process. By averaging the attack gradients over all of them, the authors are able to create robust attacks transfer effectively across models. Another improvement along these lines is Witches' Brew [38] which introduces a *gradient alignment* component to the attack. Overall, they seek to match the direction of the attacker's loss gradient on a target image with the classifier's loss gradient on that image. Intuitively, what this does is ensures that when the classifier takes an optimization step based on that image, it is also reducing the attacker's loss on that image, furthering the poisoning attack's goal.

Rather than directly or approximately trying to solve the bilevel problem underlying the poisoning task, some methods train generative models to directly produce poisoned images. One such work [158] uses auto-encoders to speed up the poison generation process. Experimentally, the computation time is significantly lower, though their generated attacks are on average less effective than the iteratively produced baseline. Another [84] uses a GAN based model where the generator is trained against a classifier *and* against a discriminator (which seeks to detect the difference between a poisoned instance and a clean one). In contrast with the previous work, this serves to create unnoticeable poisoned instances that are also effective for the attack goal. Furthermore, this enables them to study

differences between attackers with various levels of imperceptibility concerns simply by tuning the ratio of the discriminator's loss to the classifier's loss when training the generator.

Another work [119] pioneered what are called feature collision attacks. Essentially, they seek to misclassify a target image, i, in the test set as some target class, c. Their mechanism for doing this is by manipulating instances of c in the training set such that their feature space representation moves closer to that of i. Additionally, they find that overlaying a mostly transparent watermark of i to the poisoned training set images boosts the power and the transferability of these attacks.

2.3.3 Defense and Robustness. Naturally, much research [128] has also been done into making models resistant to such attacks. Interestingly, researchers [144] have found a tradeoff between adversarial robustness (against evasion attacks) and backdoor robustness (against poisoning attacks). This suggests that deployed models should be careful to consider both threats lest they increase their vulnerability to one when addressing the other.

Designed to mitigate both poisoning and evasion attacks, researchers [143] follow previous work in using randomized smoothing during training. Furthermore, they're able to theoretically analyze the robustness bound against poisoning attacks, proving that their defense is effective. Prior work focused on empirical robustness against poisons; research into certified defenses against *poisoning* attacks is crucial and still sparse. Experimentally, they also show the value of their technique on a variety of datasets.

To address backdoor attacks, work has investigated systematically detecting and covering up the trigger [137]. This method is notable for requiring no insight to

the model being used, and no modifications to the training process itself. Instead, they simply test inputs to the trained model to identify any backdoor triggers. Then, they cover the trigger image using the dominant color of the original image to ensure similarity. While they provide no theoretical guarantees, empirically, their method outperforms existing work, even when compared to white box methods.

Another approach [165] uses metagradients to "unlearn" the backdoor triggers after a model has been trained. The general approach of unlearning triggers was well-established before this paper, but compared to the existing techniques, this work is able to accomplish the task an order of magnitude more efficiently. Furthermore, unlike other approaches, their process remains effective in the case where access to clean samples is highly limited.

Yet another direction focuses on training directly on poisons to mitigate their potential effect [39]. While this approach was well studied to defend against evasion attacks, this work's contribution was to consider it against training time attacks. Furthermore, they find that it generalizes well against multiple threat models (including highly targeted attacks) and is more resource efficient than comparable methods.

In graph learning, one work [67] observes that existing attacks tend to prefer similar nodes. Based on that observation, they seek to create a "universal" defence against attacks which could be applied to arbitrary nodes on the graph. Essentially, this method removes or adds edges to key nodes that are believed to be potential attack targets. Unlike prior research, their approach is designed (and shown) to work against targeted attacks.

Another work [151] tries to identify poisoned edges using Jaccard similarity, taking the ones with the lowest score and then removing them from the graph. To

ensure that the graph structure isn't too damaged by this defense, they utilize the minimum connectivity principle as the termination condition for their algorithm. Their experimental results are encouraging, showing effective defenses against poisoning attacks with notably less performance impact than existing methods.

Using random smoothing, researchers [139] were able to provide robustness guarantees for any arbitrary graph neural network against both node classification and graph classification tasks. To compute the perturbation size, they formulate finding the optimal random perturbation magnitude as an optimization problem. Solving this problem exactly is unrealistic so they devise an innovative technique based on analyzing regions within the graph. Experimentally, their certified accuracy results on real-world datasets are encouraging.

2.4 Miscellaneous

Meta Learning. In machine learning, meta learning is an approach designed to optimize the training process itself. Historically, meta learning was focused on *improving* models, though more recently, meta learning based attacks have proven effective. Muñoz-González et al. used a metagradient method to optimize a poisoning attack by back-propagating *through* the learning process [83]. They demonstrated that this approach was effective against a variety of decision makers, for multiple different tasks. Interestingly, they found that these poisoning attacks could be effectively transferred to models other than the one against which they were optimized [83]. Zugner et al. utilized the metagradient method to attack graph learning problems, creating an attack capable of dramatically reducing global node classification accuracy [177]. MetaPoison uses shallow metagradients averaged over multiple models at each poison optimization step to produce a robust yet subtle attack on image classification that can be effectively transferred beyond the

original target model [53]. Similar to these papers, our work in Chapter 3 focuses on a metagradient poisoning attack. Our contribution is in extending metagradient attacks to data based decision making models, and providing a detailed overview of the challenges in creating poisoning attacks in this setting.

Reinforcement Learning in Healthcare. RL has been widely used in tackling various problems in healthcare. In particular, RL was used in developing effective personalized treatment plans which can be adaptive to the dynamic changes of clinical states. There are several works in this line of research including studies of chronic diseases such as cancers [171, 2, 46, 94], diabetes [15, 26, 92, 9], anemia [36, 37, 73], and HIV [162, 98]. In addition, there is an increasing number of studies that applied RL techniques to problems in critical care such as generating optimal sepsis treatment policies [117], and anesthesia control [80, 123]. RL was also used in automated medical diagnosis [116, 40, 24, 56]. We refer readers to [163] for a complete literature review.

The research topic that is closest to our work in Chapter 5 is the problem of health management. Specifically, there are works on using RL to optimize messages sent to users to improve their physical activities [47, 161]. They essentially developed a mobile phone app that runs in the background of patients' smartphones and automatically collects data of physical activity performed by patients. They then run RL that utilizes the collected data to determine which SMS message is likely to increase the physical activity of the patient. This approach requires consensuses from patients to record all of their physical activities. Our work focuses on developing a personalized text message mechanism which targets not only physical activities but also diabetes-related knowledge and food

consumption. This is accomplished through an automatic message/question/answer process in which participants can opt to respond or not.

Reinforcement Learning with Sparse Rewards. Sparse rewards are a well known issue in RL. Over time, various approach have been proposed to solve the problem including reward shaping [50, 133], curiosity-driven exploration [99, 111, 173], and return decomposition [7, 35, 114]. Return decomposition has received particular interest since RUDDER [7] proved itself to perform well at model-free learning in the presence of sparse rewards. It uses an LSTM to predict state values based on state-action history, and assigns rewards based on the differences in predictions for consecutive states in a trajectory. Later work [100] has investigated improving RUDDER by making it more sample efficient.

Other return decomposition approaches have been devised, such as a least-squares based approach for assigning rewards to all states in a trajectory [29] and the uniform reward redistribution of IRCR [35] which divides the total episodic reward equally among transitions, and then averages this value over multiple trajectories. These approaches both have their merits, but the least-squares based approach scales poorly to large problems with long trajectories, while the uniform approach cannot identify the temporal structure of episodic rewards [114], which makes it a poor fit for our setting. A recent state of the art return decomposition approach called RRD [114] draws random sub-sequences of state-action pairs from a trajectory buffer to learn their contribution to the total episodic reward. This represents a balance between the effectiveness of least-squares return decomposition [29] and the computational efficiency of uniform reward distribution [35].

Common among the return decomposition approaches discussed in this subsection is the assumption of perfect state observations. Our work, on the other hand, focuses on RL settings in which there presents not only the challenge of sparse rewards but also the challenge of incomplete state observations.

Decision Transformer. Transformer models have drawn a great attention from the RL community. Decision transformer, in particular, was in introduced in [21], was able to generate future actions that achieve the desired return in offline reinforcement learning. There are other variants of transformer model were developed to tailor different offline RL settings [130, 55, 32, 136]. A recent work by [174] introduces online decision transformer that can work for online RL by blending offline pre-training with online fine-tuning. Note that these transformer-based RL approaches rely heavily on a significant amount of training data to pre-train the transformer part of their model.

Knowledge Tracing. In Chapter 5, our behavior modeling of participants is related to knowledge tracing, an important research research areas for enhancing personalized education [25]. The main task is to build machine models of the knowledge of a student as they interact with coursework. Recently, given the rise of deep learning, there have been several works that utilized deep neural nets to model the student learning [105, 160, 152, 41, 95]. Our work, on the other hand, focuses on building a ML model of behavior change of participants in our study as they interact with our intervention system. We target not only diabetes-related knowledge improvement of participants, but also their physical activity and food consumption dynamics, as influenced by our messages sent to them on a weekly basis.

2.5 Conclusion

In this chapter, we provided an overview of concerns surrounding real-world data. We investigated security games, which model interactions between adversaries. Defenders often rely on attacker's past behavior to build defenses against them, meaning that savvy attackers could manipulate this data nefariously. In the field of data-based decision making, we investigated various applications of this paradigm, discussed the different approaches to find solutions, and mentioned the lack of adversarial research here so far. Through the field of adversarial learning, we explored various approaches for attacks (primarily poisoning attacks) as well as defense techniques. Combining insights from all these fields could allow us to build more robust data-based decision making systems and reduce the threat of attacks to AI applications deployed in the real world.

CHAPTER III

DIRECT ATTACKS ON DATA-BASED DECISION MAKING MODELS

Acknowledgment. This chapter is adapted from a paper published in GameSec-22. I was the first author, and I performed all the programming and experimental work. Writing, as well as the theoretical analysis, was a group effort between myself and my co-authors: Wong Wai Tuck, Professor Arunesh Sinha, and Professor Thanh H. Nguyen.

As machine learning has been gaining applications and interest from both research and industrial communities, the opportunities for and the potential cost of failure grow. Some sources of failure are well explored, such as poorly chosen models and biased datasets. More recently, research has also considered another avenue for failure: intelligent adversaries that wish to manipulate the results of machine learning models [70, 51]. For example, adversaries can perform *evasion attacks* [11, 66, 97] to alter the classification of particular samples at test time; this requires access to some data that will be taken as input by a pre-trained model. Alternatively, with access to the training data, attackers can perform *poisoning attacks* [54, 12, 119]. The goal of poisoning attack is to manipulate the training data such that the resulting model offers advantage to the adversary. *Adversarial machine learning* is the field that includes study of both evasion and poisoning attacks, as well as design of models resistant to these attacks.

Another emerging area of study is that of *data-based decision making*. Many machine learning applications involve a data to decision pipeline: first using known data to construct a predictive model, then applying the predictive model to unknown data, and lastly making decisions based on those predictions. Traditional approaches here have been *two-stage*, with the predictive model being optimized

solely for its prediction accuracy [140, 30, 82, 154, 141]. If the prediction model is perfect over the prediction space, the two-stage approach would be optimal. However, complicated prediction boundaries in high dimension spaces can never be modeled perfectly even with large but finite data; in fact, for data driven decsion making the end goal is to make the best decisions possible, but the prediction model itself is not being optimized with that goal in mind. As a consequence of this observation, a method often referred to as *decision focused* learning seeks to directly integrate the decision optimizer into the prediction model during training. Hence, decision focused learning uses the decision quality to train the network. Updating the model via gradient descent, then, can be accomplished by differentiating *through* the solution to the decision optimization. This approach has proven more effective than corresponding two-stage models in some applications. However, this approach is significantly more computationally expensive, as each forward pass in the training process requires solving the optimization.

Our work lies at the intersection of data-based decision making and adversarial learning. We investigate the vulnerabilities of data-based decision making methods by developing poisoning attacks against these methods. To our knowledge, our work is the first one exploring this topic. Specifically, we look into using *end-to-end attacks* against both the aforementioned data-based decision methods designed for convex optimization, as well as a third model which we call the *simple joint* model. Here, the optimizer is itself approximated by a neural network. Furthermore, as it is important to understand the *transferability* of poisoning attacks between different models, we also investigate how effectively our generated attacks can be transferred beyond the originally targeted method (e.g.

computing an attack against a two stage model and then also testing the generated poison on a decision focused model).

Our *first* contribution is to create a meta-gradient based poisoning attack. Put simply, we unroll the target model's training procedure (which consists entirely of differentiable steps) to differentiate through the training and calculate gradients of the attacker's loss function with respect to the training data itself. Then, we use these gradients to perform projected (into the feasible space defined by constraints on the attack) gradient descent.

Our *second* contribution is to demonstrate that existing state of the art methods in other domains (specifically Metapoison [53] in computer vision) may not be directly applicable to the field of data-based decision making. We accomplish this by attacking a simple data-based decision making learner (using Metapoison to solve the attack) as well as testing the found attack on both two-stage and decision focused learners. The ineffectiveness of this approach for our problem suggests that new techniques may have to be developed for poisoning attacks on data-based decision making models.

Our experiments yield several findings that should be of use to future research. Most notably, attacking a decision-focused learner directly is a particularly difficult ask due to the complexity of the learner's training process. Beyond the (significant) computational requirements of solving the attack, any stability or precision issues within the learner's gradient calculation are compounded when computing the meta-gradient. Common machine learning pitfalls such as exploding or vanishing gradients appear frequently and are harder to counteract. Furthermore, the complexity of the solution space (which scales with model size, optimization objective, and constraints) means that many

optima of various quality exist, and finding a good one with gradient descent is not guaranteed. These effects are less noticeable when attacking the two-stage model or the simple joint model, as their training gradient updates do not involve backpropagating through an optimization problem.

Furthermore, we investigate the transferability of our method's attacks. Previous work has shown the transferability of meta-gradient based poisoning attacks [83]. Our experiments show that this property still applies, to varying degrees, across data-based decision making methods. This finding aligns with the general *transferability phenomenon* found in adversarial machine learning [96]. Primarily, we observe that poisons created against a two-stage learner effectively transfer to a decision-focused learner.

3.1 Data-based Decision Making

Data-based decision making refers to a common paradigm in artificial intelligence in which we are concerned with three related pieces of information: directly observable data (denoted by u), data that will be unobservable at test time (denoted by θ), and a *decision* that must be made (denoted by x). The decision, x, depends directly on θ, which in turn can be predicted based on u. The ultimate goal in a data-based decision making problem is to find an optimal decision to maximize a utility function, abstractly represented as follows:

$$\max_{x \in X} f(x, \theta)$$

where x is the decision variable and $X \subseteq \mathbb{R}^K$ is the set of all feasible decisions. Note that the objective, f, depends directly on the *unobservable* parameter θ, which must be inferred from the correlated observable data, u. In this chapter, we focus on the problem setting in which the decision space X can be represented as a set of linear constraints $X = \{x \in \mathbb{R}^K : Ax \leq b\}$ where (A, b) are constant.

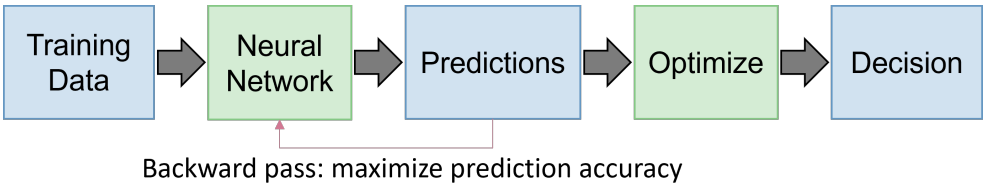


Figure 3. Depiction of a two-stage learner

In literature, there are two main approaches used to tackle the data-based decision making problem. The first approach, named **two-stage approach**, divides the problem into two separate phases. The first phase is the learning phase in which the unobserved parameter θ is learnt based on some training dataset $\mathcal{D} = \{(u_1, \theta_1), (u_2, \theta_2), \cdots, (u_n, \theta_n)\}$ in which each data point i is associated with a feature vector $u_i \in \mathbb{R}^d$ and a true label $\theta_i \in \mathbb{R}^K$ ($\theta_i \in \mathbb{N}^K$ if it is categorical). Then in the second phase which is called the decision-making phase, the decision x will be optimized based on the learning outcome θ. The second approach, named **decision-focused** learning, on the other hand, considers a single end-to-end pipeline (with an intermediate learning layer) that attempts to directly optimize the decision based on the training data $\mathcal{D}$. In addition to these two main approaches, in this paper, we create a third simple approach, named **simple joint approach** that formulates the data-based decision making as a simple learning problem. We consider this approach as a baseline to study poisoning attacks in this data-based decision making setting.

In the following, we first describe in details all these three approaches. We then present our optimization formulations to compute poisoning attacks to these three approaches, which are challenging to solve. Our proposed methodology to solve these optimization problems will be presented in Section 3.5.

3.2 Two-Stage Approach

3.2.1 Learner Description. The traditional approach to data-based decision making is *two-stage* [140, 30, 141]. The first of these stages is predicting the unknown parameter θ from the observed feature vector u. The second stage, then, is to compute the optimal x given the predicted θ (Figure 3). Predicting the *unknown* parameter θ can be done using a parametric model, denoted by $\hat{\theta} = g(u, w)$. Here, w is the model parameter that needs to be determined. Given a training dataset $\mathcal{D} = \{(u_1, \theta_1), (u_2, \theta_2), \cdots, (u_n, \theta_n)\}$ in which each data point i is associated with a feature vector $u_i \in \mathbb{R}^d$ and a true label $\theta_i \in \mathbb{R}^K$ ($\theta_i \in \mathbb{N}^K$ if it is categorical), the decision maker first trains a predictive model $g(u, w)$ to predict the label of a data point u. The learner seeks an optimal model parameter w^* that minimizes the training loss, abstractly formulated as follows:

$$\min_w \mathcal{L}(\mathcal{D}, w)$$

For example, one can use mean squared error as the training loss:

$$\mathcal{L}(\mathcal{D}, w) = \frac{1}{n} \sum\nolimits_i (\theta_i - g(u_i, w))^2$$

Once the model has been trained (yielding w^*), the decision maker can use observed u values to predict θ value (i.e., $g(u, w^*)$), then use that prediction to find an optimal decision by solving the following optimization problem:

$$\max_{x \in X} f(x, g(u, w^*))$$

3.2.2 Poisoning Attack Formulation. In designing poisoning attacks, we assume an adversary can alter the training data by injecting a small perturbation to every data point. More specifically, each feature vector u_i can be altered by adding a small quantity ϵ_i with the constraint that $lb_i \leq \epsilon_i \leq ub_i$. Here, $lb_i < 0$ and $ub_i > 0$ represent the maximum perturbation the adversary can apply

to the data point i. Intuitively, (lb_i, ub_i) captures the adversary's capability. The adversary attempts to optimize some goal, for example, minimizing the decision maker's utility in the test set or forcing the decision maker to produce a particular target decision output for some data points in the test set. We represent this poisoning attack on a two-stage model with the following general formulation:

$$\min \mathcal{L}^{adv}(x^*, \theta^{target}) \tag{3.1}$$

$$\text{s.t. } x^* \in \underset{x \in X}{\operatorname{argmax}} f(x, g(u^{target}, w^*)) \tag{3.2}$$

$$w^* \in \underset{w}{\operatorname{argmin}} \mathcal{L}(\mathcal{D}^{poison}, w) \tag{3.3}$$

$$\mathcal{D}^{poison} = \{(u_1 + \epsilon_1, \theta_1), \cdots, (u_n + \epsilon_n, \theta_n)\} \tag{3.4}$$

$$\epsilon_i \in [lb_i, ub_i], \forall i = 1, 2, \cdots, n \tag{3.5}$$

where $(u^{target}, \theta^{target})$ is the adversary's target element. Line 3.1 simply represents a general objective for the adversary. For example, if the adversary's goal is to minimize the decision maker's utility on this target, then $\mathcal{L}^{adv}(x^*, \theta^{target}) = f(x^*, \theta^{target})$. Line 3.2 is the optimization problem solved by the learner given the network output. Equation 3.3 then represents the optimal network weights as a function of the learner's training. Next, line 3.4 denotes the training dataset, altered by the attacker with poison values ϵ. Lastly, line 3.5 denotes the restrictions on the attacker's power, specifically a magnitude constraint on each poison element.[1] Solving the above optimization problem optimally is challenging since it has multiple connected levels of optimizations.

[1]Note that our formulation can be generalized to multiple targeted data points in the test set by taking the sum of losses over these data points. In addition, this can be also extended to incorporate perturbations on labels θ_i by introducing additional perturbation variables α_i to add to the labels.

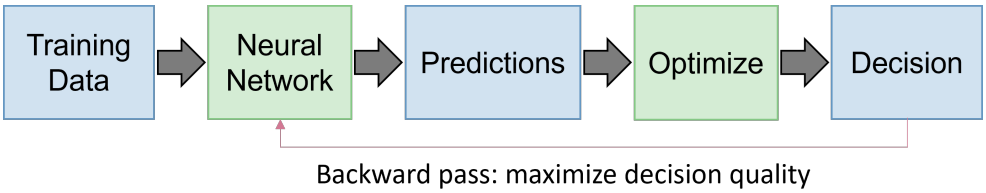


Figure 4. Depiction of a decision focused learner

3.3 Decision Focused Approach

3.3.1 Learner Description. While the two-stage approach is straightforward and effective, its training process is disconnected from the end goal of the system. More specifically, the model is being trained for prediction accuracy, whereas the ultimate objective is to produce good decisions [147].

A recent approach called *decision focused* learning seeks to bridge the disconnect between the training and the decisions produced, while still utilizing an explicit optimization solver (Figure 4). In theory, this approach can improve final decision quality by concentrating the (inevitable) prediction errors in areas that will have the least detrimental effect. For each training data point, θ is predicted from u and the optimization problem is solved to produce x. Then, the network weights are updated via gradient descent to maximize the decision quality. Intuitively, we can think of this as incorporating a convex optimization layer as the last layer of a neural network. This method can give improved results over the two-stage approach, at the cost of training time [147]. Essentially, in a decision-focused approach, the loss function the learner minimizes is the negative mean decision quality:

$$\min_{w} \ \mathcal{L}(\mathcal{D}, w)$$
$$\text{where } \mathcal{L}(\mathcal{D}, w) = -\frac{1}{n}\sum\nolimits_i f\left(\theta_i, x^*(\hat{\theta}_i)\right)$$

In this case, x^* is a result of solving the following optimization problem:

$$x^*(\hat{\theta}_i) \in \underset{x \in X}{\operatorname{argmax}} f(x, \hat{\theta}_i)$$

where $\hat{\theta}_i = g(u_i, w)$ is the network output.

Unlike the two-stage approach, here one must differentiate *through* the decision optimization problem to optimize the model parameters w. This can be accomplished by using the implicit function theorem on the KKT conditions of the optimization problem [5].

3.3.2 Poisoning Attack Formulation. Given the decision-focused formulation, we now can represent the problem of finding an optimal poisoning attack as the following optimization problem:

$$\min \mathcal{L}^{adv}(x^*, \theta^{target}) \tag{3.6}$$

$$\text{s.t. } x^* \in \underset{x \in X}{\operatorname{argmax}} f(x, g(u^{target}, w^*)) \tag{3.7}$$

$$w^* \in \underset{w}{\operatorname{argmin}} \left[\mathcal{L}(\mathcal{D}^{poison}, w) = -\frac{1}{n} \sum\nolimits_i f\Big(\theta_i, x^*(\hat{\theta}_i)\Big) \right] \tag{3.8}$$

$$\text{given } x^*(\hat{\theta}_i) \in \underset{x \in X}{\operatorname{argmax}} f(x, \hat{\theta}_i) \tag{3.9}$$

$$\text{and } \hat{\theta}_i = g(u_i + \epsilon_i, w) \text{ is the network output.} \tag{3.10}$$

$$\mathcal{D}^{poison} = \{(u_1 + \epsilon_1, \theta_1), \cdots, (u_n + \epsilon_n, \theta_n)\} \tag{3.11}$$

$$\epsilon_i \in [lb_i, ub_i], \forall i = 1, 2, \cdots, n \tag{3.12}$$

At a high level, the general attack formulation in this setting is similar to the two-stage case. However, solving the above optimization problem is much more challenging since the learner's training Eq. (3.8 – 3.10) involves an inner optimization layer which depends on the decision optimizer for every training data point.

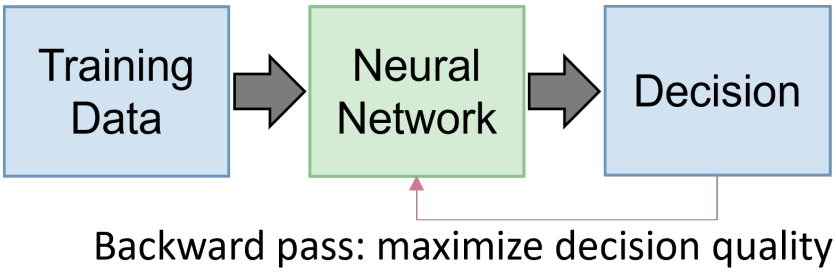


Figure 5. Depiction of a learner using the simple joint approach

3.4 Simple Joint Approach

3.4.1 Learner Description. A naive approach to data based decision making is to train a parametric model using the features (u) to directly predict the optimal decision (x) (Figure 5). Similar to the decision focused approach, we use negative mean decision quality as the loss function:

$$\min_w \ \mathcal{L}(\mathcal{D}, w)$$
$$\mathcal{L}(\mathcal{D}, w) = -\frac{1}{n}\sum\nolimits_i f\Big(\theta_i, \hat{x}(u_i)\Big)$$

In this case, however, $\hat{x}$ itself is the network output: $\hat{x}(u_i) = g(u_i, w)$. This bypasses the need to directly predict the labels ($\hat{\theta}$). Intuitively, we can think of this as implicitly learning the predictive task "inside" of the network.

Alternatively, we could solve the optimization problem for each training set instance prior to training (producing x_i^* where $x_i^* \in \text{argmax}_{x \in X} f(x, \theta_i)$) and then train the network to produce decisions as close as possible to these x^* values. In this method, we could use MSE as the loss function: $\mathcal{L}(\mathcal{D}, w) = \frac{1}{n}\sum_{i=1}^{n}(x_i^* - g(u_i, w))^2$. However, we found this approach less effective and more prone to overfitting than directly maximizing decision quality.

One complication when training a network to solve optimization problems is the constraints (if any exist) on the solution. Inspired by Shah et. al [120], we

utilize a specially designed neural network layer to enforce constraints throughout the training process, ensuring valid decisions are made [120].

In practice, this naive approach is often ineffective as training networks to directly solve optimization problems is difficult. However, we investigate this simple model as a target for generating poisoning attacks that can then be transferred to the more sophisticated models.

3.4.2 Poisoning Attack Formulation. The attack formulation here is similar to the previous cases. The difference is in the second line; rather than producing predictions, this simple joint model simply treats the network output as the decision itself, and is trained accordingly:

$$
\begin{aligned}
&\min \mathcal{L}^{adv}(x^*, \theta^{target}) \\
&\text{s.t. } x^* = g(u^{target}, w^*) \\
&\quad w^* \in \underset{w}{\operatorname{argmin}} \left[\mathcal{L}(\mathcal{D}^{poison}, w) = -\frac{1}{n} \sum_i f\Big(\theta_i, \hat{x}(u_i + \epsilon_i)\Big) \right] \\
&\quad \mathcal{D}^{poison} = \{(u_1 + \epsilon_1, \theta_1), \cdots, (u_n + \epsilon_n, \theta_n)\} \\
&\quad \epsilon_i \in [lb_i, ub_i], \forall i = 1, 2, \cdots, n
\end{aligned}
$$

3.5 Attack Generation Methodology

To solve the aforementioned optimization problems and determine poisoning attacks against each of these decision making approaches, we follow projected gradient descent. The core of gradient descent is to compute the gradient of the adversary loss $\mathcal{L}^{adv}$ with respect to the data perturbation ϵ, denoted by $\frac{d\mathcal{L}^{adv}}{d\epsilon}$. This gradient computation is challenging given that all the optimization problems involve multiple connected optimization levels.

Despite the differences among the aforementioned three data-based decision approaches, we employed two main computation techniques: (i) *computing*

gradients via meta gradient [10]— the main idea of this technique is to differentiate through the gradient descent steps in solving inner optimization levels. The main advantage of this technique is that it can be can be applied for any non-convex optimization problems. One disadvantage of this technique is that it is generally computationally expensive; and (ii) *computing gradient via implicit function theorem* [28, 146] — the main idea of this technique is to leverage convexity property, allowing us to differentiate through the KKT optimality condition. This technique is significantly less computationally expensive compared to the first technique. However, this technique is only applicable for convex optimization problem. Therefore, depending on the convexity of the problems, we then decide on one of these two techniques.

In the following, we first present in detail our proposed method to compute attacks to two-stage learning. Later, we will mainly highlight the differences or challenges regarding the decision-focused learning and the simple joint learning.

3.6 Attack to Two-Stage Approach

To solve the poisoning attack in this setting using gradient descent, the key is the gradient calculation of $\frac{d\mathcal{L}^{adv}}{d\epsilon}$, which can be decomposed into different gradient components via the chain rule:

$$\frac{d\mathcal{L}^{adv}}{d\epsilon} = \frac{d\mathcal{L}^{adv}}{dx^*}\frac{dx^*}{dg}\frac{dg}{d\epsilon} \qquad\qquad \frac{dg}{d\epsilon} = \frac{dg}{dw^*}\frac{dw^*}{d\epsilon}$$

Computing $\frac{d\mathcal{L}^{adv}}{dx^*}$ is straightforward, and $\frac{dg}{dw^*}$ is a result of the standard neural network back-propagation computation. On the other hand, computing the gradient components, $\frac{dx^*}{dg}$, the gradient of the optimal decision with respect to the label prediction $g(u^{target}, w^*)$, and $\frac{dw^*}{d\epsilon}$, the gradient of the optimal model parameter w^* w.r.t the perturbation ϵ, is not straightforward. This is because there is no explicit close-formed representation of x^* and w^* as a function of g and ϵ

respectively, despite the fact that x^* depends on g and w^* depends on ϵ. In the following, we present our meta-gradient based method to approximate $\frac{dw^*}{d\epsilon}$ given the learning part (neural network function) is non-convex. We will then present the implicit function theorem based method to compute $\frac{dx^*}{dg}$ since the decision optimization part is convex.

3.6.1 Computing decision gradient via implicit function theorem.

We focus on the problem setting in which the decision optimization is convex (i.e., the utility function $f(x, \theta)$ is convex in the decision variable x). This convexity setting has been widely considered in previous studies on data-based decision making [147, 146, 28, 1]. Based on this convexity characteristic, we leverage the implicit function theorem [65] to differentiate through the decision-optimization layer (i.e., computing $\frac{dx}{dg}$). Given the predicted value $\hat{\theta} = g(u^{target}, w^*)$, the decision-optimization component is formulated as a convex optimization problem:

$$\max_x \ f(x, \hat{\theta}) \ \text{ s.t. } \ Ax \leq b$$

Since this is a convex optimization problem, any solution that satisfies the following KKT conditions is optimal:

$$-\nabla_x f(x, \hat{\theta}) + \lambda \cdot \nabla_x (Ax - b) = 0$$

$$\lambda \cdot (Ax - b) = 0$$

$$Ax \leq b, \lambda \geq 0$$

where λ is the dual variable. Observe that the first equation indicates that x and λ are functions of $\hat{\theta}$. Based on the implicit function theorem, we can differentiate

through the first two equations to obtain the following gradient computation:

$$\begin{bmatrix} \frac{dx}{d\hat{\theta}} \\ \frac{d\lambda}{d\hat{\theta}} \end{bmatrix} = \begin{bmatrix} \nabla_x^2 f(x,\hat{\theta}) & A^T \\ diag(\lambda)A & diag(Ax-b) \end{bmatrix}^{-1} \begin{bmatrix} \frac{d\nabla_x f(x,\hat{\theta})}{d\hat{\theta}} \\ 0 \end{bmatrix} \tag{3.13}$$

3.6.2 Computing learning gradient via meta gradient. While we can leverage the convexity of decision optimization to compute the gradient of a decision with respect to the coefficients (as will be done when attacking the more complex models), we cannot apply the same approach for computing the learning gradient, $\frac{dw^*}{d\epsilon}$. This is because model learning is generally a non-convex optimization problem (as neural network models are non-convex in general). On the other hand, the implicit function theorem approach is most usefully applied to convex optimization. In order to tackle this challenge, we adopt the meta-gradient method [6].[2] This method works by assuming the model learning problem is solved via gradient descent. This is a reasonable assumption since neural network training typically relies on gradient descent method and its variants.

Based on this assumption, we can differentiate through the gradient descent steps. More specifically, we're concerned with the model's learning problem, abstractedly represented as follows:

$$\min_w \mathcal{L}(\mathcal{D}^{poison}, w)$$
$$\text{where } \mathcal{D}^{poison} = \{(u_1+\epsilon_1, \theta_1), \cdots, (u_n+\epsilon_n, \theta_n)\}$$

At each gradient step t, given the previous value of the model parameters w_{t-1}, the gradient descent update is as follows: $w_t = w_{t-1} - \delta \frac{d\mathcal{L}}{dw_{t-1}}$, where δ is the learning rate. Note that $\mathcal{L}$ is a function of the perturbation variables $\epsilon = \{\epsilon_1, \cdots, \epsilon_n\}$.

[2]We can also apply this method to compute the decision gradient. However, meta-gradient is much more computationally expensive compared to the implicit function theorem method for convex problems.

Algorithm 1: Poisoning Attack Generation for Two-Stage Learning

Input: training data $\mathcal{D} = \{(u_1, \theta_1), (u_2, \theta_2), \cdots, (u_n, \theta_n)\}$;
Input: target $(u^{target}, \theta^{target})$;
Randomly initialize perturbation values $\epsilon = \{\epsilon_1, \cdots, \epsilon_n\}$.
for $j = 1 \rightarrow nIter$ **do**
    `// Model learning`
    Initialize optimal learning loss $optL = \infty$;
    for $r = 1 \rightarrow nRound$ **do**
        Randomly initialize model parameter values w_0;
        for $t = 1 \rightarrow T$ **do**
            Update $w_t = w_{t-1} - \delta \frac{d\mathcal{L}}{dw_{t-1}}$;
            Differentiate $\frac{dw_t}{d\epsilon} = \frac{dw_{t-1}}{d\epsilon} - \delta \frac{d\left(\frac{d\mathcal{L}}{dw_{t-1}}\right)}{d\epsilon}$;
        if $\mathcal{L}(\mathcal{D}^{poison}, w_T) < optL$ **then**
            Update optimal learning $w^* = w_T$;
            Update learning gradient: $\frac{dw^*}{d\epsilon} = \frac{dw_T}{d\epsilon}$;
    `// Decision optimizing`
    Compute optimal decision based on the learnt model w^*:
    $x^* \in \text{argmax}_{x \in X} f(x, g(u^{target}, w^*))$
    Compute decision gradient w.r.t $\hat{\theta} = g(u^{target}, w^*)$ using Eq. (3.13)
    `// Projected gradient step`
    Update perturbation variable $\epsilon = \epsilon - \delta \frac{d\mathcal{L}^{adv}}{d\epsilon}$ where:
    $\frac{d\mathcal{L}^{adv}}{d\epsilon} = \frac{d\mathcal{L}^{adv}}{dx^*} \frac{dx^*}{dg} \frac{dg}{dw^*} \frac{dw^*}{d\epsilon}$;
    Project ϵ_i to feasible perturbation space: $[lb_i, ub_i]$ $\forall i$;
return ϵ;

Therefore, w_t is also a function of ϵ (except for w_0 which is the initial value, a constant). As a result, we can differentiate through this gradient step as follows:

$$\frac{dw_t}{d\epsilon} = \frac{dw_{t-1}}{d\epsilon} - \delta \frac{dG}{d\epsilon}$$

where $G(w_{t-1}, \epsilon) = \frac{d\mathcal{L}}{dw_{t-1}}$. By applying the chain rule, we obtain:

$$\frac{dG}{d\epsilon} = \frac{\partial G}{\partial \epsilon} + \frac{\partial G}{\partial w_{t-1}} \cdot \frac{dw_{t-1}}{d\epsilon}$$

If we run gradient descent in T steps, we can approximate the gradient of the optimal w^* with respect to perturbations ϵ as follows: $\frac{dw^*}{d\epsilon} \approx \frac{dw_T}{d\epsilon}$.

3.6.3 Projected gradient descent algorithm. Given this gradient computation, we illustrate our approach in Algorithm 1 where we run an iterative projected gradient descent process to compute an optimal attack. At each iteration j, given the current value of perturbation variables ϵ, Algorithm 1 first runs another inner gradient descent process to optimize the parameters w of the predictive model $g(u, w)$ based on the poison data $\mathcal{D}^{poison}$ (lines 5-13). At the end of this inner process, we obtain a trained model w^*. During this process, we simultaneously compute the learning gradient $\frac{dw^*}{d\epsilon}$.

Given the trained model w^*, Algorithm 1 proceeds into the decision optimization to compute the optimal decision x^* w.r.t the target u^{target} (line 14). Along with that computation, the gradient $\frac{dx^*}{dg}$ is computed (line 15). Finally, we update the value of ϵ based on the previous gradient computation (lines 16-17). This entire procedure (lines 5-17) is repeated until we reach a local optimal value of ϵ or reach the predetermined maximum number of iterations $nIter$.

3.7 Attack to Decision Focused Approach

Similar to the two-stage approach, in this setting, we aim to compute the gradient $\frac{d\mathcal{L}^{adv}}{d\epsilon}$, which can be decomposed into multiple components using chain rule:

$$\frac{d\mathcal{L}^{adv}}{d\epsilon} = \frac{d\mathcal{L}^{adv}}{dx^*}\frac{dx^*}{dg}\frac{dg}{d\epsilon} \qquad\qquad \frac{dg}{d\epsilon} = \frac{dg}{dw^*}\frac{dw^*}{d\epsilon}$$

However, as the two methods use different training processes, the details of the learning gradient calculation ($\frac{dw^*}{d\epsilon}$) differ significantly. In fact, it becomes much more complicated and computationally expensive due to the involvement of the optimizer in the training process itself.

Indeed, recall that for the calculation of the learning gradient ($\frac{dw^*}{d\epsilon}$), in general, we follow gradient descent at every to solve the model learning problem

and then differentiate through the gradient steps as explained in the previous section. That is, we have the following differentiation update:

$$\frac{dw_t}{d\epsilon} = \frac{dw_{t-1}}{d\epsilon} - \delta \frac{d\left(\frac{d\mathcal{L}}{dw_{t-1}}\right)}{d\epsilon}$$

where $\mathcal{L}$ is the training loss. In two-stage approach, this training loss has a closed-formed representation as a function of ϵ. Therefore, the above gradient computation is straightforward. On the other hand, in decision-focused approach, the model training is represented in Eq. (3.8–3.10), in which multiple decision optimizations for every data point is involved. The gradient $\frac{d\mathcal{L}(\mathcal{D}^{poison},w)}{dw}$ now depends on the gradient of the optimal decision w.r.t the prediction outcomes $\frac{dx^*(\hat{\theta}_i)}{d\hat{\theta}_i}$ for all i, since we have according to the chain rule:

$$\frac{d\mathcal{L}}{dw} = \sum_i \frac{d\mathcal{L}}{dx^*(\hat{\theta}_i)} \frac{dx^*(\hat{\theta}_i)}{d\hat{\theta}_i} \frac{d\hat{\theta}_i}{dw}$$

Computing the gradient $\frac{dx^*(\hat{\theta}_i)}{d\hat{\theta}_i}$ for all data points can be done via implicit function theorem as discussed in Section 3.6, which already involves complex computations including inverse matrix computation and the second derivative computation, etc. As a result, it becomes very challenging to take a further gradient step of $\frac{d\left(\frac{d\mathcal{L}}{dw}\right)}{d\epsilon}$. We discuss this challenge in the experiment section.

3.8 Attack to Joint Simple Approach

Finally, solving the attack on the joint sample approach is the simplest:

$$\frac{d\mathcal{L}^{adv}}{d\epsilon} = \frac{d\mathcal{L}^{adv}}{dx^*}\frac{dx^*}{d\epsilon} \qquad\qquad \frac{dx^*}{d\epsilon} = \frac{dx^*}{dw^*}\frac{dw^*}{d\epsilon}$$

In this case, x^* is simply the output of the neural network. Computing $\frac{d\mathcal{L}^{adv}}{dx^*}$ is straightforward, and $\frac{dx^*}{dw^*}$ is a result of the standard neural network back-propagation computation. The only challenging component here is $\frac{dw^*}{d\epsilon}$ as w^* is a

function of ϵ yet cannot be expressed in closed form. As when attacking the other models, we use the metagradient method to calculate this.

3.9 Experiment Setup

3.10 Attack Methods

For our experiments, we utilize three different methods to generate attacks and compare their effectiveness. Starting with the simplest model, the first method is based on MetaPoison [53] and is formulated against the naive end-to-end learner. More specifically, this attack utilizes multiple target models (each at a different stage of training) and *averages* their metagradients (limited to 2 training steps). Then, the attack is optimized alongside the target models. This method was found to produce effective, unnoticeable, and transferable attacks in the computer vision domain [53]. Our idea is to use this method against a simple learner in the data-based decision making domain to produce attacks that can then be leveraged against the more sophisticated learners (two-stage and decision-focused).

The second attack generation technique we consider involves attacking the two-stage learner directly. Here, rather than using the MetaPoison technique, we consider an attack trained against a single learner which is trained from scratch at each attack epoch, giving us a more complete metagradient (computed using Higher [43]). Unlike the previous method, this one requires differentiating through the solution to an optimization problem as the two-stage learner explicitly solves that optimization problem at test time. For this component, we use Qpth [5]. Once again, after an attack is generated, we further evaluate it by testing it against the decision-focused learner.

The third and most computationally complex attack we consider is one formulated directly against the decision focused learner. On the surface, this attack

is nearly identical to the one against the two-stage learner. When considered in more depth, however, it's a significantly more difficult problem, for reasons previously discussed. Thus, we are motivated to investigate the feasibility of attacking this model directly.

3.11 Experiment Domains

Synthetic Data. For our synthetic data experiments, we consider the following decision optimization problem:

$$\min f(x, \theta) = \frac{1}{2} x^T Q x - \theta^T x \text{ s.t. } ||x|| \leq D, Ax \leq b \tag{3.14}$$

where Q is a diagonal positive-definite matrix, serving as a penalty parameter to make the problem convex, and θ is the unknown parameter that needs to be trained. $||x|| \leq D$ is simply a magnitude constraint on the decision variable, while $Ax \leq b$ represents some other constraints on the decision space. This decision optimization formulation is typically used for representing shortest path, maximum flow, bipartite matching, and a range of other domains [146].

In our experiments, in order to predict the unknown parameter θ, we consider a simple neural network and randomly (according to the normal distribution) generate synthetic data to train this predictive network. The labels are computed as a function of the features, plus a small amount of random noise. In addition, regarding the decision optimization, we randomly generate decision constraints. The amount of constraints used are varied across experiments to explore how this affects the attack generation. These constraints are added incrementally: an experiment with 9 constraints would include the same constraints as the corresponding experiment with 7 constraints, in addition to 2 new constraints.

Stock Market Portfolio Optimization. In addition to this simplified artificial problem, we demonstrate our attack in the portfolio optimization domain. This is naturally modeled via data-based decision making, where, prior to the optimization itself, future stock returns and the covariances between stocks must be predicted. This makes the domain a natural choice for our decision-focused attack. Similar to other recent work [141], we utilize the Markowitz model [72] to maximize expected return while encouraging a diverse portfolio. Overall, the objective function of the optimization problem combines maximizing immediate return at each time step with minimizing risk, formulated as follows:

$$f(x, \theta, p, Q) = p^T x - \lambda x^T Q x$$

Where x is the investment decision made (a vector that sums to 1, representing percentage of investment in each stock), p is the expected immediate return, λ is a risk aversion parameter, and Q is a matrix capturing the covariance between the expected returns of all stocks. Intuitively, Q represents how correlated individual stocks are, and it is more risky to invest in correlated stocks. Thus, the penalty term incentivizes diverse investment.

The learning problem, then, is to utilize historical information about the stocks themselves to learn *both* the expected returns (p) as well as a 32 dimensional embedding for each stock. This embedding is then used to calculate the covariance between each pair of stocks, using cosine similarity. Specifically, we use the prices at the previous time step as well as rolling statistics as the input of the neural network to (separately) learn p and Q. As in [141] these statistics include a variety of sliding window means, as well as variances, of the historical stock prices. Loss functions for both the two-stage and the decision-focused models utilize ground-truth p and Q values directly computed from the dataset. Note that both p and

Q depend on the price data from *future* timesteps: p is the next timestep's return, while Q is the cosine similarity of the returns over the next 10 timesteps.

We utilize real-world historical stock data, downloaded from the Quandl WIKI dataset, from 2004 to 2017 [109]. The stocks used belong to the SP500, giving us 505 potential stocks to work with. Attacking the features exclusively is not meaningful here, as the features are computed based on the raw price. Due to this, we target our attack on the *raw* historical stock market data, which affects the features, the labels (p), and the covariance matrix (Q). We restrict our experiments to a setting with 50 stocks and 500 timesteps.

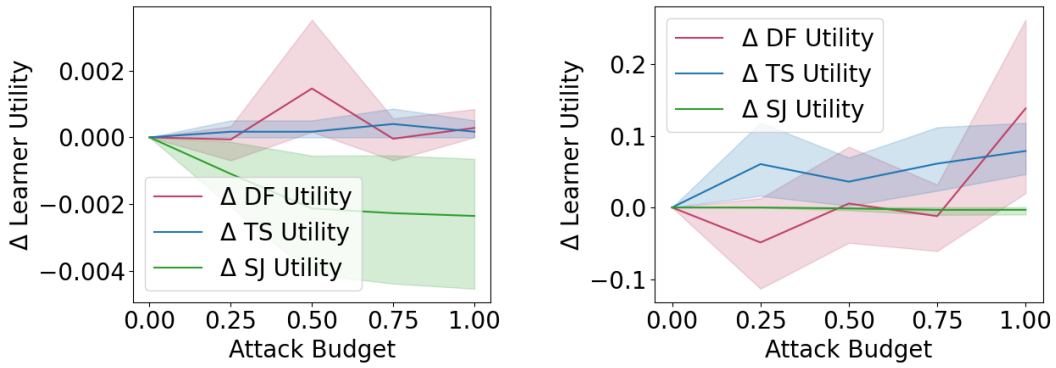
3.12 Results

Now we present the results of our experiments. For all the graphs, the results are averaged over 5 random seeds which determine both the initial network weights as well as the randomized attack starting points. In the synthetic data domain, this also corresponds to 5 different data sets (generated using the same normal distribution). For supplemental results, see the linked appendix[3].

3.13 Synthetic Data

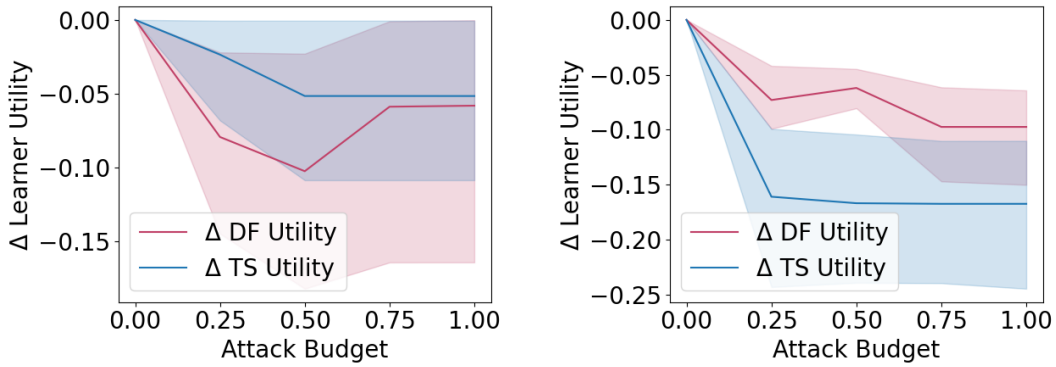
On the following graphs, a 'small dataset' refers to a setting with 250 instances in the training dataset, while a 'large dataset' refers to one with 750 elements. **Simple Joint Model.** In Figure 6, we display the effectiveness of attacks generated against a simple joint learner. Both cases here demonstrate similar trends. First, that the found attacks are only minimally effective against the simple joint learner itself. Secondly, we observe that when transferring these attacks to the decision focused and the two-stage learners the effect on their utility is inconsistent and follows no clear trends. This finding stands in contrast to the

[3]https://www.dropbox.com/s/6lznj4c1imk5qcm/DataBasedSupplemental.pdf



(a) Attack on Large Dataset (b) Attack on Small Dataset

Figure 6. Attacks generated against a simple joint model.

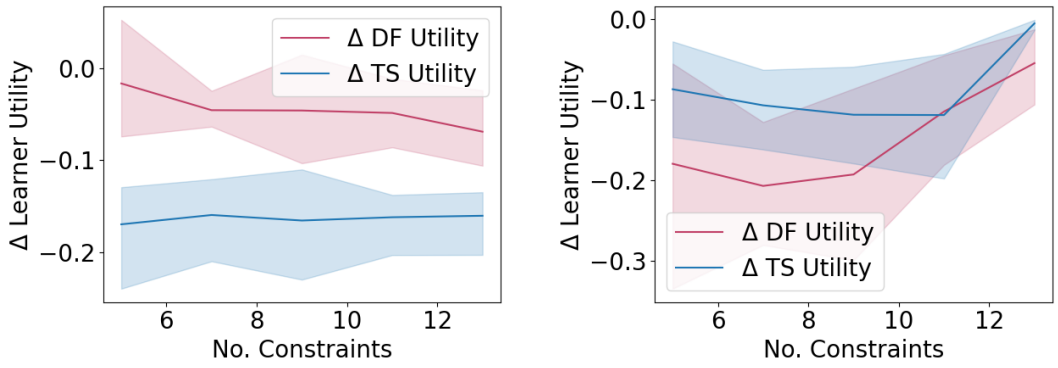


(a) Attack on Large Dataset (b) Attack on Small Dataset

Figure 7. Attacks generated against a two-stage learner

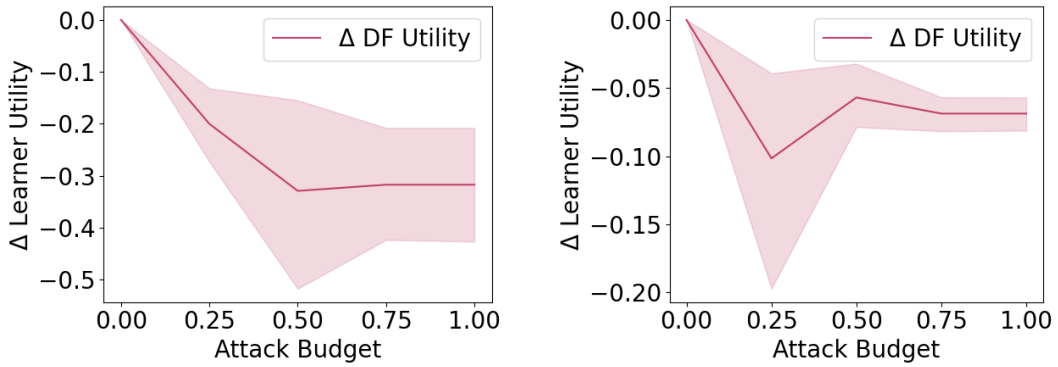
results obtained by MetaPoison in the field of computer vision [53]. While this result may be surprising, the problems being solved in data-based decision making are notably different from computer vision tasks, and the models we utilize are significantly less complex.

Two-Stage Model. In Figure 7 we show our results when generating an attack on the two-stage learner. Contrasted with the attacks in Figure 6, we observe significantly higher effectiveness, both against the two-stage learner itself and when transferring the attack to the decision-focused learner. This contrast further



(a) Attack on Large Dataset (b) Attack on Small Dataset

Figure 8. Effect of adding constraints on attack results



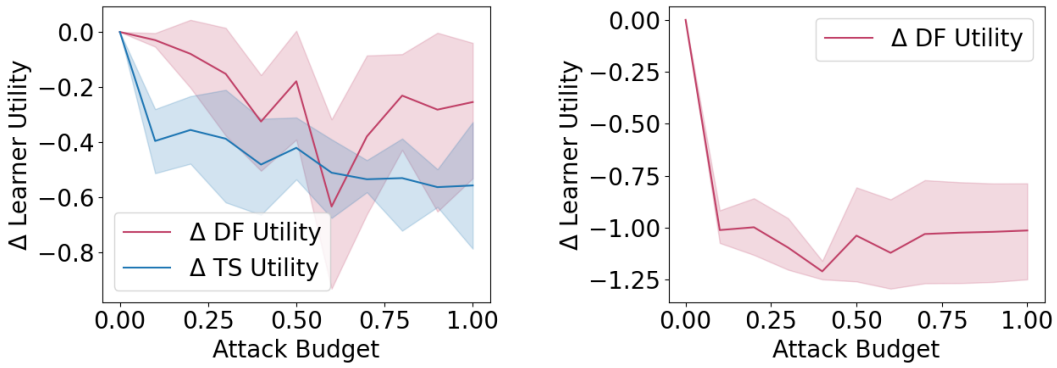
(a) Attack on Large Dataset (b) Attack on Small Dataset

Figure 9. Attacking a decision-focused model directly

suggests that methods from other domains (such as computer vision) may not be directly applicable when attacking data based-decision making models.

Figure 8 demonstrates the effect of introducing more constraints to the optimization problem. What we observe here is that while the effectiveness of our attacks is dependent on the constraints, there is no simple trend when varying them. This makes it hard to predict how effective a metagradient based attack will be when attacking a new problem in data-based decision making.

Decision Focused Model. In Figure 9 we examine the effectiveness of attacking



(a) Attack on two-stage learner (b) Attack on decision focused learner

Figure 10. Attacking a portfolio optimization model

a decision-focused learner directly. While our method is able to find good attacks in some scenarios, it is unreliable. Even in this simple setting, gradient descent often struggles to find a good optima, and this issue becomes even more apparent with a larger attack space. Combined with the prohibitive compute requirements of this attack, this is unlikely to be a practical approach in many data based learning settings.

3.14 Portfolio Optimization

Two-Stage Model. Figure 10a shows the results of attacking a two-stage model for portfolio optimization, as well as transferring that attack to a decision focused learner. Notably, we see that transferring the attack is often effective, though is inconsistent. This is likely due to the increased complexity of this domain compared to our synthetic setting, which includes both the transformation of raw prices into features as well as the objective of the optimization problem.

Decision Focused Model. In Figure 10b, we display the results of attacking a decision-focused model directly. In this case, this attack is on average more effective than the attacks on the two-stage model. Most of this is likely due to the decision

focused learner performing better when unattacked, getting an objective value of -0.005 to -0.094, compared to the two-stage learner that obtains objective values between -0.32 and -0.65. We also observe once again that higher 'budget' attacks (meaning a larger attack space) often lead to worse attacks (higher utility for the learner), further demonstrating the complexities of solving these attacks.

3.15 Conclusion

In this chapter, we formulated a generalized meta-gradient based poisoning attacks against two-stage models, decision focused models, and a simple joint model. We were able to provide insight into the difficulties of this attack by conducting extensive experiments in a synthetic domain as well as a real-world stock market portfolio optimization problem. These experiments show the following results. First, we observe that existing meta-gradient based techniques [53] may be ineffective here, despite being quite effective in the domain of computer vision. Next, we provide analysis showing that direct attacks on a decision-focused model are discouragingly difficult and problem dependent. Furthermore, despite the inherent training differences between two-stage and decision-focused learners, our results show that poisons crafted on a two-stage model can be effective against decision-focused models as well.

CHAPTER IV

COUNTERING POISONING ATTACKS ON GAME THEORETIC DATA-BASED DECISION MAKING MODELS

Acknowledgment. This chapter is adapted from a paper published in GameSec-21. I was the first author, and I performed all the programming and experimental work. Writing, as well as the theoretical analysis, was a group effort between myself and my co-authors: Professor Arunesh Sinha, and Professor Thanh H. Nguyen.

Learning adversary behavior from historical attack data is a firmly established methodology in adversarial settings, both in academic literature [87, 101], and in real world applications such as wildlife security [30, 132]. Herein lies a vulnerability: a clever attacker may modify its own behavior in order to conceal information or mislead the defender. This deceptive behavior can influence the defender's learning process, creating future gainful opportunities for the attacker. Indeed, such deception has received considerable attention in security games literature [34, 166, 90]. However, robustness of the defender to the adversary's deceit is much less explored. In this chapter, we investigate the defender's counteraction against attacker deception in a Stackelberg security game setting.

Our work builds upon the *partial behavior deception* model [88] in which the defender models the behavior of the entire attacker population using a single Quantal Response (QR) [75] model of which the parameter $\lambda \in \mathbb{R}$ is learned from past attack data. Among the attackers, however, there is a rational attacker who can cause harm to the defender by manipulating part of attack data. Such manipulation makes the defender learn an incorrect λ value, leading to an

ineffective defender strategy. Addressing the attacker deception is still an open problem, which is the focus of our paper.

As our *first contribution*, we develop a new technique to estimate the true behavior of the non-deceptive attackers (represented by a parameter value λ^{true} of `QR`), given the perturbed training data. Our technique leverages the Karush-Kuhn-Tucker conditions of the rational attacker's optimization to formally express the relation between true behavior of non-deceptive attackers (λ^{true}) and learning outcome (λ^{learnt}) forced by the deceptive attacker. Based on this relation, we find that there is an interval of possible values for λ^{true} which leads to the same deception outcome λ^{learnt}. Moreover, bounds of this interval are increasing in λ^{learnt}. We thus propose a binary-search based method which uses λ^{learnt} to guide the search for these bounds within an ϵ-error.

As our *second contribution*, we extend our first contribution, perhaps surprisingly, to apply in scenarios with small number of attacks. The core issue is that the empirical attack distribution induced by limited attack samples may be far different from the true attack distribution induced by λ^{true}, making it challenging to characterize the relation between the true behavior and the deceptive outcome. We overcome this challenge by re-formulating the attack sampling process as choosing random *seeds* $\mathbf{u}$ drawn from the uniform distribution on $[0, 1]$ followed by a deterministic computation on $\mathbf{u}$.

We first prove that given any fixed $\mathbf{u}$, all mathematical results (from our first contribution) hold for small number of attacks. As the random seed chosen by nature is unknown, we then leverage the above result to perform binary search for *multiple* random seeds and construct a new interval spanning all found intervals as our final estimate for the range of λ^{true}.

As our *third contribution*, we propose a maximin approach to optimize the defender strategy against the worst case within the uncertainty interval for λ^{true}. We formulate this maximin problem as a multiple non-linear programs, each corresponds to a particular optimal attack choice of the deceptive attacker. *Finally*, via extensive experiments, we show that, even when optimizing against a wide uncertainty interval of λ^{true}, our algorithm gives significantly higher utility for the defender, and less benefit for the deceptive attacker.

4.1 Related Work

Adversarial Learning Adversarial learning is a field within machine learning that has become increasingly popular [70, 124, 52, 71, 169]. The attacker deception here is analogous to a *causative attack* (or poisoning attack) in adversarial learning [52]. A significant difference between our work and adversarial learning is that we seek to maximize defender utility *through* predicting the attacker's behavior, whereas in adversarial learning, the end goal is prediction accuracy.

Attacker Behavior Inference Learning the behavior of bounded rational attackers is crucial, and a major area of interest in security games. Various models including `QR` have been explored [159, 57, 166, 122, 103]. As this learning is used to create a defender strategy, the training attack pool is vulnerable to manipulation by a clever attacker. This paper focuses on addressing this challenge in security games. Our work overlaps with settings in which one or more players has limited information [4].

Deception in Security Games Historically, most work has focused on deception from the defender side [175, 45]. In this scenario, the defender typically exploits information asymmetry to fool the attacker (e.g. in network security,

concealing some system characteristics). More recently, research has investigated deception from the attacker side [34, 90, 166] in SSGs, and the follower side in general Stackelberg games [33]. Much of this chapter concentrates on a single attacker whose payoff values are unknown to the defender. The attacker-deception model we utilize [88], on the other hand, describes a realistic scenario in which the defender must contend with multiple attackers of *unknown* behavior.

4.2 Preliminaries

4.3 Stackelberg Security Games (SSGs)

In SSGs [132], the *defender* must protect a set of T targets from one or more *attackers*. The defender has a limited number ($K < T$) of *resources* that each can be allocated to protect a single target. A pure strategy of the defender is defined as a one-to-one allocation of resources to targets. A mixed defense strategy, $\mathbf{x}$, is a probability distribution over these pure strategies. For the purposes of this paper, we consider no scheduling constraints to the defender's strategy, meaning that a mixed strategy can be compactly represented as a coverage probability vector, given by $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ where $x_i \in [0, 1]$ represents the probability that target i is protected by the defender and $\sum_i x_i \leq K$. We denote by $\mathbf{X}$ the set of all feasible defense strategies. In SSGs, the attacker is fully aware of the defender's mixed strategy and chooses a target to attack based on this knowledge.

An attack on target i gives each player a reward or a penalty, depending on whether the defender is currently protecting target i. If i is unprotected, the attacker gains reward R_i^a and the defender receives penalty P_i^d. Conversely, if target i is protected, the attacker takes penalty $P_i^a < R_i^a$ and the defender gains reward $R_i^d > P_i^d$. Given coverage probability x_i, the expected utilities for the

defender and the attacker for an attack on target i can be formulated as follows:

$$U_i^d(x_i) = x_i R_i^d + (1 - x_i) P_i^d$$

$$U_i^a(x_i) = x_i P_i^a + (1 - x_i) R_i^a$$

*Quantal Response Behavior Model (*QR*).* QR is an well-known model describing attacker behavior in SSGs [75, 159]. Intuitively, QR provides a mechanism by which higher expected utility targets are attacked more frequently. Essentially, the probability of attacking target i is given as follows:

$$q_i(\mathbf{x}; \lambda) = \left(e^{\lambda U_i^a(x_i)}\right) \Big/ \left(\sum\nolimits_j e^{\lambda U_j^a(x_j)}\right) \tag{4.1}$$

4.4 Partial Behavior Deception Model

Our work on developing an optimal counter-deception strategy for the defender is built upon the partial behavior deception model introduced by [88]. In this model, multiple attackers are present, who have the same payoffs but different attack behavior due to different rationality levels. Among these attackers, there is a rational attacker who intends to play deceptively to mislead the defender. The defender, on the other hand, is aware of the attackers' payoffs but is uncertain about the behavior of the attackers. The defender thus attempts to to build a behavior model, i.e., the QR model, to predict the attack distribution of the entire attacker population. Real-world applications such as wildlife conservation also use this single-behavior-modeling approach as park rangers usually cannot differentiate data collected, such as poaching signs, among multiple sources [57].

Two-phase learning-planning of defender. This model describes a *one-shot two-phase learning-planning* problem for the defender, consisting of a learning phase and a planning phase. This is the typical security game model used in literature [132, 159]. Essentially, in the learning phase, the defender uses training

attack data to estimate the parameter λ of QR using the Maximum Likelihood Estimation method (MLE), as formulated below:

$$\lambda^{\text{learnt}} \in \underset{\lambda}{\text{argmax}} \sum_{m} \sum_{i} z_i^m \log q_i(\mathbf{x}^m; \lambda) \tag{4.2}$$

where x_i^m is the defender's coverage probability at target i and step m and z_i^m is the corresponding number of attacks.

During the planning phase, the defender utilizes the learned λ^{learnt} value to optimize his defense against such an attacker. The optimal strategy, $\mathbf{x}^*$, is given by:

$$\mathbf{x}^* \in \underset{\mathbf{x} \in \mathbf{X}}{\text{argmax}} \sum_{i} q_i(\mathbf{x}; \lambda^{\text{learnt}}) U_i^d(x_i) \tag{4.3}$$

Behavior deception of attacker. [88] Since the (naive) defender uses the entire learning dataset to construct a single attacker model, a clever attacker might change its own behavior during the learning phase in order to benefit during the planning phase[1]. It is naturally assumed that only perfectly rational attackers display such deceptive behavior. Therefore, the partial behavior deception model centers on a single perfectly rational deceptive attacker, amongst the bounded rational attackers, that can alter some fraction of the training dataset. The bounded rational attackers attack non-deceptively according to a fixed unknown QR parameter λ^{true}. Essentially, the deceptive attacker wants to find the best perturbation of the training data to maximize its utility in the planning phase,

[1]In this paper, we focus on the one-shot game which only consists of a learning phase and planning phase—a commonly-used security game model in literature. Therefore, the deceptive attacker can simply play perfectly rationally in the planning phase after deceiving the defender in the learning phase. This model can also serve as the basis for repeated security games which involve multiple learning-planning rounds where the attacker plays deceptively in all rounds except the last round.

denoted by $U^a(\mathbf{x}^*(\mathbf{z}))$, as follows:

$$(\texttt{DecAlter}): \max_{\mathbf{z}=\{z_i^m\}} U^a(\mathbf{x}^*(\mathbf{z})) \tag{4.4}$$

$$\text{s.t. } z_i^m \geq n_i^m, \forall m, i \tag{4.5}$$

$$\sum_i z_i^m \leq (f+1) \cdot \sum_i n_i^m, \forall m. \tag{4.6}$$

where $\mathbf{x}^*(\mathbf{z})$ is the defender's strategy determined based on his learning-planning method in (4.2–4.3). In addition, n_i^m is the number of attacks by the non-deceptive attackers and $f \in \mathbb{R}$ is the ratio of deceptive attacks to non-deceptive attacks at each step m. Constraints (4.5–4.6) guarantee that the deceptive attacker can only control its own attacks. We denote by $\mathbf{z} = \{z_i^m\}$ the deception outcome of the deceptive attacker, which includes the non-deceptive attacks ($\mathbf{n} = \{n_i^m\}$). The defender learns a (deceptive) parameter λ^{learnt} using $\mathbf{z}$.

4.5 Cognitive Hierarchy Approach

In order to determine a counter-deception strategy for the defender, a possible approach is to compute a fixed point equilibrium of the deception game in which each player reasons about its opponent's strategy recursively till infinity. However, finding a fixed point equilibrium in our game is extremely challenging. This is because the defender has no information (or prior) about the behavior of the non-deceptive attackers. As a result, the defender has to relate the equilibrium outcome for every possible true behavior of these non-deceptive attackers to the observed (manipulated) attacks. This task is challenging (as well as impractical) given that the behavior space of attackers is infinite.

In real world settings, cognitive hierarchy models have been proven more effective than equilibrium based approaches at realistically modeling player behavior [19, 17, 49]. This is because human players do not exhibit infinite level

strategic reasoning. Cognitive hierarchy theory states that players in games can be divided into different *levels* of thinkers, each assuming that no players are on levels above them [149]. In a mixed attacker deception setting, we can model the levels as follows:

- Level 1: The rational attacker plays truthfully. The defender follows the two-stage learning-planning approach to compute a defense strategy.
- Level 2: The rational attacker plays deceptively, assuming the defender is at level 1. The level 2 defender, on the other hand, attempts to counter the attacker deception, assuming the attackers are at levels 0, 1, or 2.
- Level $l > 2$: The strategic reasoning is similar to level 2. Specifically, the attacker assumes the defender is at level $l-1$ while the defender assumes the attackers are at any one of the levels *up to and including* l.

Previous work has shown that distributions of human players in normal form games mostly consist of lower level players [149]. The aforementioned partial behavior deception model focuses on the deception by a level 2 attacker [88]. Our paper studies the counter-deception by a level 2 defender.

4.6 Finding Non-Deceptive Attacker Behavior

In order to determine an effective defense strategy, we begin our analysis by characterizing the space of *possible* attack behavior (described by $\mathtt{QR}$) of the non-deceptive attackers, given the perturbed data $\mathbf{z}$. Recall that the non-deceptive attackers respond according to a fixed λ^{true}, unknown to the defender. Instead, the defender obtains a learning outcome λ^{learnt} given perturbed training data. Our goal is to estimate the possible values of λ^{true} given observed learning outcome λ^{learnt}.

4.7 Characterizing Deceptive Attacker's Behavior

We first analyze the deception possibilities for the deceptive attacker *given any value* λ^{true} of the non-deceptive attackers. The results we establish here help us in our goal of estimating λ^{true}. For analysis sake, we assume that the number of attacks is large enough such that the sampled attacks is close to the actual attack probability distributions. We will relax this assumption later. Mathematically, we assume:

$$\left(n_i^m\right) / \left(\sum_j n_j^m\right) \approx q_i^m(\mathbf{x}^m, \lambda^{\text{true}}), \forall m \tag{4.7}$$

where n_i^m refers to the number of attacks committed by the *non-deceptive* attacker at target i. As shown in (`DecAlter`), the objective utility function of the deceptive attacker depends on the strategy of the defender, which in turn is governed by the training data $\{z_i^m\}$, and the training data contains attacks by the non-deceptive attacker too ($\{n_i^m\}$). Thus, the outcome of λ^{learnt} depends on the behavior of the non-deceptive attacker λ^{true} (or $\{n_i^m\}$). We thus also use the notion $\texttt{DecAlter}(\lambda^{\text{true}}) = \lambda^{\text{learnt}}$ to represent the dependence of the learning result (*altered* by deception) on λ^{true}.

For this portion of our analysis, we relax the domain of $\mathbf{z}$ to be continuous. This allows our proofs to be simpler and more concise. In practice, this value is limited to discrete integers; fractional attacks are nonsensical. Later, we will extend the methods to the discrete $\mathbf{z}$ case, and show why they still apply. We exploit the `KKT` condition for the optimality of the deceptive λ^{learnt} as the outcome of the defender's learning, formulated in optimization (4.2). Essentially, λ^{learnt} has to

satisfy the following KKT condition:

$$\sum_m \Big[\sum_i z_i^m\Big]\Big[\frac{\sum_i z_i^m U_i^a(x_i^m)}{\sum_i z_i^m} - \underbrace{\sum_i q_i(\mathbf{x}^m;\lambda^{\text{learnt}})U_i^a(x_i^m)}_{\text{Attacker utility } U^a(\mathbf{x}^m;\lambda^{\text{learnt}})}\Big] = 0$$

where $U^a(\mathbf{x}^m;\lambda^{\text{learnt}})$ is the attacker's expected utility when the defender plays $\mathbf{x}^m$ and the attacker plays according to λ^{learnt}. In our theoretical analysis, we leverage the following important monotonicity property of this utility function:

Observation 1 ([89]). *$U^a(\mathbf{x}^m,\lambda)$ is an increasing function of λ for any given strategy $\mathbf{x}^m$.*

Let's assume, WLOG, the attacker's utilities at each target has the following order: $U_1^a(x_1^m) \le U_2^a(x_2^m) \le \ldots \le U_T^a(x_T^m)$ for all m. Observation 1 aids us in showing that all feasible (not necessarily optimal) deceptive λ values form an interval $[\lambda_{\min}^{\text{learnt}}, \lambda_{\max}^{\text{learnt}}]$ with $\lambda_{\min}^{\text{learnt}}$, $\lambda_{\max}^{\text{learnt}}$ specified as follows:

Theorem 1 (Characterization of Deception Space). *Given λ^{true} and the attack ratio f, the space of deceptive parameters inducible by the deceptive attacker forms an interval $[\lambda_{\min}^{\text{learnt}}, \lambda_{\max}^{\text{learnt}}]$, where $\lambda_{\max}^{\text{learnt}}$ is the unique solution of:*

$$\sum_{m,j} n_j^m\big[U^a(\mathbf{x}^m;\lambda^{\text{true}}) + fU_T^a(x_T^m) - (f+1)U^a(\mathbf{x}^m,\lambda_{\max}^{\text{learnt}})\big] = 0$$

and $\lambda_{\min}^{\text{learnt}}$ is the unique solution of:

$$\sum_{m,j} n_j^m\big[U^a(\mathbf{x}^m;\lambda^{\text{true}}) + fU_1^a(x_1^m) - (f+1)U^a(\mathbf{x}^m,\lambda_{\min}^{\text{learnt}})\big] = 0$$

All formal proofs are in the appendix. Essentially, Theorem 1 states that given some true behavior of the non-deceptive attacker λ^{true}, the deceptive attacker can force the deceptive λ to be any value in $[\lambda_{\min}^{\text{learnt}}, \lambda_{\max}^{\text{learnt}}]$. Further, the deceptive attacker cannot make the defender learn any λ outside of this range. Based on Theorem 1, we present the following corollaries which characterize the

monotonicity of $\lambda_{\min}^{\text{learnt}}$ and $\lambda_{\max}^{\text{learnt}}$, as well as the monotonicity of the optimal deception $\lambda^{\text{learnt}} = \texttt{DecAlter}(\lambda^{\text{true}}) \in [\lambda_{\min}^{\text{learnt}}, \lambda_{\max}^{\text{learnt}}]$ with respect to the non-deceptive attacker behavior λ^{true}.

Corollary 1. *Consider two different behavior parameters,* $\lambda_1^{\text{true}} \leq \lambda_2^{\text{true}}$. *Denote by* $[\lambda_{\min,1}^{\text{learnt}}, \lambda_{\max,1}^{\text{learnt}}]$ *and* $[\lambda_{\min,2}^{\text{learnt}}, \lambda_{\max,2}^{\text{learnt}}]$ *the corresponding deceptive parameter ranges, we have:* $\lambda_{\max,1}^{\text{learnt}} \leq \lambda_{\max,2}^{\text{learnt}}$ *and* $\lambda_{\min,1}^{\text{learnt}} \leq \lambda_{\min,2}^{\text{learnt}}$.

Based on Corollary 1, we obtain Corollary 2 showing the monotonicity relation between λ^{learnt} and λ^{true}.

Corollary 2. *Consider two different behavior parameters,* $\lambda_1^{\text{true}} \neq \lambda_2^{\text{true}}$. *Then, we have:*

$$\lambda_1^{\text{true}} \leq \lambda_2^{\text{true}} \Longrightarrow \texttt{DecAlter}(\lambda_1^{\text{true}}) \leq \texttt{DecAlter}(\lambda_2^{\text{true}}) \tag{4.8}$$

$$\texttt{DecAlter}(\lambda_1^{\text{true}}) < \texttt{DecAlter}(\lambda_2^{\text{true}}) \Longrightarrow \lambda_1^{\text{true}} < \lambda_2^{\text{true}} \tag{4.9}$$

Corollary 3. *Consider two different behavior parameters* $\lambda_1^{\text{true}} \leq \lambda_2^{\text{true}}$. *If the corresponding optimal deception solutions:* $\texttt{DecAlter}(\lambda_1^{\text{true}}) = \texttt{DecAlter}(\lambda_2^{\text{true}})$, *then for any* $\lambda^{\text{true}} \in [\lambda_1^{\text{true}}, \lambda_2^{\text{true}}]$, *we also have its optimal deception solution:* $\texttt{DecAlter}(\lambda^{\text{true}}) = \texttt{DecAlter}(\lambda_1^{\text{true}})$.

4.8 RaBiS: Characterizing Behavior of Non-Deceptive Attacker

In this section, we attempt to find the range of possible values for λ^{true}, which is unknown to the defender, as only the deceptively altered `QR` parameter λ^{learnt} is observed. We leverage the results of Corollaries 2 and 3 for this analysis.

Lemma 1. *Given some learned* λ^{learnt}, *there exists an interval* $[\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}]$ *such that all values* $\lambda^{\text{true}} \in [\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}]$ *leads to the same outcome* λ^{learnt}. *In addition, both bounds* $\lambda_{\min}^{\text{true}}$ *and* $\lambda_{\max}^{\text{true}}$ *are increasing in* λ^{learnt}.

Based on the above result, we propose a binary-search based approach, `RaBiS` (**Ra**nge-finding **Bi**nary **S**earch), to find the interval $[\lambda^{\text{true}}_{\text{min}}, \lambda^{\text{true}}_{\text{max}}]$ within an ϵ-error in a polynomial time for arbitrary small $\epsilon > 0$. `RaBiS` consists of two binary searches: the first binary search is to find the upper bound $\lambda^{\text{true}}_{\text{max}}$ and the second binary search is to find the lower bound $\lambda^{\text{true}}_{\text{min}}$. Both binary searches maintain a pair of bounds for binary search (lb, ub). While in theory the range of λ^{true} is $[0, \infty)$, in practice, a limited range of $[0, M]$, where M is a very large constant, ensures that the attacker's `QR` behavior with $\lambda^{\text{true}} = M$ is close enough to $\lambda^{\text{true}} = \infty$. Therefore, in our algorithm, we initialize $lb = 0$ and $ub = M$.

At each iteration, we examine the mid-value $r = (lb + ub)/2$ by comparing the deception calculation $\lambda' = \texttt{DecAlter}(r)$ with the actual deception outcome computed by the defender, λ^{learnt}. In particular, in the binary search for finding $\lambda^{\text{true}}_{\text{max}}$, if $\lambda' \leq \lambda^{\text{learnt}}$, there must be a $\lambda^{\text{true}}_{\text{max}} \in [r, ub]$ such that $\texttt{DecAlter}(\lambda^{\text{true}}_{\text{max}}) = \lambda^{\text{learnt}}$ and any $\lambda > \lambda^{\text{true}}$ implies $\texttt{DecAlter}(\lambda) > \lambda^{\text{learnt}}$. Thus, in order to find $\lambda^{\text{true}}_{\text{max}}$, we update the lower bound $lb = r$. Conversely, if $\lambda' > \lambda^{\text{learnt}}$, it means all $\lambda^{\text{true}} \in [r, ub]$ will lead to a deceptive parameter value strictly greater than λ^{learnt}. Therefore, we update the upper bound $ub = r$. This process stops when $ub - lb < \epsilon$. The binary search process for finding $\lambda^{\text{true}}_{\text{min}}$ is similar.

4.9 Principled Approach for Low-Data Challenge

Thus far, our analysis of the range of the non-deceptive attacker λ^{true} was performed under the approximation assumption of Equation 4.7. However, in practice, this assumption may not hold true. This is because the attacker may conduct a limited number of attacks, which leads to a substantial difference

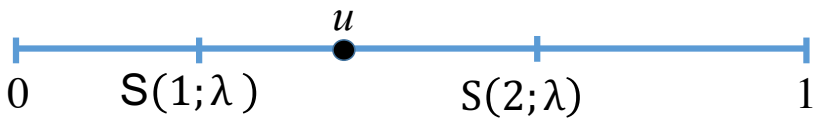


Figure 11. Attack generation by transforming uniform dist.

between the empirical attack distribution and the true attack distribution, that is:

$$\left(n_i^m\right) / \left(\sum_j n_j^m\right) \neq q_i^m(\mathbf{x}^m; \lambda^{\text{true}}), \forall m$$

To address this challenge, we first investigate the generation of limited attack samples from the true distribution under a *static random seed*. We show that our previous theoretical results for the ideal scenario still hold in this "limited-attack" scenario. We then leverage this result for a static random seed to address the general case of *unknown* random seed.

Sampling by transformation. Sample generation from certain parameterized distributions can be split into a two step process by using a transformation of known distributions [108, 60]. We show that such split generation is possible for our problem. Let u be a real valued random variable that is distributed uniformly between 0 and 1. Given a defense strategy, $\mathbf{x}^m$, and `QR` parameter λ, we define the function f_λ such that $P\big(f_\lambda(u) = i\big) = q_i(\mathbf{x}^m; \lambda)$. Note that f_λ is a deterministic function dependent on λ, which we define explicitly next. For any given $\mathbf{x}^m$, partition the interval $[0, 1]$ according to the attack probabilities $q_i(x^m; \lambda)$ specified by `QR` with parameter λ, with the following partition boundary points: $S(0; \lambda) = 0$, $S(i; \lambda) = \sum_{j=1}^{i} q_j(\mathbf{x}^m; \lambda)$, and $S(T; \lambda) = 1$. Figure 11 is an example when the number of targets is $T = 3$. Given this division, we define $f_\lambda(u) = i$ when $u \in [S(i-1; \lambda), S(i; \lambda)]$; it can be readily verified that $P\big(f_\lambda(u) = i\big) = q_i(\mathbf{x}^m; \lambda)$. In the case of $N > 1$ attacks, we can view the attack generation process as N samples

of u to get $\mathbf{u} = \{u_1, \ldots, u_N\}$ and then applying f_λ to each of those samples to obtain the targets attacked.

Static random seed generation. For our problem with parameter λ^{true}, after separating the randomness (u) and the effect of the parameter ($f_{\lambda^{\text{true}}}$) in attack generation, the main idea of a static random seed is to assume that the N uniformly sampled values $\mathbf{u}$ are the same for any value of λ^{true} that we consider in the binary search for $\lambda^{\text{true}}_{\min}$ or $\lambda^{\text{true}}_{\max}$. By controlling the randomness, we establish a deterministic baseline to compare the empirical distribution arising from the different λ^{true} that we consider. A big advantage of controlling randomness is that it allows us to carry over all the previous proofs to a low data setting, as described next.

Let $E(\mathbf{u}, \lambda^{\text{true}})$ be the empirical distribution when attacks are computed using $f_{\lambda^{\text{true}}}$ and the generated N samples $\mathbf{u}$. We can define the attacker expected utility w.r.t. this distribution, denoted by $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$, exactly analogously to how $U^a(\mathbf{x}^m; \lambda^{\text{true}})$ is defined w.r.t. the true distribution. We obtain Lemma 2 which is analogous to Observation 1.

Lemma 2. *For a fixed seed,* $\mathbf{u}$*, the attacker expected utility computed based on the corresponding empirical distribution,* $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$*, is an increasing function of* λ^{true}*.*

In all results previously (including corollaries), we only used the Observation 1 property of $U^a(\mathbf{x}^m; \lambda^{\text{true}})$. With the result above, we can replace $U^a(\mathbf{x}^m; \lambda^{\text{true}})$ by $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$ and all proofs still go through. Hence, our Theorem 1 holds with respect to $U^a(\mathbf{x}^m; E(\mathbf{u}, \lambda^{\text{true}}))$ (which replaces $U^a(\mathbf{x}^m; \lambda^{\text{true}})$

in the equations presented in Theorem 1). This result shows that for a fixed random seed $\mathbf{u}$ we can recover all previous results.

Extension to unknown random seed. The random seed used (by nature) in the generation of the training data is not known to the defender. To overcome this challenge, we extend our binary search to consider multiple random seeds. For each random seed, we run `RaBiS` to obtain an interval of possible values for λ^{true}. Taking a worst-case approach, we consider the smallest interval that spans all of these ranges as the uncertainty set containing all possible values of λ^{true}.

4.10 Maximin to Optimize Defender Utility

After finding the range $[\lambda^{\text{true}}_{\text{min}}, \lambda^{\text{true}}_{\text{max}}]$, the defender must optimize its strategy accordingly. Essentially, the defender is aware that there are attacks not only from a rational (deceptive) attacker (who will act optimally in the defender's planning phase) but also from bounded rational attackers (whose λ^{true} can be any value within $[\lambda^{\text{true}}_{\text{min}}, \lambda^{\text{true}}_{\text{max}}]$). In order to overcome the uncertainty about the behavior of these attackers, we take a maximin approach where the defender seeks to *maximize* its utility against the *worst* possible (for the defender) λ value within the calculated range. In practice, to deal with the computational challenge due to an infinite number of possible values in $[\lambda^{\text{true}}_{\text{min}}, \lambda^{\text{true}}_{\text{max}}]$, we break down this range into a set of possible discrete values $\{\lambda^{\text{true}}_{\text{min}}, \lambda^1, \lambda^2, \ldots, \lambda^{\text{true}}_{\text{max}}\}$. Furthermore, since the rational attacker will choose an optimal target to attack in the planning phase, we decompose our defense problem into multiple non-linear programs, each corresponds to a particular optimal target to attacker for the rational attacker. In particular, our non-linear program corresponding to an optimal target j can be

formulated as follows:

$$\max_{\mathbf{x}} f \cdot U_j^d(x_j) + U^d_{\text{worst-case}} \tag{4.10}$$

$$\text{s.t. } U_j^a(x_j) \geq U_i^a(x_i), \forall i \tag{4.11}$$

$$U^d_{\text{worst-case}} \leq \sum\nolimits_i q_i(\mathbf{x}; \lambda) U_i^d(x_i), \tag{4.12}$$

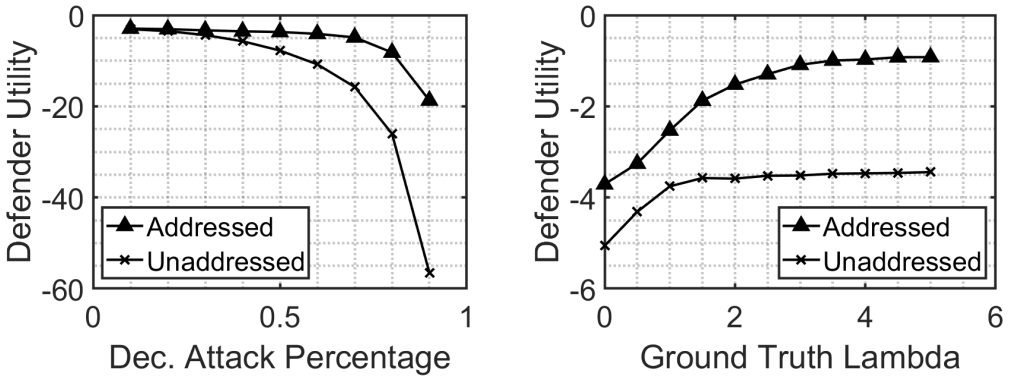
$$\forall \lambda \in \{\lambda^{\text{true}}_{\text{min}}, \lambda^1, \lambda^2, \ldots, \lambda^{\text{true}}_{\text{max}}\}$$

$$\sum\nolimits_i x_i \leq K, x_i \in [0,1], \forall i \tag{4.13}$$

The objective (line 4.10) balances optimization against the fully rational attacker, $U_j^d(x_j)$, and the worst possible bounded rational attacker, $U^d_{\text{worst-case}}$, with multiplier f corresponding to the ratio of deceptive to non-deceptive attacks. Constraint (4.11) ensures that the target chosen by the fully rational attacker, j, is indeed the highest-utility target. Constraint (4.12) effectively iterates through the λ range, setting $U^d_{\text{worst-case}}$ equal to the lowest defender utility value among all possible lambdas. In a zero sum game, these lines could be replaced by simply setting $\lambda = \lambda^{\text{true}}_{\text{max}}$. Lastly, constraint (4.13) provides logical bounds to the defender's strategy: the total coverage percentage of all targets cannot exceed the number of resources, and all targets have coverage probability between 0 and 1.

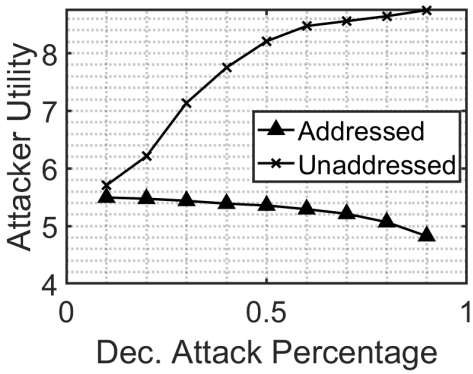
4.11 Experiments

In our experiments, we analyze: (i) the defender's utility gain by addressing deception, and (ii) the loss of utility for the devious attacker. The training data includes attacks from both the fully rational deceptive attacker and a boundedly rational attacker whose behavior is described by `QR`. We use 5 defender training strategies ($M = 5$) each with 50 non-deceptive attacks ($\sum_i n_i^m = 50$) sampled from the `QR` distribution with λ^{true} of the bounded rational attacker. Each data point is averaged over 200+ games, generated using GAMUT (http://gamut.stanford.edu).

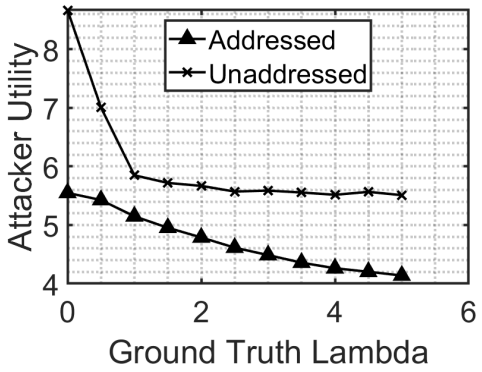


(a) Vary % of dec. attacks

(b) Vary λ^{true}



(c) Vary % of dec. attacks



(d) Vary λ^{true}

Figure 12. Players Utility Evaluation

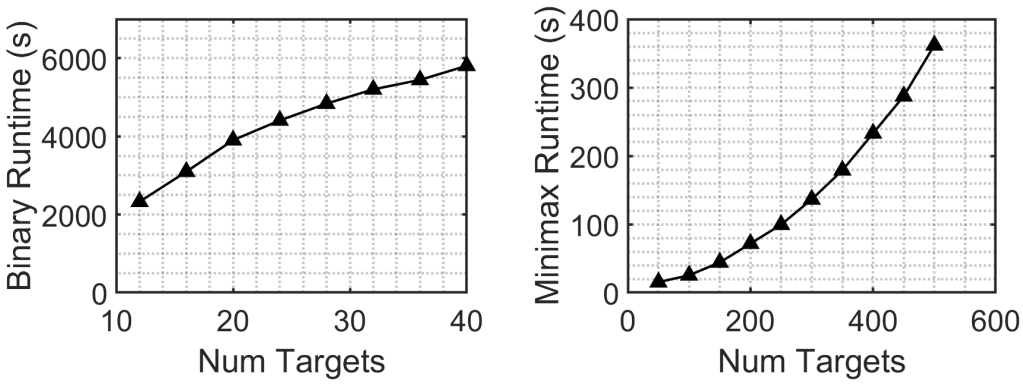
For our trials, we vary (i) the true non-deceptive lambda λ^{true} value and (ii) the fraction f of attacks done by the devious adversary. Due to limited space, we will only highlight important results. Additional results are included in our appendix. All utility results are statistically significant under bootstrap-t (α=0.05) [145].

Figures 12a and 12b display the defender's utility in two cases: (i) `Addressed` — the defender addresses the attacker's deception using our counter-deception algorithm; and (ii) `Unaddressed` — the defender simply does not take the attacker's deception into account. In these two figures, the y-axis represents the defender's expected utility on average. Both figures show that the defender can significantly increase his utility for playing our maximin counter-deception strategy.

In Figure 12a we observe that, when deception is unaddressed, the defender's utility decreases exponentially as the deceptive attack ratio increases. On the other hand, when the defender *does* address deception, the slope is far more gradual. Figure 12b shows how defender utility increases as the non-deceptive λ^{true} value does. This effect tapers off on the upper end of the spectrum. This result is expected because the non-deceptive attacker gets more rational as λ^{true} increases, leading to less changes in the defender's maximin strategy. Furthermore, in Figure 12b, the lowest utility point for the defender is when λ^{true} gets to zero. This makes sense: as the non-deceptive attackers become completely non-strategic ($i.e., \lambda^{\text{true}} = 0$), the non-deceptive attackers will have less influence on the training data, or equivalently, the deceptive attacker has more power to manipulate the data.

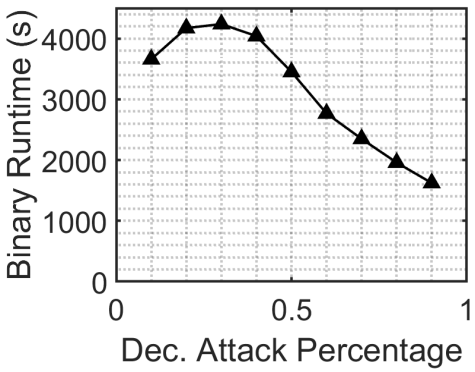
Naturally, we observe an opposite trend in the attacker-utility graphs shown in Figures 12c and 12d. That is, the utility of the attacker reduces substantially when the defender addresses the attacker deception. Figure 12c shows that when the defender plays our maximin strategy, the attacker's utility actually decreases w.r.t. the percentage of attacks controlled by the deceptive attacker. This result appears to be counter-intuitive at first glance. However, it's logical: our maximin algorithm knows the attack ratio so it tailors more of the defense strategy towards a fully rational attacker (the actual rationality of the deceptive attacker).

Lastly, we analyze runtime performance of both portions of the algorithm in Figure 13. For the binary search, runtime is high across the board due to the sheer number of partial deception games (`DecAlter`) solved in each search. However, this runtime scales linearly w.r.t. the number of targets (Figure 13a), implying that the algorithm can be scaled to large games. Furthermore, when varying the attack percentage (Figure 13c), we see that the runtime peaks with a percentage

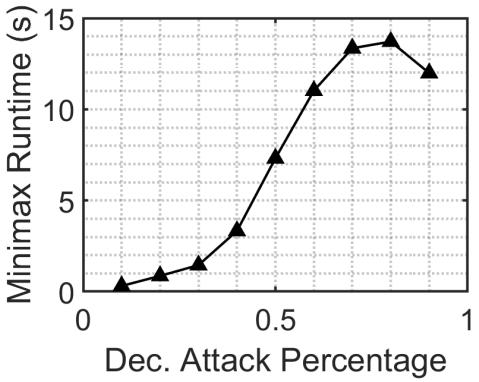


(a) Binary Search Runtime

(b) Maximin Runtime



(c) Binary Search Runtime



(d) Maximin Runtime

Figure 13. Runtime Evaluation

around 0.3. This peak is shifted compared to the runtime for solving (`DecAlter`) only, which peaks around 0.5 [88]. This is because the range, $[\lambda_{\min}^{\text{true}}, \lambda_{\max}^{\text{true}}]$ increases as the deceptive attack percentage does, meaning the total search time decreases as `RaBiS` exits earlier.

Figure 13b shows how the maximin runtime increases w.r.t. the number of targets. This is expected since the number of non-linear programs involved is equal to the number of targets. The maximin optimization can scale to large games: 500 target games are solved in less than 10 minutes. Observe that we examine a larger spread of targets here than for the binary search portion of the algorithm; the binary search runtime is orders of magnitude higher, reaching our

100 minute cut-off with far fewer targets. Figure 13d shows that maximin runtime initially increases as the percentage of attacks that are deceptive does, reflecting the wider range of possible values for λ^{true}. At higher values this effect diminishes and runtime ends up decreasing at the 0.9 marker, indicating that it is easier to optimize a strategy against mostly rational attacks.

4.12 Conclusion

We successfully addressed attacker deception in security games, showing both theoretically and experimentally the value of our approach. Through mathematical analysis we explored the characteristics of deception and defense and developed effective countermeasures: `RaBiS` allowed the defender to see through the deceptively altered historical attack data, after which a maximin approach yielded a robust strategy. Our experiments showed the wary defender receiving much higher utility than its naive counterpart.

CHAPTER V

USING REINFORCEMENT LEARNING FOR DATA-BASED DECISION MAKING IN PUBLIC HEALTH MESSAGING

Acknowledgment. This chapter is adapted from a paper published in IJCAI-23 (focusing on the initial deployment of our RL system), and a paper that is currently under review at AAAI (focusing on improving the RL system).

For the first paper, I was a joint first author, along with Jack Wolf. Jack was the primary researcher at the start of the project, and was responsible for the real-world RL system design, implementation, and deployment. Following his departure from the program, I took over the project. I was responsible for mid and post-deployment analysis, including the behavior modeling work. The writing is a result of collaboration between myself and the other academic co-authors: Professor Arunesh Sinha, and Professor Thanh H. Nguyen. Lastly, the real-world deployment of this system was done in collaboration with Arogya World, an NGO in India. The credited co-authors from Arogya World are: Nalini Saligram, Varun Ramesan, Meeta Walavalkar, and Nidhi Jaswal.

For the second paper, I was the first author, and I performed all the programming and experimental work. Writing, as well as the theoretical analysis, was a group effort between myself and my co-authors: Professor Arunesh Sinha, and Professor Thanh H. Nguyen.

Non-communicable diseases (NCDs) such as cardiovascular disease, diabetes, and cancer, are among the top health challenges of the century. According to WHO, NCDs kill 41 million people each year, equivalent to 74% of all deaths globally. Notably, 85% of premature deaths from NCDs occur in low- and middle-income countries [93]. Fortunately, these serious diseases are largely preventable.

According to WHO, NCDs are preventable with three lifestyle changes: eat healthy food, increase physical activity, and avoid tobacco. Yet barriers in underprivileged regions often prevent engagement in these activities, particularly poor education about the disease, lack of social support, and limited healthcare access.

We propose to overcome some of these barriers by building a new AI-based system which can help in improving diabetes risk behavior of people in such underprivileged regions. We build an effective messaging intervention system, that dynamically sends personalized messages to participants (through Whatsapp). These messages contain information about diabetes causes and complications, and the impact of nutrition and fitness on preventing diabetes. We chose Whatsapp to transmit our messages since mobile phone uptake is high in India, and Whatsapp is an essential communication channel that is accessible to almost everyone. Existing non-profit healthcare programs often pre-design fixed non-personalized messages [104, 112]. Our work seeks to improve this using AI, leveraging techniques in RL to optimize our message selection policy and tailoring sent messages to align with each individual participant's needs and preferences.

We provide four main contributions. First, we build an online diabetes-targeted intervention system that automatically sends out messages to participants in our study on a weekly basis. Each week, messages sent to each participant are determined based on behavior dynamics of the participant. In the same week, our system collects information about changes in behavior of participants through a question/answer mechanism, which is leveraged to improve our message generation in later weeks. Second, we model the problem of optimizing message generation as a RL problem and develop a RL algorithm for solving this problem, tackling concrete real-world challenges exhibited in our problem domain. Our algorithm is

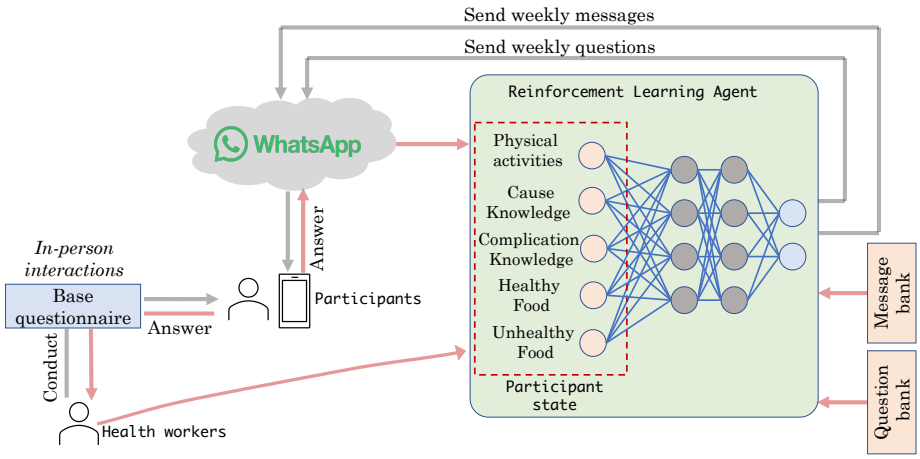


Figure 14. Personalized message-based intervention system overview

an extension of DQN [78], a well-known RL method, with adaptations to handle simultaneous interactions with multiple participants in an online learning fashion.

Third, we run an extensive field study that involves over 1000 participants from local villages that received messages generated by our system for more than six months. Our analysis shows that there are significant improvements in the participants' diabetes-related knowledge, physical activities, and high-fat food avoidance at the end of our field study. Finally, we build a new neural net-based behavior model to predict participants' behavior changes, leveraging the data collected during our study. We show that our model, which exploits inherit differences in behavior changes across different types (knowledge, physical activity, and food consumption) obtains the best prediction accuracy compared to baselines.

5.1 Personalized Message-Generation System

In this chapter, our goal is to optimize the impact of our message intervention program on participants' lifestyle behavior. The challenge is that participants have varied lifestyle and also may have various kinds of reaction to our messages. Therefore, it is important that we can personalize the message

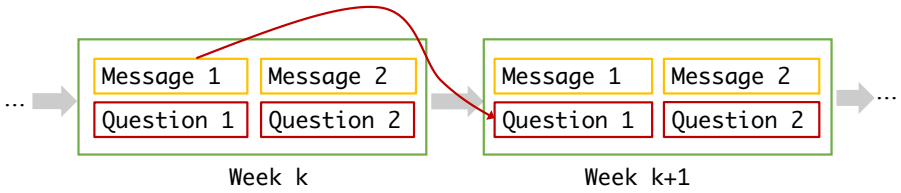


Figure 15. Weekly message/question flow

Message example on week k: (Fitness) You can help avoid diabetes by being physically active. Walk to the temple or shops, climb stairs. Walk briskly or exercise for 30 mins daily.

Question example on week k+1: In last week, how often would you have done any form of exercise (Yoga/Running/Jogging/In Gym/Aerobics etc.)?
Answer format: 1. Daily days 3. Selective days/weekends
2. Alternate 4. Never

Figure 16. An example of weekly messages and questions

selection policy for each individual participant. We propose to apply techniques in reinforcement learning to serve this purpose. Overall, given a message bank as an input, our RL-based system runs for a number of rounds (i.e., weeks). In each round, our system selects a pair of messages for each participant and sends out these messages to the participants. In addition, the system sends out a separate pair of questions to each participant and collect answers from them. These questions collect information regarding the changes in the participants' lifestyle on a weekly basis. Our system then uses the participants' responses to these questions as feedback to update our message selection policy. The overview of our system is illustrated in Figure 14.

Our messaging process has three distinct phases. In phase one of participant sign-up, health workers visit villagers in person. The health workers (i) sign up people for receiving our health-related messages; (ii) provide detailed instructions of the program to the participant; and (iii) distribute initial questionnaire and collect answers from participants. The initial questionnaire includes questions about

the participants' demographics, family health-related history, knowledge about diabetes, physical activities, and food intake, etc. We use participants' responses to this questionnaire to build the initial state of participants.

In more detail, each participant's *state* consists of "scores" for the five categories: (a) healthy food intake, (b) unhealthy food/tobacco/alcohol intake, (c) fitness/physical activity level, (d) diabetes cause knowledge, and (e) diabetes complication knowledge. The scores take value in $\{1, 2, 3\}$ where 1 means "Low", 2 means "Medium", and 3 represents "High". For example, the state for a participant can be $(1, 3, 2, 2, 3)$ where the score for the health food intake of this participant is 1, it means that this participant rarely consumes healthy food such as vegetables and fruits. Thus, this score implies that this participant should receive messages that encourage the participant to eat more healthy foods.

In phase two, participants receive two messages and two questions each week, each message or question targets one of the above five categories. The sets of messages and questions are carefully designed in local language by domain experts. In this phase, our message-based intervention system interacts with participants via Whatsapp. The flow of the weekly interactions with each participant is illustrated in Figure 15 and an example of weekly messages/questions is provided in Figure 16. As we show in Figure 16, the purpose of the questions asked in each week is to determine the impact of the messages sent out in the prior week on the participants' behavior change. We remark that our system sends out only two messages and two questions on two days per week (i.e., Tuesdays and Fridays and one message/question per day) for the sake of participants' comfort. Overloading the participants with messages/questions can potentially disrupt participants' daily activities, causing unnecessary burden, decreasing their engagement in our

program. Finally, responses of participants to our questions is used to update the participants' state. Our system then leverages the participants' updated state to generate new messages and questions in the following week.

Finally, in phase three we perform a post-study analysis, in which our health workers meet participants in person again and conduct the same questionnaire as in phase one. Our goal is to have a complete comparison of the behavior changes in beneficiaries before and after participation in our program.

5.2 RL-based Message Generation Algorithm

The core of our message-based intervention system is the RL agent that actively selects weekly messages and questions for individual participants in a personalized manner based on their previous responses. We adopt ideas from Deep Q-Learning (DQN) [78], a well-known RL method in literature, to build our RL agent. However, we face the following challenges. *First*, RL methods are effective typically when they are pre-trained before the actual real-world deployment. However, in this setting, there is no historical participant data that can be used for pre-training RL methods. Our RL agent has to be trained directly on the job during limited weekly interactions across only 25 weeks. *Second*, ideally, we can treat each participant as a separate Markov Decision Process and train a separate RL agent for each participant. However, our system can only obtain a single trajectory of state transition for each participant, which is extremely limited information for building an RL agent for each individual. Conversely, a single RL agent for the whole population may fail to capture the diverse behavior of participants. *Third*, traditionally, RL models are iteratively trained by sequentially acquiring different episodes of interactions with the environment. However, here our system interacts with multiple participants in parallel across a single episode.

Algorithm 2: Adaptive Message Generation.

Calculate participant initial states based on their responses to the base questionnaire $\{s_0^{(i)}\}$;
Warm-up step: Pre-train action-value network Q with parameters θ based on initial states of participants;
Initialize target action-value network $\widehat{Q}$ with $\widehat{\theta} = \theta$;
for $t = 1 \rightarrow T$ **do**
  for $i = 1 \rightarrow N$ **do**
    With prob. ϵ, select a random action (i.e., a pair of messages) $a_t^{(i)}$ for participant i;
    With prob. $1 - \epsilon$, select $a_t^{(i)} = \text{argmax}_a \, Q(s_{t-1}^{(i)}, a, \theta)$;
    if $t > 1$ **then**
      Send questions $q_{t-1}^{(i)}$ and obtain $\text{answer}_{t-1}^{(i)}$;
      Update participant state: $s_t^{(i)} = \text{Update}\left(s_{t-1}^{(i)}, q_{t-1}^{(i)}, \text{answer}_{t-1}^{(i)}\right)$;
      Calculate reward $r_{t-1}^{(i)}$ based on state update and add transition $\left(s_{t-1}^{(i)}, a_{t-1}^{(i)}, s_t^{(i)}, r_{t-1}^{(i)}\right)$ to D;
  for $n = 1 \rightarrow \textit{numUpdate}$ **do**
    Sample random minibatch of transitions (s_j, a_j, s_{j+1}, r_j) from D;
    Perform a gradient descent step to update θ on $\left[r_j + \gamma \max_{a'} \widehat{Q}\left(s_{j+1}, a', \widehat{\theta}\right) - Q\left(s_j, a_j, \theta\right)\right]^2$;
  if $t \mod \textit{step} = 0$ **then** Update $\widehat{\theta} = \theta$;

Given these real-world deployment challenges, we create a new variant of DQN with the following revisions. *First*, we provide a warm-up stage in which we "pre-train" our model; we leverage the participants' responses to the initial questionnaire to initialize values of learnable parameters of our model. Intuitively, these values are determined such that our model generates messages for each participant that target state categories in which the participant has low scores. For example, if the participant has low physical activity level (i.e., score is 1), the model will likely select a message that encourages the participant to do more physical exercise. *Second*, based on initial state scores, we use a clustering

method to divide participants into three groups such that participants in the same group have similar initial states. We aim to train a single RL agent for each group, anticipating similar in-group behavior. Note that this means the RL agent is trained based on data collected from all participants within a group; however, messages and questions selected by the RL agent for each participant are determined by their individual state. By doing so, we use enough data to train our model while still personalizing messages. *Third*, we modify the DQN training process to allow multiple updates of model training in each step (a week) and simultaneous state and message updates for all participants. The details are presented in Algorithm 2, which runs separately for each participant group.

Essentially, in each week t, each participant i is associated with a state $s_{t-1}^{(i)}$ which represents the latest status of the participant lifestyle behavior as we discussed previously in Section 5.1. The goal of Algorithm 2 is to train a neural net model with unknown parameters θ to predict the q-value $Q(s_{t-1}^{(i)}, a, \theta)$ of the state-action pair $(s_{t-1}^{(i)}, a)$. Here, an action a represents a pair of messages selected from the weekly message bank. Intuitively, the q-value $Q(s_{t-1}^{(i)}, a, \theta)$ is the total expected reward that we receive if the action a is chosen for participant i given the latest state value $s_{t-1}^{(i)}$. This total expected reward captures the long-term impact of the selected action on the behavior change of participant i in the future.

Warm-up. In the warm-up phase, we pre-train our neural net model by minimizing the following MSE:

$$\theta^{init} \in \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{a \in A} \Big[Q(s_0^{(i)}, a, \theta) - \text{init.value}(s_0^{(i)}, a) \Big]^2$$

where $\text{init.value}(s_0^{(i)}, a)$ is the estimated importance score of action a for participant i given the participant initial state is $s_0^{(i)}$. Note that this initial state is estimated

based on the participant's response to the questionnaire. The score init.value($s_0^{(i)}, a$) is determined such that this value will be high if the action a includes messages that target categories that the participant has low scores, and vice versa. For example, if the participant has score 1 for the physical activity category in $s_0^{(i)}$, the value of init.value($s_0^{(i)}, a$) is 3 if the action a has physical activity-related messages. Here, N is the total of participants in the considering group and A is the set of all possible actions (i.e., pairs of messages).

Weekly message/question selection. In the weekly message/question phase, at every week t, for every participant i, with a probability of $\epsilon > 0$, we select an action $a_t^{(i)}$ for participant i uniformly at random. And with a probability of $1-\epsilon$, we select the optimal action $a_t^{(i)} = \text{argmax}_a\, Q(s_{t-1}^{(i)}, a, \theta)$. This ϵ-greedy approach allows us to balance between exploration and exploitation during the learning process. We then send a pair of questions $q_{t-1}^{(i)}$. These questions are used to measure the impact of messages the participants received in the last week $t-1$. We remark that if we receive responses from participants for these questions quickly, we can immediately use these responses to update participants' state and then generate new messages for this week. However, this is not the case due to delayed updates, meaning that we have to send out messages for this week prior to receiving responses. Therefore, we have the following message/question procedure:

- Week $t = 1$: we send out messages $a_1^{(i)}$ based on $s_0^{(i)}$.
- Week $t = 2$: we send out messages $a_2^{(i)}$ based on state $s_1^{(i)} = s_0^{(i)}$. We then send out questions $q_1^{(i)}$ regarding impact of $a_1^{(i)}$ and receive responses $\text{answer}_1^{(i)}$. We update state $s_2^{(i)} = \text{Update}(s_1^{(i)}, q_1^{(i)}, \text{answer}_1^{(i)})$.

– Week $t = 3$: we send out messages $a_3^{(i)}$ based on state $s_2^{(i)}$. We send out questions $q_2^{(i)}$ regarding impact of $a_2^{(i)}$ and receive responses. We update state $s_3^{(i)} = \text{Update}(s_2^{(i)}, q_2^{(i)}, \text{answer}_2^{(i)})$, and so on.

Note that, in reality, participants may not respond every week. When we do not receive any answer from a participant, there will be no state change for that participant, i.e., $s_t^{(i)} = s_{t-1}^{(i)}$.

Model training update. Similar to DQN, we maintain a replay buffer D of historical interactions with participants. At every step t, transitions $\left(s_{t-1}^{(i)}, a_{t-1}^{(i)}, s_t^{(i)}, r_{t-1}^{(i)}\right)$ of every participant i will be added to D. Here, the reward $r_{t-1}^{(i)}$ is calculated based on the state change $(s_{t-1}^{(i)}, s_t^{(i)})$. For example, if the physical activity score part of the state is updated from a value of 1 in $s_{t-1}^{(i)}$ to a value of 2 in $s_t^{(i)}$, then the reward $r_{t-1}^{(i)} = 1$. This reward value indicates that the action $a_{t-1}^{(i)}$ had positive impact on the participant i's exercise behavior. If multiple categories in the state have their scores changed, then the reward is computed as the sum of rewards over all these categories. This buffer D is used to update the neural net parameters θ. In addition to D, we also maintain a target network $\widehat{Q}$ with target parameters $\widehat{\theta}$. The values of the parameters $\widehat{\theta}$ are updated periodically based on the network Q. The replay buffer D and target network $\widehat{Q}$ are the main ideas of DQN that help in stabilizing and improving the q-learning process. Finally, at each week t, after updating D with new transitions, we run a number of iterations *numUpdate* to perform gradient descent updates on θ based on the loss $\left[r_j + \gamma \max_{a'} \widehat{Q}\left(s_{j+1}, a', \widehat{\theta}\right) - Q\left(s_j, a_j, \theta\right)\right]^2$ which is computed based on a mini-batch of transitions (s_j, a_j, s_{j+1}, r_j) sampled from D.

5.3 Real-world Deployment

For real world deployment, our primary goals were to lay the groundwork for our message intervention program. These included completing all the required preparations, obtaining IRB approval, working with domain experts to design a bank of messages and questions, completing front-line worker training, collecting baseline data, and testing the transmission system. In addition, our critical objectives included recruiting beneficiaries and commencing the message transmission.

In January and February of 2022, we successfully set up the automated messaging pipeline by linking our Google Cloud VM to our partner's cloud storage, allowing for seamless and automatic transmission of messages and questions selected by our RL system to participants. We also completed Facebook business verification, obtained approval for WhatsApp message transmission, and developed the AI tool. Crucially, we ensured that key people from multiple project partners could seamlessly share de-identified data files and responses from the villagers each week. Finally, we completed training for 20 front-line workers on the project implementation.

In February and March of 2022, our front-line workers collected behavior surveys (questionnaire) from 1698 participants who are local villagers in India. We successfully completed a pilot test of the diabetes-related AI messaging bot to verify functionality of the overall system. In the end, we successfully recruited 1049 participants to opt-in to receive the diabetes messages. To compare with the existing static message program in which all participants received the same sequence of messages, we randomly divided participants into two groups: 548

participants joined our AI-based message program (we call this group the AI group) and 501 joined the baseline static message program (the non-AI group).

In March of 2022, we began the message transmission for the first batch of users with those in the AI cohort receiving two messages and two questions weekly on Fridays and Tuesdays. Participants in the non-AI group received two messages on Thursdays and Mondays each week. We remark that all participants did not opt-in at the very start but gradually joined over a couple of weeks. The villagers in the AI group responded to questions asked — engagement levels were around 35% past week 8 of the study.

The study ended in November 2022, at which time the post-study questionnaire was sent to the participants. This is the same questionnaire used at the beginning of our study. Based on responses of participants, we are able to evaluate the effectiveness of the AI system by comparing the performance of participants between the AI and non-AI groups.

5.4 Post-Study Intervention Result

We analyze behavior changes in both AI and non-AI participant groups based on their responses to the same questionnaire before/after joining our study. We divide questions into different types that focus on *knowledge*, *physical activity*, and *dietary*. Responses to each question are converted into three scores: 1 (low), 2 (medium), and 3 (high). The final score of a participant in each category is averaged over all questions in that category. We compute the score difference between the pre- and post-studies for each participant. Positive score changes imply participants improve their behavior at the end of our study. Our analysis results are shown in Figures 17, 18, 19 where the x-axis represents the participants and the y-axis represents the score change.

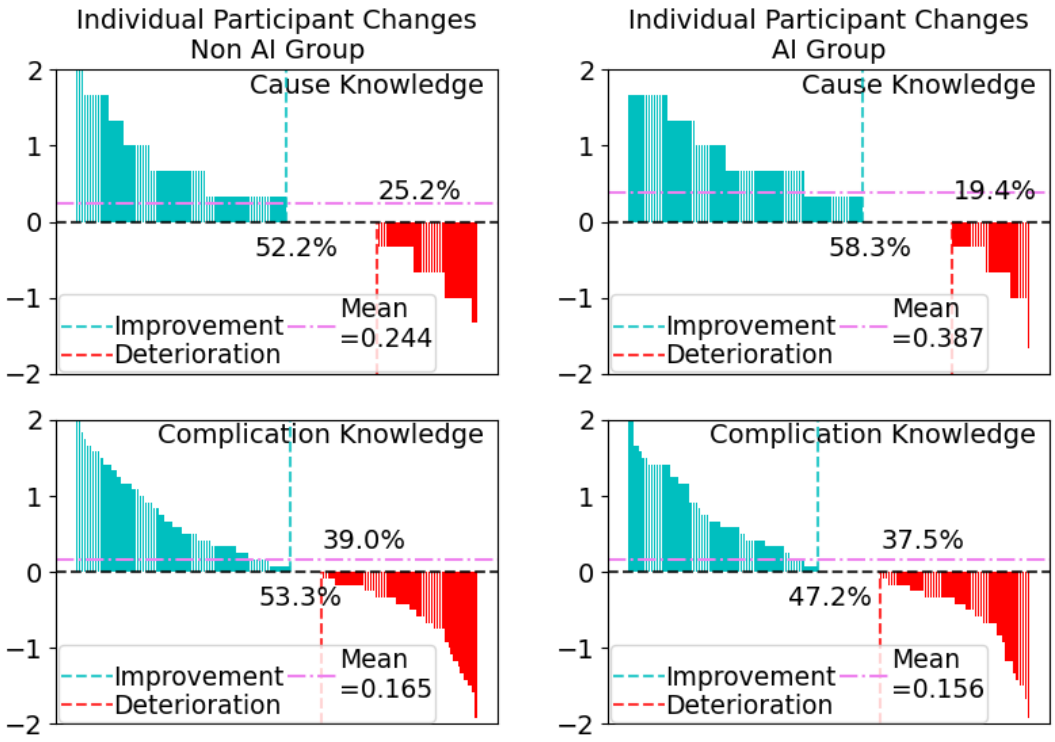


Figure 17. Knowledge category results

Knowledge Comparison. We plot results on *cause knowledge* and *complication knowledge* (of diabetes). Overall, AI group shows a substantial improvement in both cause and complication knowledge scores. In particular, AI group outperforms the non-AI group significantly in cause knowledge. In complication knowledge, non-AI group has a higher percentage of participants who have positive score changes but also a higher percentage of participants who have negative score changes — In the end, the AI-group obtains a higher mean score improvement compared to the non-AI group.

To compare our AI versus the non-AI intervention, we consider three statistics: (i) mean score change across each population; (ii) percentage of participants who improve their scores; and (iii) percentage of participants who decrease their scores. For example, in Figure 17 for the AI group, 58.3% of the participants have a positive score changes while 19.4% have a negative score changes in the diabetes-cause knowledge type. In addition, the mean score change is 0.387. For the AI group, we only consider participants who have response rates of

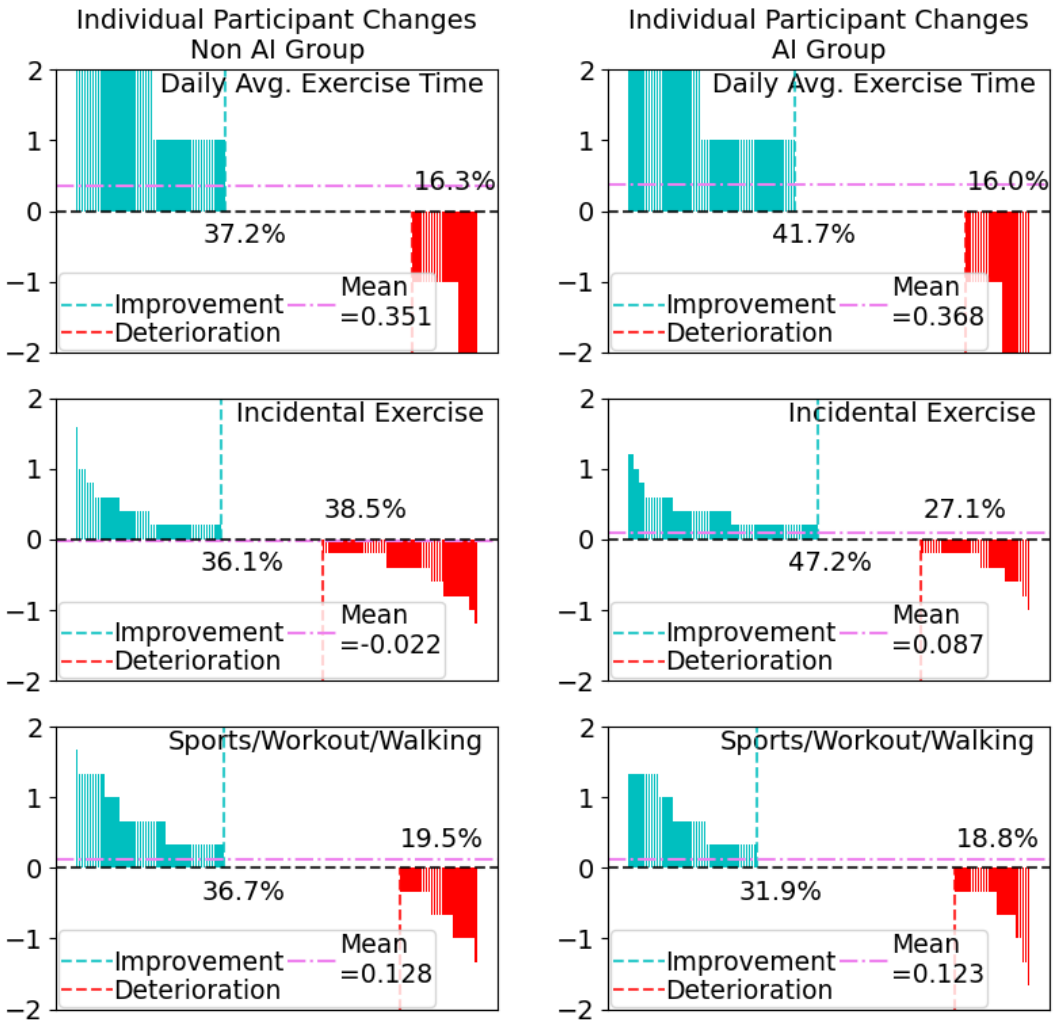


Figure 18. Physical activity category results

Physical Activity Comparison. We plot results on *daily average exercise time*, *incidental exercise* (this refers to exercises incurred throughout daily activities, such as choosing to walk for errands, walking around the house, and taking stairs instead of elevators, and *sport/workout/walking*). Overall, AI group shows a substantial improvement in both average exercise time and incidental exercise, which outperforms the non-AI group. In the sports/workout/walking type, non-AI group has a higher percentage of participants who have positive score changes; nevertheless, the mean score change is not much different.

at least 50%. Our rationale is that low-response rate participants do not engage in the study, and thus, their behavior will not be impacted by the AI messages.

Figure 19. Dietary category results

Dietary Comparison. We plot results on unhealthy food, fruit, and vegetable consumption. Overall, in the unhealthy food consumption category, the AI group shows a substantially more improvement compared to non-AI group — (25% improvement, 13.2% deterioration, 0.236 mean) versus (17.5% improvement, 13.8% deterioration, 0.073 mean). In the fruit/vegetable category, we observe somewhat negative changes in both non-AI and AI groups. That is, a higher percentage of participants had negative score changes compared to positive score changes. However, the mean score changes in both groups are very close to zero.

5.5 Human Behavior Modeling

We aim to build a predictive model characterizing how participants behave in response to our message intervention program. Such predictive model could be used in the future to create simulated data to refine our message selection policy.

Feature Extraction. We use data collected from our field study for this modeling task. We divide the questionnaire and the weekly messages/questions into 17 finer categories. The questionnaire responses are then used to compute scores in each of these categories for all participants, which are then used as features. Additionally, we include the message and question ID that the participant received, along with the category that they belong to. To provide the model with more context, we include the previous week's information for all these categories, along with the responses received from participants. Lastly, we also include a coarser categorization feature, indicating whether the question asked is regarding one of three types: knowledge, physical activity, or dietary.

Model Description. The prediction task, then, is to use the extracted features to predict participants' responses to the questions they receive. We cast this problem as a multi-classification problem. In each week, for each participant, we take each question and other features associated with that participant as an input to produce a prediction of the corresponding response of that participant (which is categorized into three levels 1 (low), 2 (medium), and 3 (high)).

We consider three different models as baselines for this task: (i) the classic logistic regression; (ii) a simple neural network (NN) with two fully-connected linear layers; and (iii) a Long Short Term Memory (LSTM) based model [48] (i.e., a

LSTM block followed by a simple linear layer). We remark that LSTM is commonly used in deep knowledge tracing [105].

Importantly, we observe participants' behavior changes vary across different types of diabetes-related activities. As a result, a single behavior model may perform poorly in predicting responses to various types of questions. Therefore, we aim to build predictive models that can differentiate behavior changes of three different types: food consumption, physical activities, and diabetes-related knowledge. More specifically, we propose the three models: (i) type-trifecta logistic regression — this model consists of three separate logistic regression components, each produces predictions for responses in one of the aforementioned types; (ii) type-trifecta simple neural network — this model consists of three separate neural net components, each is a simple 2-layer neural net; and (iii) type-trifecta LSTM — this model has a shared LSTM block followed by three separate blocks of linear layers.

Accuracy Evaluation. All of our behavior prediction models were trained on dual Intel E5-2690v4 processors. All experiments were trained in PyTorch using cross entropy loss and the Adam optimizer. For our experiments, we collect all results over 30 random seeds (resulting in different model initializations and test/train splits) and report the mean along with the standard deviation. We remark that there are a lot of missing responses in our training data set (response rates of participants is less than 40%). Therefore, in our experiments, we examine two options: one is to simply encode missing responses as "-1" and the another is to replace missing responses by our model predictions. In addition, we try adding noise (i.e., zero mean Gaussian noise) to participants' responses. The purpose is to examine if this noise helps in improving the robustness of our models or not. Our

Model	Acc. Train	Acc. Test
Simple NN	0.533 ± 0.016	0.525 ± 0.022
LSTM	0.541 ± 0.022	0.533 ± 0.030
Logistic Regression	0.524 ± 0.009	0.515 ± 0.018
Simple NN Type Trifecta	0.59 ± 0.005	0.58 ± 0.019
LSTM Type Trifecta	0.583 ± 0.006	0.575 ± 0.019
Logistic Regression Type Trifecta	0.585 ± 0.005	0.578 ± 0.018

Table 1. Evaluation with no noise, no predicted feature insertion.

Model	Acc. Train	Acc. Test
Simple NN	0.536 ± 0.017	0.528 ± 0.022
LSTM	0.548 ± 0.023	0.542 ± 0.028
Logistic Regression	0.525 ± 0.009	0.516 ± 0.018
Simple NN Type Trifecta	0.59 ± 0.005	0.58 ± 0.019
LSTM Type Trifecta	0.584 ± 0.005	0.576 ± 0.019
Logistic Regression Type Trifecta	0.584 ± 0.005	0.578 ± 0.019

Table 2. Evaluation with noise, but no predicted feature insertion.

prediction accuracy results for all models are shown in Tables 1–3, corresponding to three settings: (i) no noise is added to participants' responses and missing responses are encoded as "-1" (Table 1); (ii) similar to (i) but Gaussian noise is added (Table 2); and Gaussian noise is added and missing responses are replaced with the model prediction (Table 3).

All three tables show that differentiating behavior changes according to three different categories of food consumption, physical activities, and knowledge significantly improves the prediction accuracy of our models compared to the baselines. For example, in Table 1, the type-trifecta simple NN model obtains

Model	Acc. Train	Acc. Test
Simple NN	0.531 ± 0.014	0.523 ± 0.019
LSTM	0.548 ± 0.023	0.535 ± 0.029
Logistic Regression	0.53 ± 0.010	0.522 ± 0.019
Simple NN Type Trifecta	0.592 ± 0.005	0.578 ± 0.019
LSTM Type Trifecta	0.579 ± 0.008	0.572 ± 0.022
Logistic Regression Type Trifecta	0.587 ± 0.006	0.577 ± 0.019

Table 3. Evaluation with noise and predicted feature insertion.

an averaged prediction accuracy of 59% and 58% on the training and test sets, respectively. This is significantly higher than the prediction accuracy of the single simple NN model (i.e., 53.3% and 52.5%). We also observe the similar performance enhancement trend for the logistic regression and the LSTM-based models. In addition, interestingly, unlike in knowledge tracing [105] where LSTM-based models are shown to be superior in predicting the knowledge of students, our results show that the type-trifecta simple NN model performs the best. This phenomenon perhaps comes from the limited data availability (with missing responses) in our domain that potentially deteriorates the performance of complex models like LSTM. Lastly, we do not observe a substantial changes in prediction accuracy of all models when we introduce Gaussian noise to responses or insert model predictions to replace missing responses.

5.6 Deployment Results and Learned Lessons

In summary, we developed an RL-based personalized messaging system for diabetes intervention tailored to people living in rural areas where access to healthcare, social support, and education are limited. We ran an extensive field study that involves more than 1000 local villagers participating in our

messaging program. Our post-study analysis results show the significant benefit of our approach (compared to the existing system) in the participants' diabetes-related knowledge, physical activities, and high-fat food avoidance. Furthermore, our behavior models which leverage characteristics of participants' responses outperform baselines including the single LSTM model that is commonly used in knowledge tracing.

We would like to highlight some important lessons learned during this work. First, collaborations among different partners with different areas of expertise including NGOs, health domain experts, and academics are the key to the success of social impact projects. Second, real-world domains exhibit various challenges that we may not be able to anticipate when building our AI models. Continuing to improve our models and solutions with the adaptation to rising challenges is essential to the long-term impact of the project. For example, our current RL model does not directly account for missing responses from participants. Furthermore, answers about behavior are self-reported and could be misleading. We plan to address these limitations in future work. Third, available real-world data in this domain is extremely limited. Thus, simple models may work better than complex deep learning models.

Non-communicable diseases are a growing problem in low and middle income countries [77]. These include diseases such as hypertension and diabetes. Lifestyle, especially diet and exercise, can contribute to such diseases, though in some patients the cause is genetic. While effective awareness efforts and good healthcare facilities have led to reduction and management of such lifestyle diseases in developed countries, these diseases afflict many people in the developing countries. According to WHO, such diseases kill 41 million people each year, with

as much of 85% of premature deaths from lifestyle diseases occurring in developing countries [93]. But, in developing countries with weak healthcare infrastructure and limited access to health education, such lifestyle diseases are often under diagnosed and proper management regimes are not followed, making management of such diseases very challenging.

One effective approach to this problem is to prevent such diseases before onset, by spreading awareness about them. In many cases these diseases can be prevented by following a healthy lifestyle, such as regular physical activity, eating healthy food, and avoiding consumption of harmful products such as alcohol and tobacco. We posit that AI based approaches can be effective at spreading awareness and education to underserved populations.

There exists prior work that specifically uses such an AI based approach [61]. They designed a reinforcement learning (RL) based system to send targeted messages spreading awareness about diabetes to participants in a pilot study conducted in India. While the study was successful in nudging people to follow a healthier lifestyle, a number of real world challenges were abstracted away in the model design. Briefly, in this prior work, the RL action is to choose and send out messages to participants in the pilot study on a weekly basis. The messages are determined based on current behavior of the participant, which is stored as part of the state of the RL system. Their system collected information about changes in behavior of participants through a question/answer mechanism, that was used to update the state of the RL system. Participant feedback allows for more targeted messaging. However, this prior work did not tackle pertinent issues: (1) participants may not respond to all or some questions about their behavior, thus, the knowledge about the behavior of any participant may be incomplete at

the current time step; technically, this is the issue of incomplete state observation and (2) whether the messaging was successful or not was determined by a post-study questionnaire, thus, the reward for this RL task was received at the very end; technically, this is the issue of sparse rewards. Jointly, these two issues present a significant challenge that needs to be overcome to make the messaging intervention system more effective on ground.

Our work aims to address these challenges leading to more effective awareness campaigns for lifestyle diseases. In particular, we propose two novel ideas: (i) predict the unobserved state (or unobserved part of state) by building a state predictor that takes as input the action from previous time step and the observed part as well as the predicted part of the state from the previous time step; and (ii) redistribute the reward received at the end of the episode to intermediate states (formed by observed and predicted state) following a recent scalable approach [114] — it is known that such redistribution helps better learning in sparse reward problems. Note that this recent approach works only with fully observed states, thus, our state prediction is a novel design that enables the application of this approach to more general problems with incompletely observed states. Using these two ideas, we learn an effective messaging policy by utilizing a soft actor critic (SAC) architecture to learn from the predicted states and redistributed rewards. Additionally, we provide a mathematical guarantee that bounds the deviation of the learned long term reward in our case from the optimal long term return in the fully observed state scenario, as a function of the state prediction error.

Finally, we conduct extensive experiments on both a study using a data-driven simulator of the diabetes messaging system and standard Mujoco

benchmarks. In particular, we use available data from the prior pilot study to build a simulator of human responses, which serves as an environment for the messaging RL task. We utilize an LSTM as the state predictive model. Our results clearly show the advantage of addressing the incomplete state observation and sparse reward issues.

5.7 Next Steps: Improving the RL System

Motivated by the success of this real world deployment, we then resolved to find ways to improve upon the reinforcement learning system for future use.

Real-world Challenges of Sparse Rewards and Incomplete State Observations. To recap, we conducted a field study in two villages in India in which more than 1000 local villagers received messages and responded to questions sent out by our RL system over a period of six months. While our pilot study shows some promising results, we identified key real-world challenges present in the domain that our initial RL algorithm failed to address.

The first challenge is regarding incomplete state observations. In a real-world setting, only certain components of participant states are observable in every week. Recall that participant states are updated based on their responses to the questions sent to them in that week. However, to prioritize participants' convenience and comfort, questions are typically crafted to include simple content and focus solely on one or a few components of their state. An example of a question is "In the last week, how often would you have done any form of exercise?", then if the participant responds, their answer only provides information to update the physical activity in his state. Furthermore, our field study shows that the response rate of participants is only roughly 40%.

The second challenge is regarding sparse rewards. For our initial deployment, rewards of messages are measured based on how much these messages influence participants' behavior. However, a thorough assessment of these behavioral changes can only take place upon completion of the study, when a post-study questionnaire is conducted. In other words, we can only accurately compute the episodic reward associated with the messages at the end of the study.

Our work, therefore, focuses on developing new practical RL algorithms that addresses these two challenges.

5.8 Background

We consider an MDP which can be represented as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \rho_0)$ in which $\mathcal{S}$ is a set of states and $\mathcal{A}$ is a set of actions, and γ is the discount factor. For example, in diabetes prevention, these states represent participants' health-related states which comprise of knowledge levels, physical activities, and food intakes, etc while actions correspond to the health-related messages. The initial state distribution is denoted by ρ_0. The dynamics of the environment is denoted by p in which $P(s' \mid s, a)$ is the probability the agent will move to a new state s' given the agent takes an action a at state s. For every transition (s, a, s'), the agent receives an immediate reward $R(s, a)$. Given the reward function, the goal of reinforcement learning is to find an optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ for the agent that maximizes the expected sum of discounted rewards, formulated as follows:

$$\pi^* \in \operatorname*{argmax}_{\pi} J(R, \pi) = \operatorname*{argmax}_{\pi} \mathbb{E}_{\tau} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

where $\tau = (s_0, a_0, s_1, a_1, s_2, a_2 \cdots)$ is a trajectory with $s_0 \sim \rho_0(\cdot), a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim P(\cdot \mid s_t, a_t), \forall t$.

We now considers the episodic RL setting in which the agent can only obtain one reward feedback at the end of each trajectory, assuming all trajectories

terminate in finite steps. The episodic reward function $R_{ep}(\tau)$ is defined on the trajectory space. In practice, a common structural assumption is that there exists an underlying Markovian reward function $\hat{R}(s, a)$ based on which the episodic reward can be approximated as the following sum-form decomposition:

$$R_{ep}(\tau) = \sum_{t=0}^{T} \hat{R}(s_t, a_t)$$

Existing works on episodic RL aim at recovering this reward function $R(s_t, a_t)$ based on episodic reward feedback, assuming full observations of states s_t at every time step t. Our work, on the other hand, studies episodic RL in the context of incomplete state observation. That is, only certain parts of states are observable. In particular, at every step t, given the true state s_t, the observed parts of s_t is denoted by $o(s_t)$, which is sampled from a distribution $o(s_t) \sim q(\cdot \mid s_t)$ where the support $sup(q(\cdot \mid s_t)) = O(s_t)$ consists of all possible subsets of components of s_t that can be observed. For example, in the diabetes domain, the observation space consists of all possible subsets of elements (e.g., physical activity level, cause knowledge, complication knowledge, healthy food, unhealthy food, etc) that comprise the participants' state.

5.9 State Prediction Embedded with Randomized Return Decomposition

Our goal is to find an optimal policy that maximizes the expected trajectory return. We propose a new learning model, named SPER2D (**S**tate **P**rediction **E**mbedded with **R**andomized **R**eturn **D**ecomposition) that tackles the aforementioned challenges of episodic returns (aka. sparse rewards) and incomplete state observations. In the following, we first describe our new learning model. We then present our theoretical analysis of the proposed model. Finally, we introduce our practical SPER2D algorithm accordingly.

5.9.1 SPER2D Model Description. Overall, our model is an integration of three key components: (i) state prediction — providing a prediction $\bar{s}_t = f_w([o(s_{t-1}), \neg o(\bar{s}_{t-1})], a_{t-1}))$ at time step t given the observable parts $o(s_{t-1})$, the prediction of remaining unobservable parts $\neg o(\bar{s}_{t-1})$ of the true state s_{t-1}, and action a_{t-1} at previous step $t-1$; (ii) reward redistribution — redistributing the episode reward $R_{ep}(\tau)$ of a trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots, s_T, a_T)$ over its (predicted) state-action pairs $\hat{R}_\theta(\bar{s}_t, a_t)$; and (iii) actor-critic policy learning — generating a policy $\pi_\psi(\cdot \mid \bar{s}_t)$. Here, (w, θ, ψ) are the parameters of the three component's neural networks. In the following, we explain these components in details.

State Prediction. At every time step t, given the current underlying state s_t, the agent only has an observation of some components of s_t, which we denoted by $o(s_t)$ while other components of the state are hidden. Our state-prediction model, f_w, produces a state prediction $\bar{s}_t = f_w([o(s_{t-1}), \neg o(\bar{s}_{t-1})], a_{t-1}))$ at every time step t. Recall that $\neg o(\bar{s}_{t-1})$ represents the prediction of the hidden parts of s_{t-1}. Given the current fixed policy π_ψ, the model parameter w is trained based the following loss function:

$$\mathcal{L}_{SP} = \mathbb{E}\Big(\sum_{t=0}^{T} \big[o(\bar{s}_t) - o(s_t) \big]^2 \Big)$$

where $s_o \sim \rho_0$, $s_t \sim P(\cdot \mid s_{t-1}, a_{t-1})$, $\bar{s}_t \sim (f_w \circ q)(\cdot \mid s_{t-1}, a_{t-1})$, $a_t \sim \pi_\psi(\cdot \mid \bar{s}_t)$. Here, the loss is computed based on the difference between the observed parts $o(s_t)$ and the prediction of the observed parts $o(s_t)$ of the true state s_t at every time step t. For the sake of representation, we use the notion $\bar{s}_t \sim (f_w \circ q)(\cdot \mid s_{t-1}, a_{t-1})$ to represent $o(s_{t-1}) \sim q(\cdot \mid s_{t-1})$, $\bar{s}_t = f_w([o(s_{t-1}), \neg o(\bar{s}_{t-1})], a_{t-1}))$ The overview of this state prediction is illustrated in Figure 20.

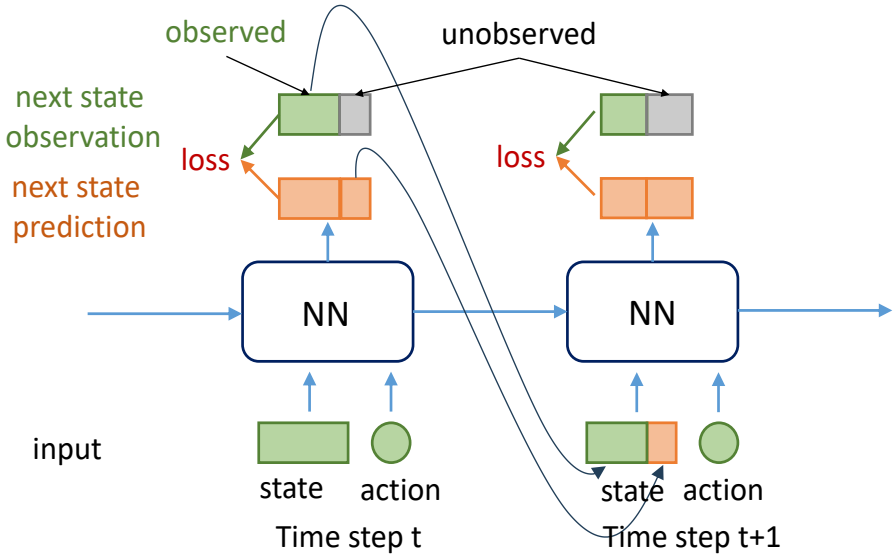


Figure 20. State prediction component

Reward Redistribution. We extend the idea of randomized return decomposition introduced in [114], to accommodate the incomplete state observation using the outcomes of our state prediction component. In particular, given a fixed state prediction f_w and a policy π_ψ, we attempt to find a Markovian reward function $\hat{R}_\theta(s, a)$ which minimizes the following loss function:

$$\mathcal{L}^w_{RRD}(\theta) = \mathbb{E}_{\bar{\tau}}\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\left[\left(R_{ep}(\tau) - \frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t)\right)^2\right]$$

where the trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots)$ and the state-prediction trajectory $\bar{\tau} = (s_0, \bar{s}_0, a_0, s_1, \bar{s}_1, a_1, \cdots)$ with $s_t \sim P(\cdot \mid s_{t-1}, a_{t-1})$, $\bar{s}_t \sim (f_w \circ q)(\cdot \mid s_{t-1}, a_{t-1})$, $a_t \sim \pi_\psi(\cdot \mid \bar{s}_t)$. Note that the reward function is defined w.r.t the predicted states s_t for every step t since the agent has access to observations of states only. Here, we use a Monte-Carlo estimator which only samples a subset of time steps $t \in \mathcal{I}$ in approximating the episodic return where $\mathcal{I}$ is a subset of indices of a size $K < T$. The distribution $\rho_T(\cdot)$ is defined as follows:

$$\rho_T(\cdot) = \text{Uniform}(\{\mathcal{I} \subseteq \mathbb{Z}_T : |\mathcal{I}| = K\})$$

As shown in [114], this approach improves the scalability of least-squares-based reward redistribution methods.

Policy Learning. Finally, we aim at optimizing the policy π_ψ based on outcomes of both the state prediction f_w and reward redistribution $\hat{R}_\theta$. In particular,

$$\max_{\psi} J^w(R_\theta, \pi_\psi) = \max_{\psi} \mathbb{E}_{\bar{\tau}} \sum_t \gamma^t \hat{R}_\theta(\bar{s}_t, a_t)$$

where the trajectory $\bar{\tau} = (s_0, \bar{s}_0, a_0, s_1, \bar{s}_1, a_1, \cdots)$ with $s_0 \sim \rho_0(\cdot), \bar{s}_t \sim (f_w \circ q)(\cdot \mid s_{t-1}, a_{t-1}), a_t \sim \pi_\psi(\cdot \mid \bar{s}_t), s_{t+1} \sim P(\cdot \mid s_t, a_t)$ for all time steps $t = 0, 1, \cdots$

5.9.2 Theoretical Analysis.

As described previously, our state prediction component plays a central role in influencing outcomes of both the reward redistribution and policy learning. Therefore, in this subsection, we study theoretical results on the impact of the accuracy of our state prediction component on the optimality of the learning outcomes of the later two components.

Let us first denote by $\epsilon > 0$ the prediction error of our state-prediction model. That is, for all state s, we have $||\bar{s} - s|| \leq \epsilon$ for all state prediction $\bar{s} \in$ *support*$((f_w \circ q)(\cdot \mid s))$. We make the following assumptions:

1. The reward function: $|\hat{R}_\theta(s, a) - \hat{R}_\theta(\bar{s}, a)| \leq C_r \cdot \epsilon$ for all (s, s') such that $||s - \bar{s}|| \leq \epsilon$ and all actions $a \in \mathcal{A}$. Here C_r is a positive constant.

2. The policy function: $1 - C_p^l \cdot \epsilon \leq \frac{\pi_\psi(a|\bar{s})}{\pi_\psi(a|s)} \leq 1 + C_p^u \cdot \epsilon$ for all (s, s') such that $||s - \bar{s}|| \leq \epsilon$ and all actions $a \in \mathcal{A}$. Here (C_p^l, C_p^u) are positive constants.

Optimality bound on reward function. We first examine how the state-prediction error will impact the optimality of the learning outcome of the reward function. To do so, we consider the *perfect* scenario when the agent has full

observations of states at every time step. In this case, the loss function of the reward redistribution can be represented as follows:

$$\mathcal{L}_{RRD}(\theta) = \mathbb{E}_{\tau}\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big[\Big(R_{ep}(\tau) - \frac{T}{|\mathcal{I}|}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t,a_t)\Big)^2\Big]$$

where the trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots)$. Obviously, this is an ideal situation with no state prediction error.

Theorem 2. *Let's denote by:* $\bar{\theta}^* \in \text{argmin}_\theta\, \mathcal{L}^w_{RRD}(\theta)$ *the reward learning outcome with incomplete state-observations and* $\theta^* \in \text{argmin}_\pi\, \mathcal{L}_{RRD}(\theta)$ *the learning outcome of the perfect observation scenario. We bound* $\mathcal{L}_{RRD}(\bar{\theta}^*)$ *in comparison with* $\mathcal{L}_{RRD}(\theta^*)$ *as follows:*[1]

$$\begin{aligned}
&\mathcal{L}_{RRD}(\theta^*) \le \mathcal{L}_{RRD}(\bar{\theta}^*) \le \\
&A_2(\epsilon)\Big[A_1(\epsilon)\mathcal{L}_{RRD}(\theta^*) + B_1(\epsilon)\sqrt{\mathcal{L}_{RRD}(\theta^*)} + C_1(\epsilon)\Big] \\
&+ B_2(\epsilon)\sqrt{A_1(\epsilon)\mathcal{L}_{RRD}(\theta^*) + B_1(\epsilon)\sqrt{\mathcal{L}_{RRD}(\theta^*)} + C_1(\epsilon)} \\
&+ C_2(\epsilon)
\end{aligned}$$

where the coefficients $A_1(\epsilon) = (1 + C^u_p \cdot \epsilon)^T$, $B_1(\epsilon) = (2 \cdot T \cdot C_r \cdot \epsilon)\sqrt{(1 + C^u_p \cdot \epsilon)^T}$, *and* $C_1(\epsilon) = \frac{T^2}{K}(C_r \cdot \epsilon)^2$. *In addition, we have the coefficients* $A_2(\epsilon) = \frac{1}{(1-C^l_p\cdot\epsilon)^T}$, $B_2(\epsilon) = (2 \cdot T \cdot C_r \cdot \epsilon)\sqrt{\frac{1}{(1-C^l_p\cdot\epsilon)^T}}$, *and* $C_2(\epsilon) = \frac{T^2}{K}(C_r \cdot \epsilon)^2$.

Intuitively, Theorem 2 presents an optimality bound on the learning outcome of our reward function in the presence of the state-prediction error ϵ in comparison with the actual optimal learning outcome when there is no state prediction error.

Optimality bound on policy function. In general, the policy learning outcome depends on the performance of both the state-prediction and reward

[1]All proofs are in the appendix.

redistribution components. In the following, we analyze the optimality of our policy learning result in comparison with the actual optimal policy learning outcome when there is no error in the state prediction and reward redistribution. Based on Theorem 2, we consider a general situation that $|\hat{R}_{\theta^*}(s,a) - M(\epsilon)\hat{R}_{\bar{\theta}^*}(s,a)| \leq \Delta(\epsilon)$ where $M(\epsilon) \rightarrow 1$ and $\Delta(\epsilon) \rightarrow 0$ given ϵ is the prediction error of our state-prediction model.

Theorem 3. *Let's denote by* $\pi^* \in \text{argmax}_\pi J(\hat{R}_{\theta^*}, \pi)$ *the optimal policy when states are fully observed and* $\bar{\pi}^* \in \text{argmax}_\pi J^w(\hat{R}_{\bar{\theta}^*}, \pi)$ *the result of our policy learning when the state-prediction error is* ϵ *and the corresponding reward redistribution outcome is* $\hat{R}_{\bar{\theta}^*}$. *We then obtain:*

$$
\begin{aligned}
&J(\hat{R}_{\theta^*}, \pi^*) \geq J(\hat{R}_{\theta^*}, \bar{\pi}^*) \geq \\
&\frac{1}{(1 + C_p^u \cdot \epsilon)^T \cdot M(\epsilon)} \Bigg[M(\epsilon) \cdot (1 - C_p^l \cdot \epsilon)^T \cdot J(\hat{R}_{\theta^*}, \pi^*) \\
&\quad - M(\epsilon) \cdot (1 - C_p^l \cdot \epsilon)^T \frac{\Delta(\epsilon) + C_r \cdot \epsilon}{M(\epsilon) \cdot (1 - \gamma)} - \frac{C_r \cdot \epsilon}{1 - \gamma} \cdot \frac{\Delta(\epsilon)}{1 - \gamma} \Bigg]
\end{aligned}
$$

5.9.3 Practical SPER2D Algorithm. An overview of our practical SPER2D algorithm is illustrated in Figure 21 and we give psuedocode for it in Algorithm 3. In particular, optimizing the state-prediction loss $\mathcal{L}_{SP}$ and reward redistribution loss $\mathcal{L}^w_{RRD}$ can be carried out by using mini-batch gradient descent. Note that trajectories $\mathcal{D}^r$ used to train the reward redistribution component are collected with states partially generated by our state-prediction component. Finally, in train our policy learning component, we implement the soft actor-critic method, that utilizes trajectories $\mathcal{D}^\pi$ constructed based on the outcomes of both the state prediction and reward redistribution components.

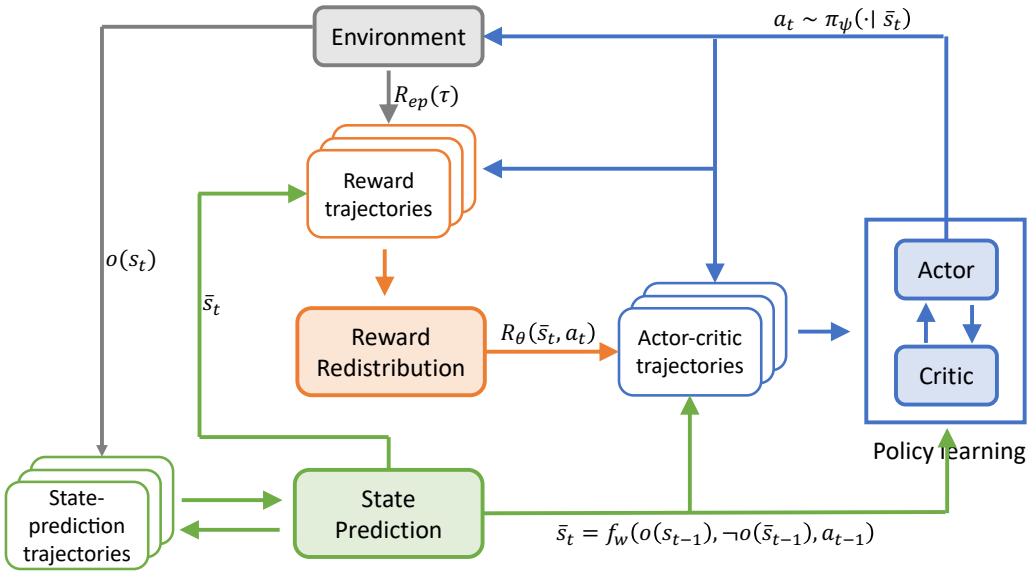


Figure 21. Overview of our SPER2D algorithm

5.10 Experiments

5.10.1 Algorithm Implementation and Baselines. We utilize a state predictive model consisting of an LSTM, followed by an optional 2-headed attention layer, with a final linear layer to produce the output. We model the reward redistribution component as well as the actor and critic of the policy learning component using neural nets with three fully connected linear layers. Details of our experiment setup and hyperparameters of our models are given in the appendix.

In addition to our SPER2D method, we consider the following baselines for comparisons:

RRD. This algorithm is introduced in [114] which only focuses on the sparse reward challenge. The incomplete state observations are used as if they are true underlying states in both the reward redistribution and policy components. In

Algorithm 3: State Prediction Embedded with Randomized Return Decomposition (SPER2D)

Input: environment `Env`, initial state, reward, and policy parameters: (w, θ, ψ) and empty buffer $\mathcal{D}$;
for $l = 1, 2, \ldots$ **do**
  `Env`.reset();
  for $t = 0, 1, \cdots$ **do**
    Observe partial state $o(s_t) \sim q(\cdot \mid s_t)$;
    Predict $\bar{s}_t = f_w([o(s_{t-1}), \neg o(\bar{s}_{t-1})], a_{t-1}))$;
    Execute action $a_t \sim \pi_\psi(\cdot \mid \bar{s})$;
  Collect $\bar{\tau}^s = (o(s_0), \bar{s}_0, a_0, o(s_1), \bar{s}_1, a_1, \cdots)$;
  Add trajectory to buffer: $\mathcal{D}^s = \mathcal{D}^s \cup \bar{\tau}^s$;
  for *i = 1, 2, ...* **do**
    Sample M_s trajectories $\{\tau_j\}$ from $\mathcal{D}^s$;
    Update state-prediction model using the loss:

$$\mathcal{L}_{SP} = \frac{1}{M_s} \sum\nolimits_j \Big(\sum\nolimits_{t=0}^{T} \big[o(\bar{s}_{j,t}) - o(s_{j,t}) \big]^2 \Big)$$

  Collect trajectory $\bar{\tau}^r = (\bar{s}_0, a_0, \bar{s}_1, a_1, \cdots)$ and episodic feedback $R_{ep}(\bar{\tau}^r)$;
  Add to buffer: $\mathcal{D}^r = \mathcal{D}^r \cup \{\bar{\tau}^r, R_{ep}(\bar{\tau}^r)\}$;
  for $i = 1, 2, \ldots$ **do**
    Sample M_r trajectories $\{\tau_j\}$ from $\mathcal{D}^r$;
    Sample subsequences $\{\mathcal{I}_j\}$ from these trajectories;
    Update reward model $\hat{R}_\theta$ using the loss:

$$\mathcal{L}^w_{RRD}(\theta) = \frac{1}{M_r} \sum_j \Big(R_{ep}(\bar{\tau}_j) - \frac{T}{|\mathcal{I}|} \sum_{t \in \mathcal{I}} \hat{R}_\theta(\bar{s}_t, a_t) \Big)^2$$

  Collect $\bar{\tau}^\pi = (\bar{s}_0, a_0, \hat{R}_\theta(\bar{s}_0, a_0), \bar{s}_1, a_1, \cdots)$;
  Add trajectory to buffer $\mathcal{D}^\pi = \mathcal{D}^\pi \cup \{\bar{\tau}^\pi\}$;
  Run soft-actor critic to update policy π_ψ using $\mathcal{D}^\pi$;

RRD, there is no state-prediction component to handle the challenge of incomplete observations.

Online Decision Transformer (ODT). We utilize an online Decision Transformer approach [22, 174] as the second baseline. Here, a transformer model designed to directly output decisions is trained offline, and then fine-tuned

online. As in the RRD baseline, we maintain a state belief at each timestep based on the most recent observation of each element. We pre-train the transformer model on the Hopper, HalfCheetah, and Walker2d tasks, before freezing the transformer block and using it in our online learning settings with sparse rewards and incomplete state observations.

5.10.2 Experimental Domains. We consider two domains in our evaluation: (i) the simulated diabetes messaging domain, built based on real-world data collected in [61]; and (ii) the OpenAI's Gym environments, powered by the MuJoCo physics simulator. This domain was chosen as they are readily available, widely used in RL research, and can be easily adapted to share the problems of our diabetes setting (namely, sparse rewards and incomplete state observability).

Diabetes Intervention. We built a simulation of the diabetes messaging domain based on real-world data provided in [61]. Determining the (partial) state of each participant is performed by a predictive behavior model of participants trained on the real-world data. This allows us to simulate the entire messaging system. We keep the model conditioned upon the real world data through several mechanisms. First, we ensure that said real world data is included in the trajectory buffer. Next, we set the starting state for each trajectory to either a starting state drawn from the real world participant pool, or to a randomized state where each element is selected from a real participant in the pool. This should help ensure that our trained SPER2D model will remain effective when applied to new real-world participants, despite the training process relying on simulated process. Lastly, the episodic reward is estimated based on responses of participants to the pre-study and post-study questionnaires.

Since this is a relatively small domain with a short horizon in which each trajectory consists of only 25 time steps (aka. 25 weeks of the field study in [61]), the decision transformer model is not appropriate due to its requirement of a large amount of training data. Therefore, we restrict our experiments on this domain to our SPER2D-LSTM algorithm and the RRD baseline. Both of these algorithms require no pre-training (unlike the decision transformer based methods), and are designed to be sample efficient during online learning.

OpenAI Gym - Mujoco. We make a couple of modifications to standard Gym [16] Mujoco [135] environments so that they are more similar to our setting. First, we delay all episode rewards to the end (as in [114]) to mimic the sparse reward function of the diabetes messaging domain. As these tasks are long horizon (1000 timesteps without early termination), setting the reward of all non terminal states to 0 while giving the entire episodic reward at the final step significantly increases the difficulty of the problem. Secondly, we introduce partial observability by concealing some elements of the state at each time step. To further the similarity between this and the diabetes domain, we identify the component parts of each MuJoCo body (e.g. the torso and individual legs of the ant) and randomly conceal 1 to 3 of these parts. This effectively gives us 3 new domains of varying difficulty for each MuJoCo setting, allowing us a clear picture of both the benefits and limitations of our method.

5.10.3 Results and Analysis.

Results on Diabetes Intervention. In Figure 22 and Table 4, we display the results of our experiments in the diabetes messaging domain. As explained previously, since this is a relatively small domain, the decision transformer model

Category	RRD	**SPER2D**
Overall	70.83±3.86	**77.68±3.73**
Junk Food Avoidance	45.15±3.82	**53.92±4.23**
Alcohol Avoidance	46.77±3.32	**53.20±3.71**
Tobacco Avoidance	46.83±3.44	**52.98±3.84**
Healthy Food Intake	45.42±4.11	**49.78±4.63**

Table 4. Percentage of participants (with standard error) showing behavior improvement at the end of intervention, selected categories. Reported over 40 random seeds.

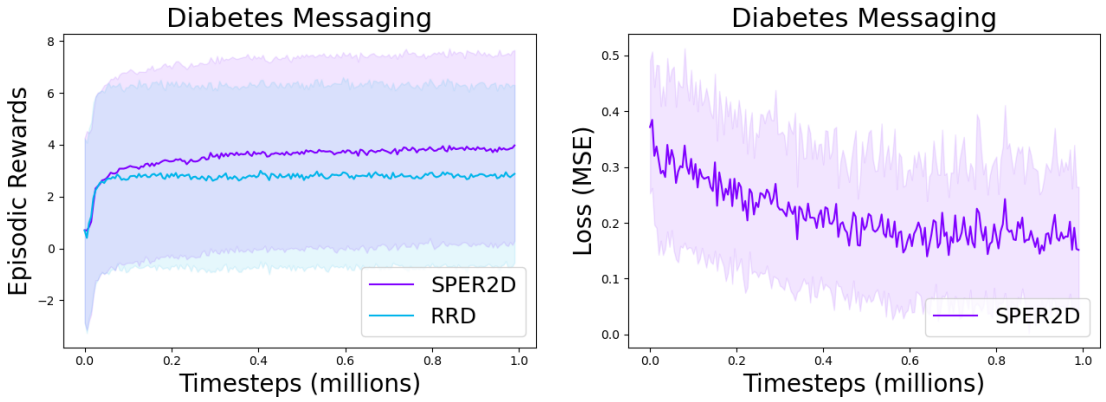


Figure 22. Diabetes simulation results
Diabetes domain episodic returns (mean returns of 50 on-policy trajectories) and state predictor training curve. Mean and standard deviation over 40 random seeds.

is not appropriate due to its large dataset pre-training requirements. Therefore, we restrict our experiments on this domain to our SPER2D algorithm and the RRD baseline, given that these algorithms require no pre-training and are designed to be sample efficient during online learning.

Compared to the baseline (RRD), our method produces consistently better policies characterized by higher episodic rewards (Figure 22), corresponding a noticeably higher percentage of (simulated) participants who have their health-related behavior improved at the end of the intervention process (Table 4). These gains are largely concentrated in the selected categories, all of which are related to food and drug consumption. Given the amount of unobservable data at each

Task	RRD	ONLINE DT	**SPER2D**
Ant (1)	1435±399	944±30	**2270±438**
Ant (2)	890±32	725±86	**1308±126**
HalfCheetah (1)	4052±154	3348±59	**7471±907**
HalfCheetah (2)	2200±288	1730±240	**4415±191**

Table 5. Mean and standard error for final episodic returns of 20 on-policy trajectories, over 10 random seeds. MuJoCo tasks.

timestep, these results show that building a predictive model for the environmental state is essential for learning an effective messaging policy that can improve substantially people's lifestyle behavior. Furthermore, the environment itself is simple enough that a relatively small recurrent model (an LSTM with attention) with no offline pre-training proves effective. Utilizing our method, future real-world deployment of health intervention messaging apps may achieve similar gains, enabling participants to improve their healthy habits.

Results on Mujoco. Our evaluation results in the (modified) MuJoCo environment are given in Table 5. In addition, Figure 23 shows the performance of all evaluated algorithms during the training process. We evaluate the performance of compared algorithms in two tasks: Ant, and HalfCheetah. These domains were selected based on their complexity: HalfCheetah uses a two dimensional robot body with a total of nine links connected by eight joints, while Ant is a much more complicated three dimensional body with four limbs and a torso. We display results separately for each task, where tasks have varying amounts of information that is concealed: 1 to 2 *component parts*, such as a rotor, of the state are unobservable at every timestep. For example, Ant (1) refers to the scenario in which one component part is hidden. Each component part includes multiple entries in the observation

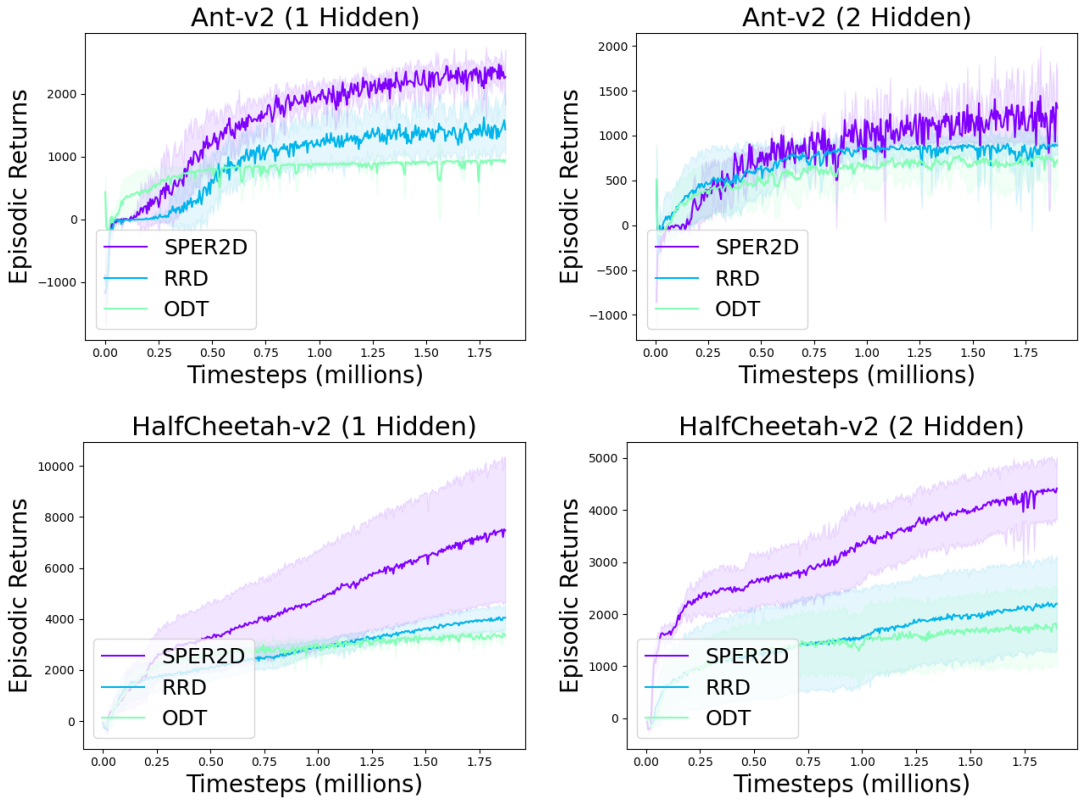


Figure 23. MuJoCo Gym environment results
MuJoCo Gym environment episodic returns (mean returns of 20 on-policy trajectories) and state predictor training curve. Mean and standard deviation over 10 seeds.

vector (e.g. both the angle and angular velocity of a limb), meaning a significant amount of information is hidden at each timestep.

Overall, our algorithm outperforms the baselines in all of these tasks, reflecting the advantages of modeling missing information. This difference in performance is most notable in the HalfCheetah domain when two components are hidden, where our method receives double the average episodic rewards compared to both the RRD and the online decision transformer baselines. Across the board, even the powerful online decision transformer is unable to implicitly account for the missing information, meaning that there is much advantage to be found

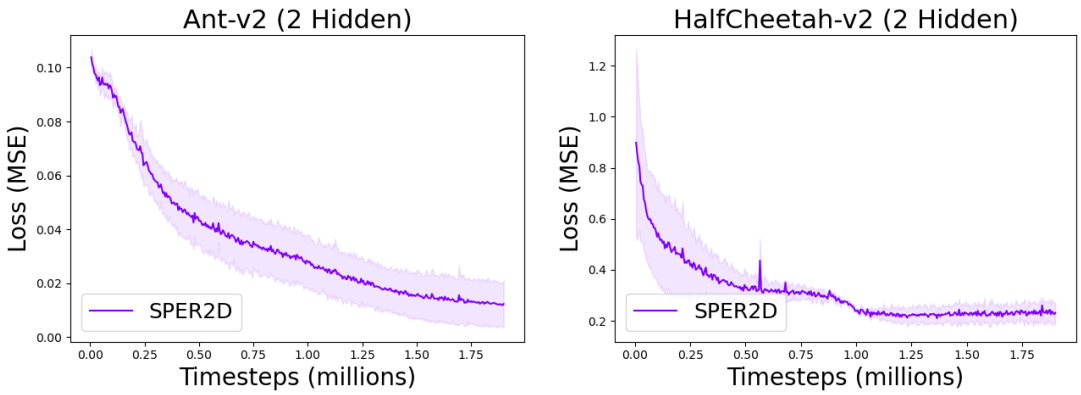


Figure 24. MuJoCo Gym training curves
MuJoCo training curves for the state predictive component. Mean and standard deviation over 10 seeds.

in explicitly modeling it, as our method does. Furthermore, the online decision transformer requires pre-training on a fully observable version of the task, while our method functions without such pre-training available.

To further investigate the performance of our fully online algorithm, SPER2D, we plot the loss of our state-prediction component in Figure 24. As we explain in previous subsections, the state-prediction component plays a central role in influencing the performance of both the reward redistribution and policy learning component. We observe that the state predictive model converges quickly and consistently in both the selected settings. This indicates that a model simple enough to be trained entirely online (LSTM with attention) is capable of handling the challenges of our real-world diabetes messaging setting.

5.11 Summary

In this study, we introduce a novel RL algorithm called SPER2D, designed to address the challenges posed by sparse rewards and incomplete state observations. Our research is driven by the practical challenges of conducting message-based diabetes interventions, where participant feedback is restricted,

and the success of the messaging is only evident upon completion of the study. Our SPER2D algorithm is a novel integration of three interdependent learning components: state prediction, reward redistribution, and actor-critic policy learning. We provide a mathematical guarantee that bounds the difference between the learned long-term reward in our scenario and the optimal long-term return in a fully observed state situation, based on the state prediction error. Finally, we empirically evaluate the performance of our SPER2D algorithm on both the data-driven simulator of the diabetes messaging system and the RL Mujoco benchmarks. Experimental results demonstrate the effectiveness of our algorithm in comparison with the state-of-the-art baselines.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Artificial intelligence necessitates careful consideration of the data being used. Said data can be simply incomplete, inaccurate, or altered by a malicious actor, and disaster can strike if it is assumed to be trustworthy when it is not. Data-based decision making models can be powerful tools, but it is important to keep in mind the vulnerabilities that reliance on real-world data introduces. This work investigated such data vulnerabilities from multiple perspectives and sought to answer three main research questions.

Towards that goal, Chapter I covered background knowledge required to understand the rest of the work. This included an overview of data-based decision making approaches, a primer on adversarial learning and the type of attacks that have been considered, and an exploration of game theoretic concepts relevant to security games. Then, we moved on to answer those three primary research questions.

Is it feasible to attack data-based decision making models directly? In Chapter II, we investigated the possibility of formulating and solving a direct poisoning attack against a data-based decision maker, using the metagradient technique. We considered an approach of solving an attack against a simplified model, then transferring this attack to the more complex, production quality, models. Furthermore, we explored direct attacks against both a two-stage model, as well as the more novel and sophisticated decision focused learner. We found some utility in these attacks, and also showed that transferring attacks between two-stage and decision focused models can be effective. However, we also found this direct

approach to poisoning attacks to require a prohibitive amount of computation, and that the degree of numerical stability required for these second order gradients may be lacking in traditional deep learning frameworks.

Given a poisoned data set, and some knowledge of the limitations of the attacker's poisoning capability, how can a wary defender successfully address this deception? In Chapter III, we studied a game theoretic threat model. Under this model, a defender forms a belief regarding attacker population behavior using historical attack data, and one of the attackers has poisoned this historical data for its own purposes. We find that a defender having sufficient knowledge can successfully "see through" the poisoned data and define a range in which the underlying "true" behavior falls. Then, said defender can utilize a binary search algorithm we developed to formulate an effective defense.

How can an online reinforcement learning model successfully utilize extremely sparse and incomplete real world data? In Chapter IV, we deployed a reinforcement learning model into the real world for the purposes of diabetes outreach and education. While this deployment did show promise, we also identified key challenges inherent to this domain that went unaddressed initially. Namely, the sparsity of responses from our participants, and the inability of our system to obtain meaningful per-step rewards. To address those challenges, we formulated a new reinforcement learning approach, building on the valuable real world data collected during the deployment. Firstly, we used an existing reward redistribution method (RRD) to effectively parcel out the episodic rewards (given by differences between the end survey and the initial survey) to the weekly decisions made by the RL model. Secondly, we used a predictive approach to fill in the gaps, accounting for the sparse responses by participants. These improvements

showed demonstrably better rewards, both in a simulated version of the diabetes messaging domain, and in standard MuJoCo Gym environments.

6.2 Future Work

This work opens up avenues that hopefully will be explored by future research projects, which can be broken down into 3 main directions. First, more thorough investigation of direct, end to end, attacks against data-based decision making models is warranted. Secondly, game theoretic formulations of data-based decision makers and potential adversaries have seen promising real world deployments, compelling research to address deception of different forms. Finally, the real world data gathered during deployment of our diabetes education RL model, and the improvements already made upon that system, should be put to good use.

6.2.1 Direct Attacks on Data-based Decision Makers.

While the direct approach we explored in Chapter II is likely not feasible against real world models, similar approaches could be. Adapting MetaPoison [53], which solves an attack against an ensemble of depth-limited learners, or Witch's Brew [38], which improves poisons by aligning the attacker's gradient with that of the defender, would be a promising place to start. Additionally, generative approaches such as the work in [84] could be effective, without requiring solving the bilevel problem of optimizing attacks via gradient descent. Furthermore, given effective attacks, it's natural to consider defenses as well. For example, if the training data for a data-based decision maker is known to contain a certain percentage of (imperceptibly) poisoned data points, how might this poison be overcome during training? What mathematical guarantees could be made about the resulting decisions?

6.2.2 Addressing Deception, Game Theoretically.

In Chapter III, we used a game theoretic approach to model both a data-based decision maker, as well as an attacker. This kind of formulation has shown useful in real world deployments such as PAWS [30] and PROTECT [121]. While we addressed one specific form of deception, others still exist. For example, deception might involve a setting in which multiple types of attackers, each with their own priorities, exist, and a deceptive attacker pretends to be a different type than it actually is. Addressing that kind of attack would require different methodology, and should be studied. Furthermore, our approach assumed that the adversary's power is known in advance (e.g. a security force has intel about a clever group of adversaries and their capabilities). If this assumption doesn't hold, can we use the data itself, in comparison with previous data, to estimate the power of the attack? What guarantees, if any, of defense could be offered in such a scenario?

6.2.3 Reinforcement Learning for Diabetes Intervention.

Lastly, in Chapter IV, we deployed a real world reinforcement learning model, collecting valuable data, created a generative model to simulate the gathering of more data, and worked on a novel RL approach to address challenges introduced by the domain. Future work here could start by considering potential improvements to our behavior modeling, and examining the effect this would have in the simulated RL domain. Following any improvements made there, another real-world deployment of the revised RL system would be invaluable. Then, data collected in that deployment should be evaluated, and the impact of the RL system improvements could be accurately measured, inviting even more improvements to tackle the hurdles of the domain.

APPENDIX A

APPENDIX TO CHAPTER 1

A.1 Experiment Domain - Bipartite Matching

Bipartite matching is a well established problem in graph learning. In form, it is essentially identical to the synthetic data setting previously discussed:

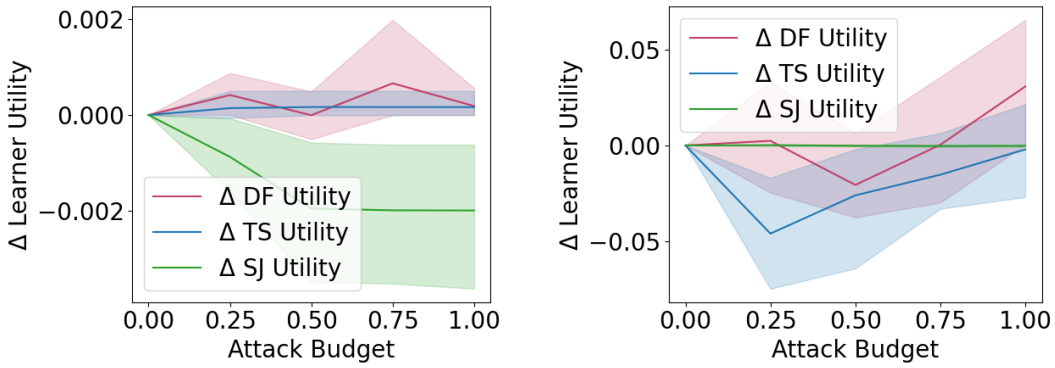
$$\min f(x, \theta) = \frac{1}{2} x^T Q x - \theta^T x \text{ s.t. } ||x|| \leq D, Ax \leq b \tag{A.1}$$

In this case, however, the constraints enforced are that x must be *doubly stochastic*. Intuitively, x is a square matrix with continuous values. Each value x_{ij} represents the probability that node i on one side of the graph will be matched with node j on the other side. For the learning component of the problem, the goal is to predict the graph's edges from the nodes' features. This means that ϵ represents per-node features, while θ is the graph's adjacency matrix (relaxed to be continuous).

For these experiments, we utilize the Cora dataset [118] which consists of scientific papers. The features here are binary values indicating the presence of keywords in the paper, while the edges in the graph are citations. In total, there are 1433 features and 2708 nodes. Inspired by a recent paper [146], we split the dataset into 27 bipartite subgraphs, with 50 nodes on each side in each subgraph. This is accomplished using Metis [58] to partition the graph.

A.2 Supplementary Experiment Results

In Figure A.1, we display the results of using Metapoison [53] to solve attacks against a simple joint learner, and transferring the found attack to the two-stage and decision focused learners. Both domains display the same trends as observed in our synthetic domain - namely, that the attack is only nominally effective against the simple joint model, and not at all effective when transferred to the other two models. Once again, this suggests that techniques from domains such

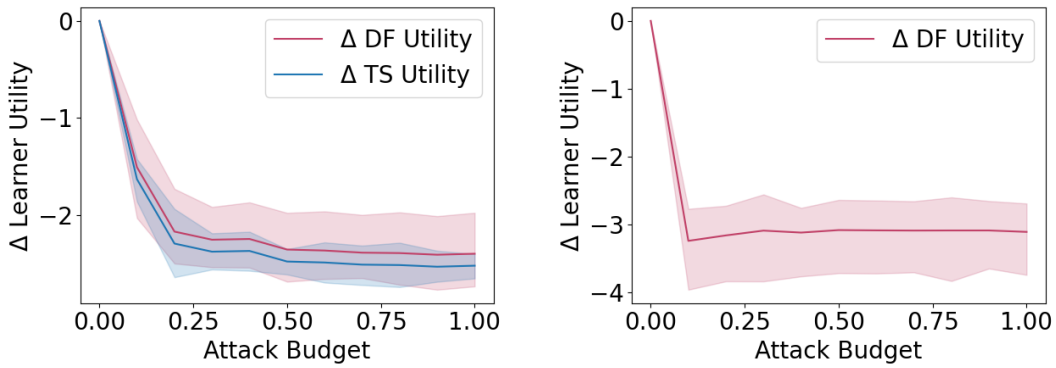


(a) Portfolio Optimization (b) Bipartite Matching

Figure A.1. Attacks generated on simple joint models

as computer vision may not be most appropriate for attacking data-based decision making models.

Figure A.2 shows the results when attacking two-stage and decision focused models for bipartite matching. The trends are once again similar to the other domains: attacks trained against a two-stage learner can effectively transfer to a decision focused learner. Furthermore, as in portfolio optimization, we observe that the decision focused learner appears more susceptible to direct attack (Figure A.2b) than is the two-stage learner (Figure A.2a). Once again, this is likely due to the decision focused learner outperforming the two-stage counterpart in the absence of attack. Unattacked, the two-stage learner achieves utility values between 2.37 and 2.90 while the decision focused learner obtains utilities between 2.65 and 4.59. Particularly when attacking the decision focused learner (Figure A.2b) we can observe the recurring trend of increased attack budgets often leading to *worse* attacks and *higher* utility for the learner, demonstrating the difficulties of finding good attack optima via (meta)gradient descent.



(a) Attack on two-stage learner

(b) Attack on decision focused learner

Figure A.2. Attacking a bipartite matching model

APPENDIX B

APPENDIX TO CHAPTER 2

B.1 Proof of Theorem 1

In order to prove this theorem, we introduce a series of lemmas (3–6). For the sake of analysis, we denote by:

$$y_i^m = \frac{z_i^m}{\sum_j z_j^m} \qquad\qquad c^m = \frac{1}{\sum_j z_j^m}$$

Intuitively, y_i^m is the empirical attack distribution estimated from the perturbed training data $\hat{\mathcal{D}} = \{x_i^m, z_i^m\}$ and c^m is the normalization term. Also, $\{y_i^m, c^m\}$ and $\{z_i^m\}$ are interchangeable. That is, given $\{y_i^m, c^m\}$, we can determine $z_i^m = \frac{y_i^m}{c^m}$. We first present the Lemma 1 which determines the deception capability of the deceptive attacker:

Lemma 3. *Given the true behavior λ^{true} of the non-deceptive attackers and the attack ratio f, the deceptive space for the deceptive attacker is specified as follows:*

$$\sum\nolimits_m \frac{1}{c^m}\left[\sum\nolimits_i y_i^m U_i^a(x_i^m) - U^a(\mathbf{x}^m, \lambda)\right] = 0 \tag{B.1}$$

$$\frac{y_i^m}{c^m} \geq n_i^m, \forall m, i \tag{B.2}$$

$$c^m \geq \frac{1}{(f+1)\sum_i n_i^m}, \forall m \tag{B.3}$$

$$y_i^m \in [0,1], \sum\nolimits_i y_i^m = 1, \forall m, i \tag{B.4}$$

That is, any deceptive λ that the defender learns has to be a part of a feasible solution (λ, y_i^m, c^m) of the system (B.1–B.4). Conversely, given any feasible (λ, y_i^m, c^m) satisfying (B.1–B.4), the deceptive attacker can make the defender learn λ by inducing the following perturbed data:

$$z_i^m = \frac{y_i^m}{c^m}$$

Proof. Equation (B.1) is simply the KKT condition presented in the previous section with y_i^m and c^m substituted in. Similarly, the constraints (B.2–B.3) correspond to the constraints for the deception capability of the deceptive attacker in (4.5–4.6). Finally, the constraint (B.4) follows from the definition of y_i^m and ensures that $\sum_i \frac{z_i^m}{\sum_j z_j^m} = 1$ and $\frac{z_i^m}{\sum_j z_j^m} \leq 1$. □

According to Lemma 3, we now can prove Theorem 1 based on the characterization of the feasible solution domain of λ for the system (B.1–B.4). We denote by:

$$\mathcal{F}(\lambda, \{y_i^m, c^m\}) = \sum_m \frac{1}{c^m} \left[\sum_i y_i^m U_i^a(x_i^m) - U^a(\mathbf{x}^m, \lambda) \right]$$

the LHS of (B.1). In addition, we denote by $\mathbf{S} = \{(y_i^m, c^m) :$ conditions (B.2–B.4) are satisfied$\}$ the feasible region of (y_i^m, c^m) which satisfy the conditions (B.2-B.4). In the following, we provide Lemmas 4 and 5 which specify the range of $\mathcal{F}$ as a function of λ. Essentially, if the value of $\mathcal{F}$ contains the point zero, then λ is a feasible solution of the system (B.1–B.4). We will use this property to characterize the feasible region of λ.

Lemma 4. *Assume that, WLOG,* $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$ *for all* m. *Given a* λ, *the optimal solution to*

$$\mathcal{F}^{\max}(\lambda) = \max_{\{y_i^m, c^m\} \in \mathbf{S}} \mathcal{F}(\lambda, \{y_i^m, c^m\}) \tag{B.5}$$

is determined as follows:

$$c^m = \frac{1}{(f+1)\sum_i n_i^m} \tag{B.6}$$

$$y_i^m = n_i^m c^m, \ \textit{when } i < T \tag{B.7}$$

$$y_i^m = 1 - c^m \sum_{i=1}^{T-1} n_i^m \ \textit{when } i = T \tag{B.8}$$

Proof. First, $\mathcal{F}(\lambda, \{y_i^m, c^m\})$ can be reformulated as:

$$\sum_m \frac{1}{c^m}\left[U_T^a(x_T^m)+\sum_{i=1}^{T-1} y_i^m[U_i^a(x_i^m)-U_T^a(x_T^m)]-U^a(\mathbf{x}^m,\lambda)\right]$$

Under our assumption that $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$, we know that $[U_i^a(x_i^m) - U_T^a(x_T^m)]$ is a strictly non-positive term for all i. Thus, maximizing $\mathcal{F}$ involves minimizing y_i^m when $i < T$. From constraint (B.2), the minimum y_i^m for all i is $n_i^m c^m$. This gives us $y_i^m = n_i^m c^m$ when $i < T$. From constraint (B.4), we know that this leaves us with $y_i^m = 1 - c^m \sum_{i=1}^{T-1} n_i^m$ when $i = T$.

Finally, given this specification of $\{y_i^m\}$, the optimization problem (B.5) is reduced to:

$$\max_{c^m} \sum_m \sum_{i<T} n_i^m[U_i^a(x_i^m)-U_T^a(x_T^m)]+\frac{U_T^a(x_T^m)-U^a(\mathbf{x}^m,\lambda)}{c^m}$$
$$\text{s.t. } c^m \geq \frac{1}{(f+1)\sum_i n_i^m} \text{ and } c^m \leq \frac{1}{\sum_i n_i^m}, \forall m$$

in which the objective function comprises of two terms: the first term does not depend on $\{c^m\}$ and the second term is a decreasing function of c^m (since $U_T^a(x_T^m) - U^a(\mathbf{x}^m, \lambda) > 0$). Therefore, it is maximized when c^m is minimized, which is $c^m = \frac{1}{(f+1)\sum_i n_i^m}$, concluding the proof. □

Lemma 5. *Assume that, WLOG, $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$ for all m. Given a λ, the optimal solution to*

$$\mathcal{F}^{\min}(\lambda) = \min_{\{y_i^m, c^m\} \in \mathbf{S}} \mathcal{F}(\lambda, \{y_i^m, c^m\}) \tag{B.9}$$

is determined as follows:

$$c^m = \frac{1}{(f+1)\sum_i n_i^m} \tag{B.10}$$

$$y_i^m = n_i^m c^m, \text{ when } i > 1 \tag{B.11}$$

$$y_i^m = 1 - c^m \sum_{i=2}^{T} n_i^m \text{ when } i = 1 \tag{B.12}$$

The proof of Lemma 5 is similar. Finally, using Lemmas (4–5) and the approximation in Eq. 4.7, we obtain:

$$\mathcal{F}^{\max}(\lambda) = \sum_m \left[\sum_j n_j^m\right] \Bigg[U^a(\mathbf{x}^m, \lambda^{\text{true}}) + fU_T^a(x_T^m) - (f+1)U^a(\mathbf{x}^m, \lambda)\Bigg] \tag{B.13}$$

$$\mathcal{F}^{\min}(\lambda) = \sum_m \left[\sum_j n_j^m\right] \Bigg[U^a(\mathbf{x}^m, \lambda^{\text{true}}) + fU_1^a(x_1^m) - (f+1)U^a(\mathbf{x}^m, \lambda)\Bigg] \tag{B.14}$$

Observe that, given λ, $\mathcal{F}(\lambda, \cdot)$ is continuous in $\{y_i^m, c^m\}$. Therefore, given a λ', if $\mathcal{F}^{\max}(\lambda') \geq 0 \geq \mathcal{F}^{\min}(\lambda')$, there must exist $\{y_i^m, c^m\} \in \mathbf{S}$ such that $\mathcal{F}(\lambda', \{y_i^m, c^m\}) = 0$. In other words, λ' is a part of a feasible solution for (B.1–B.4). Conversely, if $\mathcal{F}^{\max}(\lambda') < 0$ or $\mathcal{F}^{\min}(\lambda') > 0$, it means λ' is not feasible for (B.1–B.4). Moreover, using Observation 1, we can infer that both $\mathcal{F}^{\max}$ and $\mathcal{F}^{\min}$ are continuous and decreasing in λ. We obtain Lemma 6 which states that feasible solutions of (B.1–B.4) form an interval.

Lemma 6. *Let us assume $\lambda_1 < \lambda_2$ are two feasible solutions of (B.1–B.4). Then any $\lambda \in [\lambda_1, \lambda_2]$ is also a feasible solution of the system.*

Proof. Since λ_1 and λ_2 are feasible solutions of (B.1–B.4), we obtain the inequalities:

$$\mathcal{F}^{\max}(\lambda_1) \geq 0 \geq \mathcal{F}^{\min}(\lambda_1)$$

$$\mathcal{F}^{\min}(\lambda_2) \geq 0 \geq \mathcal{F}^{\min}(\lambda_2)$$

For any $\lambda \in [\lambda_1, \lambda_2]$, since $\mathcal{F}^{\max}$ and $\mathcal{F}^{\min}$ are decreasing functions in λ, the following inequality holds true:

$$\mathcal{F}^{\max}(\lambda) \geq \mathcal{F}^{\max}(\lambda_2) \geq 0 \geq \mathcal{F}^{\min}(\lambda_1) \geq \mathcal{F}^{\min}(\lambda)$$

which implies that λ is also a feasible solution for (B.1–B.4), concluding the proof.

□

Lemma 7 specifies the interval $[\lambda_{\min}^{\text{learnt}}, \lambda_{\max}^{\text{learnt}}]$ of feasible λ values for (B.1–B.4).

Lemma 7. *There exist* $\lambda_{\max}^{\text{learnt}} \geq \lambda_{\min}^{\text{learnt}}$ *such that:*

$$\mathcal{F}^{\max}(\lambda_{\max}^{\text{learnt}}) = \mathcal{F}^{\min}(\lambda_{\min}^{\text{learnt}}) = 0,$$

which means $\lambda_{\min}^{\text{learnt}}$ *and* $\lambda_{\max}^{\text{learnt}}$ *are feasible solutions for (B.1–B.4) and any* $\lambda \notin [\lambda_{\min}^{\text{learnt}}, \lambda_{\max}^{\text{learnt}}]$ *is not a feasible solution for (B.1–B.4).*

Proof. As noted before, $\mathcal{F}^{\max}(\lambda)$ is a continuous and decreasing function in λ. On the other hand, we have:

$$\mathcal{F}^{\max}(\lambda = +\infty) = \sum_m \left[\sum_j n_j^m\right] \left[U^a(\mathbf{x}^m, \lambda^{\text{true}}) - U_T^a(x_T^m)\right] \leq 0$$

$$\mathcal{F}^{\max}(\lambda = -\infty) = \sum_m \left[\sum_j n_j^m\right] \Big[U^a(\mathbf{x}^m, \lambda^{\text{true}}) + f U_T^a(x_T^m) - (f+1) U_1^a(x_1^m))\Big] \geq 0$$

for all λ^{true} since $U^a(\mathbf{x}^m, \lambda^{\text{true}} = +\infty) = U_T^a(x_T^m)$ and $U^a(\mathbf{x}^m, \lambda^{\text{true}} = -\infty) = U_1^a(x_1^m)$ is the highest and lowest expected utilities for the attacker among all targets , respectively, and by Observation 1, $U^a(\mathbf{x}^m, \lambda^{\text{true}})$ is increasing in λ^{true}. Since $\mathcal{F}^{\max}(\lambda)$ is continuous, there must exist a value of $\lambda_{\max}^{\text{learnt}} \in (-\infty, +\infty)$ such that $\mathcal{F}^{\max}(\lambda_{\max}^{\text{learnt}}) = 0$. The proof for $\lambda_{\min}^{\text{learnt}}$ is similar.

Finally, for any $\lambda < \lambda_{\min}^{\text{learnt}}$, we have $\mathcal{F}^{\min}(\lambda) > \mathcal{F}^{\min}(\lambda_{\min}^{\text{learnt}}) = 0$ since $\mathcal{F}^{\min}$ is decreasing in λ. Similarly, for any $\lambda > \lambda_{\max}^{\text{learnt}}$, we have $\mathcal{F}^{\max}(\lambda) < \mathcal{F}^{\max}(\lambda_{\max}^{\text{learnt}}) = 0$. Both imply that λ is not feasible, concluding our proof. □

By combining Lemmas 3,6, and 7, we obtain Theorem 1.

Proof of Corollary 1.

Proof. Corollary 1 is deduced based on the monotonicity property of the attacker's utility (Observation 1). When $\lambda_1^{\text{true}} \leq \lambda_2^{\text{true}}$, we have $U^a(\mathbf{x}^m; \lambda_1^{\text{true}}) \leq U^a(\mathbf{x}^m; \lambda_2^{\text{true}})$ for all m. Based on the relationship between $U^a(\mathbf{x}^m; \lambda^{\text{true}})$ and $U^a(\mathbf{x}^m; \lambda_{\max}^{\text{learnt}})$ presented in Theorem 1, we readily obtain $\lambda_{\max,1}^{\text{learnt}} \leq \lambda_{\max,2}^{\text{learnt}}$. Similarly, we have: $\lambda_{\min,1}^{\text{learnt}} \leq \lambda_{\min,2}^{\text{learnt}}$. □

Proof of Corollary 2.

Proof. We first prove (4.8). Let's consider the true behavior parameters $\lambda_1^{\text{true}} \leq \lambda_2^{\text{true}}$. Based on Corollary 1, the corresponding optimal deception solutions have to belong to the deception ranges: $\texttt{DecAlter}(\lambda_1^{\text{true}}) \in [\lambda_{\min,1}^{\text{learnt}}, \lambda_{\max,1}^{\text{learnt}}]$ and $\texttt{DecAlter}(\lambda_2^{\text{true}}) \in [\lambda_{\min,2}^{\text{learnt}}, \lambda_{\max,2}^{\text{learnt}}]$ where $\lambda_{\min,1}^{\text{learnt}} \leq \lambda_{\min,2}^{\text{learnt}}$ and $\lambda_{\max,1}^{\text{learnt}} \leq \lambda_{\max,2}^{\text{learnt}}$. We have two cases:

The first case is when the deception ranges do not overlap, i.e., $(\lambda_{\max}^1 < \lambda_{\min}^2)$. In this case, it is apparent that $\texttt{DecAlter}(\lambda_1^{\text{true}}) < \texttt{DecAlter}(\lambda_2^{\text{true}})$.

The other case is when the ranges overlap (i.e., $\lambda_1^{max} \geq \lambda_2^{min}$). If the optimal deceptive value for one or both does not belong to the overlap, i.e., $\texttt{DecAlter}(\lambda_1^{\text{true}}) < \lambda_{\min,2}^{\text{learnt}}$ and/or $\texttt{DecAlter}(\lambda_2^{\text{true}}) > \lambda_{\max,1}^{\text{learnt}}$), the result is clearly the same as in our previous case ($\texttt{DecAlter}(\lambda_1^{\text{true}}) < \texttt{DecAlter}(\lambda_2^{\text{true}})$). On the other hand, if both values fall within the overlap, that is $\lambda_{\min,2}^{\text{learnt}} \leq \texttt{DecAlter}(\lambda_1^{\text{true}}), \texttt{DecAlter}(\lambda_2^{\text{true}}) \leq \lambda_{\max,1}^{\text{learnt}}$, both will take on the same value

($\texttt{DecAlter}(\lambda_1^{\text{true}}) = \texttt{DecAlter}(\lambda_2^{\text{true}})$). This is true because both deceptive values $\texttt{DecAlter}(\lambda_1^{\text{true}})$ and $\texttt{DecAlter}(\lambda_2^{\text{true}})$ are being optimized to maximize the same objective: the utility of the deceptive attacker (as shown in `DecAlter`).

Finally, (4.9) can be easily deduced based on (4.8). Let's consider $\texttt{DecAlter}(\lambda_1^{\text{true}}) < \texttt{DecAlter}(\lambda_2^{\text{true}})$. We can prove $\lambda_1^{\text{true}} < \lambda_2^{\text{true}}$ by contradiction. That is, we assume $\lambda_1^{\text{true}} \geq \lambda_2^{\text{true}}$. According to (4.8), it means $\texttt{DecAlter}(\lambda_1^{\text{true}}) \geq \texttt{DecAlter}(\lambda_2^{\text{true}})$, which is a contradiction. □

Proof of Corollary 3.

Proof. Corollary 3 is a direct result of Corollary 2. Indeed, since $\lambda_1^{\text{true}} \leq \lambda^{\text{true}} \leq \lambda_2^{\text{true}}$, we obtain the inequality among optimal deception solutions $\texttt{DecAlter}(\lambda_1^{\text{true}}) \leq \texttt{DecAlter}(\lambda^{\text{true}}) \leq \texttt{DecAlter}(\lambda_2^{\text{true}})$ as a result of Corollary 2. Therefore if $\texttt{DecAlter}(\lambda_1^{\text{true}}) = \texttt{DecAlter}(\lambda_2^{\text{true}})$, we obtain the optimal deception solution: $\texttt{DecAlter}(\lambda^{\text{true}}) = \texttt{DecAlter}(\lambda_1^{\text{true}})$. □

Proof of Lemma 1.

Proof. Corollary 2 says that the deception outcome $\lambda^{\text{learnt}} = \texttt{DecAlter}(\lambda^{\text{true}})$ is an increasing (not strict) function of λ^{true}, and additionally using Corollary 3, we can say that given some deception outcome λ^{learnt}, there exists (unknown) $\lambda_{\text{min}}^{\text{true}}, \lambda_{\text{max}}^{\text{true}}$ such that any $\lambda^{\text{true}} \in [\lambda_{\text{min}}^{\text{true}}, \lambda_{\text{max}}^{\text{true}}]$ leads to the same outcome $\lambda^{\text{learnt}} = \texttt{DecAlter}(\lambda^{\text{true}})$. Any λ outside of the range $[\lambda_{\text{min}}^{\text{true}}, \lambda_{\text{max}}^{\text{true}}]$ cannot lead to the deception outcome λ^{learnt}. Corollary 2 further implies that $\lambda_{\text{min}}^{\text{true}}$ and $\lambda_{\text{max}}^{\text{true}}$ are increasing functions of λ^{learnt}. □

Proof of Lemma 2.

Proof. Assume WLOG, $U_1^a(x_1^m) \leq U_2^a(x_2^m) \leq \cdots \leq U_T^a(x_T^m)$. We claim that $S(i, \lambda^{\text{true}}) = \sum_{j=1}^{i} q_j(\mathbf{x}^m; \lambda^{\text{true}})$ for $T > i \geq 1$ is decreasing (not strictly) in λ^{true}, or in other words, the upper bound of the i^{th} segment is decreasing (not strictly) for all i except $i = T$. This means that for any single fixed u value, increasing λ^{true} implies that $f_{\lambda^{\text{true}}}(u)$ is also increasing (or stays same) because the upper bound of the interval that u lies in shifts downwards as λ^{true} increases. $f_{\lambda^{\text{true}}}(u)$ increasing means a higher value target is chosen for attack. Thus, for fixed u, a higher λ^{true} implies that the empirical distribution places more (or equal) attacks on higher utility targets and hence $U^a(\mathbf{x}^m, E(u; \lambda^{\text{true}}))$ increases (not strictly) with λ^{true}. Finally, to prove our claim at the start of the proof, we show that the derivative of $S(i, \lambda^{\text{true}})$ is non-positive everywhere. Indeed, its derivative is computed as follows:

$$\sum_{j=1}^{i} q_j(\mathbf{x}^m; \lambda^{\text{true}}) U_j^a(x_j^m) - S(i, \lambda^{\text{true}}) U^a(\mathbf{x}^m; \lambda^{\text{true}})$$

$$= S(i, \lambda^{\text{true}}) \Big[\sum_{j=1}^{i} \frac{q_j(\mathbf{x}^m; \lambda^{\text{true}})}{S(i, \lambda^{\text{true}})} U_j^a(x_j^m) - U^a(\mathbf{x}^m; \lambda^{\text{true}}) \Big] \tag{B.15}$$

decomposing the attacker utility function $U^a(\mathbf{x}^m; \lambda^{\text{true}})$, as follows:

$$S(i, \lambda^{\text{true}}) \sum_{j=1}^{i} \frac{q_j(\mathbf{x}^m; \lambda^{\text{true}})}{S(i, \lambda^{\text{true}})} U_j^a(x_j^m) +$$

$$\Big(\sum_{j=i+1}^{T} q_j(\mathbf{x}^m; \lambda^{\text{true}}) \Big) \sum_{j=i+1}^{T} \frac{q_j(\mathbf{x}^m; \lambda^{\text{true}})}{\sum_{j=i+1}^{T} q_j(\mathbf{x}^m; \lambda^{\text{true}})} U_j^a(x_j^m)$$

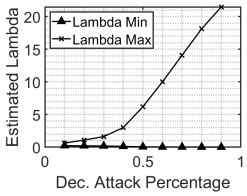
As we know that $U_1^a(x_1^m) \leq U_2^a(x_2^m) \ldots \leq U_T^a(x_T^m)$, the following inequality holds:

$$\sum_{j=i+1}^{T} \frac{q_j(\mathbf{x}^m; \lambda^{\text{true}})}{\sum_{j=i+1}^{T} q_j(\mathbf{x}^m; \lambda^{\text{true}})} U_j^a(x_j^m) \qquad \geq U_i^a(x_i^m) \geq \sum_{j=1}^{i} \frac{q_j(\mathbf{x}^m; \lambda^{\text{true}})}{S(i, \lambda^{\text{true}})} U_j^a(x_j^m)$$

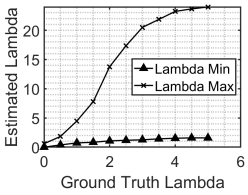
Using this we get:

$$U^a(\mathbf{x}^m; \lambda^{\text{true}}) \geq \Big(S(i, \lambda^{\text{true}}) + \sum_{j=i+1}^{T} q_j(\mathbf{x}^m; \lambda^{\text{true}}) \Big) \sum_{j=1}^{i} \frac{q_j(\mathbf{x}^m; \lambda^{\text{true}})}{S(i, \lambda^{\text{true}})} U_j^a(x_j^m)$$

$$= 1 \cdot \sum_{j=1}^{i} \frac{q_j(\mathbf{x}^m; \lambda^{\text{true}})}{S(i, \lambda^{\text{true}})} U_j^a(x_j^m)$$

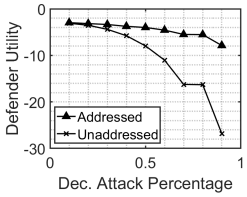


(a) Vary % of dec. attacks

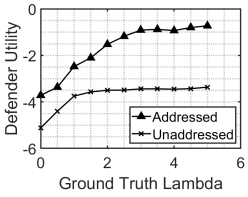


(b) Vary λ^{true}

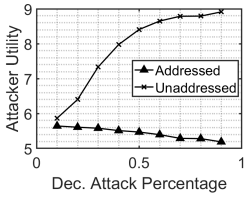
Figure B.1. Lambda Range Evaluation with 20 Targets



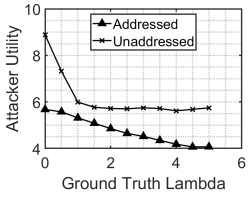
(a) Vary % of dec. attacks



(b) Vary λ^{true}



(c) Vary % of dec. attacks



(d) Vary λ^{true}

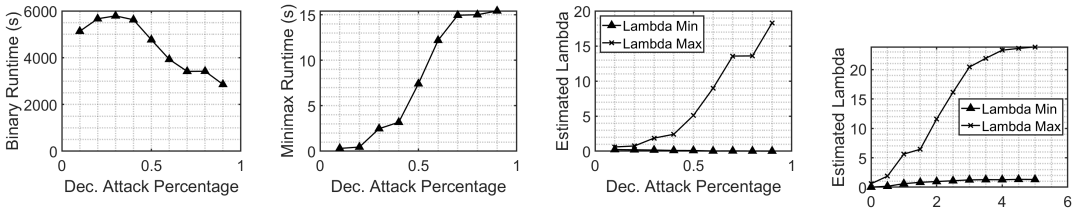
Figure B.2. Utility Evaluation with 30 Targets

Using the above in the derivative Eq. B.15, we get that the derivative of $S(i, \lambda^{\text{true}})$ is non-positive, hence it is decreasing w.r.t. λ, concluding our proof. □

Supplemental Experiments

First, in Figure B.1, we examine the range $[\lambda^{\text{true}}_{\text{min}}, \lambda^{\text{true}}_{\text{max}}]$ that the defender learns. Figure B.1a shows that the range increases w.r.t. the percentage of attacks controlled by the deceptive attacker. This is intuitive, as more manipulation gives more power to the deceptive attacker. Figure B.1b displays how this range also increases with the ground truth λ^{true} value of the non-deceptive attackers. As λ^{true} increases, the deceptive attacker produces a larger uncertainty range.

Lastly, Figures B.3 through B.2 are for 30-target games, and each corresponds to a previously discussed 20-target figure. We observe the same trends in both cases.



(a) Binary Search Runtime (b) Minimax Runtime (c) Vary % of dec. attacks (d) Vary λ^{true}

Figure B.3. Runtime and λ Evaluation with 30 Targets

Experimental Details

All experiments were run on the same HPC cluster, on instances using dual E5-2690v4 processors (28 cores). Each process was allocated 16000 megabytes of RAM. Instances run Red Hat Enterprise Linux Server, version 7.8. The Matlab version used was R2018b.

All experiments used the L-Infinity norm with a value of 2 as a *rejection threshold* for non-deceptive attack samples. This is done to prevent outlying samples from compromising the binary search. Values between .5 and 5 for this metric were tested, along with the same value ranges for the L1 and L2 norms. This norm and value were shown to produce the best results, without drastically increasing the runtime of the algorithm.

Additionally, all experiments used a value of 0.05 as a *tolerance multiplier within the binary search itself*. This prevents the inherent inaccuracy of discrete attack samples from ruining binary search. For the sake of consistency, an initial random number generation seed of 1 was used across all experiments. After defender strategy generation and solving (`DecAlter`), the binary search is run 10 times, each with a different random seed. The superset of all resulting ranges forms our final uncertainty set for λ^{true}.

The trials shown in Figures B.1a, 12a, 12c, 13, B.3c, B.2a, B.2c, and B.3 were conducted using a true lambda value of 0.4 and a resource/target ratio of 0.2. Those in Figures B.1b, 12b, 12d, B.3d, B.2b, and B.2d utilized a deceptive attack percentage of 0.3, and a resource/target ratio of 0.2. Experiments in Figures 13 and use deceptive attack percentage of 0.1, a true lambda value of 0.4, and a resource/target ratio of 0.2.

APPENDIX C

APPENDIX TO CHAPTER 3

Experiment Setup and Hyperparameters

All MuJoCo results represent the mean and standard deviation (shaded) over 10 random seeds, while results for the diabetes domain represent 40 random seeds. Within each run, episodic reward data points are collected by averaging a number of on-policy trajectories (10 for all MuJoCo domains, 50 for the diabetes messaging domain). All experiments were performed using Python 3.7.13, PyTorch 1.11.0, Gym 0.18.3, mujoco-py 2.1.2.14, and Mujoco 2.1.2.

Hyperparameters used for the transformer models match those used in the original papers. Hyperparameters for our methods match those used in RRD [114] unless otherwise specified in the following table:

Hyper-parameter	Value (MuJoCo)	Value (diabetes messaging)
discount factor γ	0.99	0.5
maximum episode length	1000	25
obs/action history length for state prediction	20	5
no. hidden layers (all non-recurrent networks)	2	2
size of hidden layers (all non-recurrent networks)	256	128
size of hidden layers (state predictive network)	128	128
transformer layer heads (state predictive network)	2	-
activation function	ReLU	ReLU
optimizer (for all components)	Adam	Adam
learning rate (actor/critic)	$3e^{-4}$	$3e^{-5}$
learning rate (α)	$3e^{-4}$	$3e^{-4}$
learning rate (state prediction)	$1e^{-4}$	$1e^{-4}$
max no. transitions in lessons buffer	$1e^{7}$	$1e^{5}$
min transitions stored to start training state predictor	$1e^{4}$	$1e^{4}$
min transitions stored to start training policy	$1e^{6}$	$1e^{4}$

Proof of Theorem 2: Optimality Bounds on Reward Redistribution

For the sake of presentation, we will π as a short representation of π_ψ in this proof. Given a reward model $\hat{R}_\theta$, we can decompose the loss of the reward redistribution in the presence of the state-prediction component as follows:

$$\begin{aligned}
\mathcal{L}^w_{RRD} &= \mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\bar{\tau}) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t)\Big)^2\Big] \\
&= \mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(\big[R_{ep}(\bar{\tau}) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t, a_t)\big] \\
&\qquad + \big[\frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t, a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t)\big]\Big)^2\Big] \\
&= \mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\bar{\tau}) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t, a_t)\Big)^2\Big] \\
&\qquad + 2\mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\bar{\tau}) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t, a_t)\Big) \\
&\qquad\qquad \cdot \Big(\frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t, a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t)\Big)\Big] \\
&\qquad + \mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(\frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t, a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t)\Big)^2\Big]
\end{aligned}$$

where the trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots)$ and the state-prediction trajectory $\bar{\tau} = (s_0, \bar{s}_0, a_0, s_1, \bar{s}_1, a_1, \cdots)$ with $s_0 \sim \rho_0(\cdot), \bar{s}_t \sim (f_w \circ q)(\cdot \mid s_{t-1}, a_{t-1}), a_t \sim \pi_\psi(\cdot \mid \bar{s}_t), s_{t+1} \sim P(\cdot \mid s_t, a_t)$ for all time steps $t = 0, 1, \cdots$ We are going to bound the above three terms in comparison with the original loss function $\mathcal{L}_{RRD}$.

First, in the presence of incomplete state observations and the state-prediction component, the probability that a trajectory τ occurs is computed as follows:

$$\bar{P}(\tau) = \prod_t P(s_{t+1} \mid s_t, a_t)\sum_{\bar{s}_t}\pi(a_t \mid \bar{s}_t)\cdot(f_w \circ q)(\bar{s}_t \mid s_{t-1}, a_{t-1})$$

On the other hand, when the states are fully observable, the probability that a trajectory τ occurs is computed as follows:

$$P(\tau) = \prod_t P(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t)$$

We obtain the trajectory-occurence ratio:

$$\frac{\bar{P}(\tau)}{P(\tau)} = \prod_t \sum_{\bar{s}_t} \frac{\pi(a_t \mid \bar{s}_t)}{\pi(a_t \mid s_t)} \cdot (f_w \circ q)(\bar{s}_t \mid s_{t-1}, a_{t-1})$$

Since $1 - C_p^l \cdot \epsilon \leq \frac{\pi(a|\bar{s})}{\pi(a|s)} \leq 1 + C_p^u \cdot \epsilon$, we obtain:

$$(1 - C_p^l \cdot \epsilon)^T \leq \frac{\bar{P}(\tau)}{P(\tau)} \leq (1 + C_p^u \cdot \epsilon)^T$$

As a result,

$$(1 - C_p^l \cdot \epsilon)^T \mathcal{L}_{RRD} \leq \mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\bar{\tau}) - \frac{T}{K}\sum_{t\in\mathcal{I}} \hat{R}_\theta(s_t, a_t)\Big)^2\Big] \leq (1 + C_p^u \cdot \epsilon)^T \mathcal{L}_{RRD}$$

Second, since $|\hat{R}_\theta(s, a) - \hat{R}_\theta(\bar{s}, a)| \leq C_r \cdot \epsilon$ for all (s, s') such that $||s - \bar{s}|| \leq \epsilon$ and all actions $a \in \mathcal{A}$, we obtain:

$$\mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(\frac{T}{K}\sum_{t\in\mathcal{I}} \hat{R}_\theta(s_t, a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}} \hat{R}_\theta(\bar{s}_t, a_t)\Big)^2\Big] \leq \big(\frac{T}{K}\big)^2 K(C_r \cdot \epsilon)^2 = \frac{T^2}{K}(C_r \cdot \epsilon)^2$$

Third, we have:

$$\begin{aligned}
&\mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\bar{\tau}) - \frac{T}{K}\sum_{t\in\mathcal{I}} \hat{R}_\theta(s_t, a_t)\Big)\Big(\frac{T}{K}\sum_{t\in\mathcal{I}} \hat{R}_\theta(s_t, a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}} \hat{R}_\theta(\bar{s}_t, a_t)\Big)\Big] \\
&\leq \frac{T}{K}(K \cdot C_r \cdot \epsilon) \cdot \mathbb{E}_{\bar{\tau}}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big|R_{ep}(\bar{\tau}) - \frac{T}{K}\sum_{t\in\mathcal{I}} \hat{R}_\theta(s_t, a_t)\Big|\Big] \\
&\leq (T \cdot C_r \cdot \epsilon)\sqrt{(1 + C_p^u \cdot \epsilon)^T \mathcal{L}_{RRD}} \qquad \text{(due to the Jensen's inequality)}
\end{aligned}$$

By combining the above inequalities, we obtain the final upper bound:

$$\mathcal{L}_{RRD}^w \leq (1 + C_p^u \cdot \epsilon)^T \mathcal{L}_{RRD} + (2 \cdot T \cdot C_r \cdot \epsilon)\sqrt{(1 + C_p^u \cdot \epsilon)^T \mathcal{L}_{RRD}} + \frac{T^2}{K}(C_r \cdot \epsilon)^2 \tag{C.1}$$

On the other hand, we can decompose the loss function when states are fully observed as follows:

$$
\begin{aligned}
\mathcal{L}_{RRD} &= \mathbb{E}_{\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\tau) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t,a_t)\Big)^2\Big] \\
&= \mathbb{E}_{\tau}\mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\tau) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t) + \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t) \\
&\qquad - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t,a_t)\Big)^2\Big] \\
&= \mathbb{E}_{\tau}\mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\tau) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t)\Big)^2\Big] \\
&\quad + \mathbb{E}_{\tau}\mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(\frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t,a_t)\Big)^2\Big] \\
&\quad + 2\mathbb{E}_{\tau}\mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\tau) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t)\Big) \\
&\qquad \cdot \Big(\frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t,a_t)\Big)\Big]
\end{aligned}
$$

Now we have:

$$
\frac{P(\bar{\tau})}{\bar{P}(\bar{\tau})} = \prod_t \frac{\pi(a_t \mid s_t)}{\pi(a_t \mid \bar{s}_t)} \leq \frac{1}{(1 - C_p^l \cdot \epsilon)^T}
$$

Therefore,

$$
\mathbb{E}_{\tau}\mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\tau) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t)\Big)^2\Big] \leq \frac{1}{(1 - C_p^l \cdot \epsilon)^T}\mathcal{L}_{RRD}^w
$$

In addition,

$$
\begin{aligned}
&\mathbb{E}_{\tau}\mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(\frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t,a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t,a_t)\Big)^2\Big] \\
&\leq \big(\frac{T}{K}^2\big)K(C_r \cdot \epsilon)^2 = \frac{T^2}{K}(C_r \cdot \epsilon)^2
\end{aligned}
$$

Third,

$$\mathbb{E}_\tau \mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big(R_{ep}(\tau) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t)\Big)\Big(\frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t) - \frac{T}{K}\sum_{t\in\mathcal{I}}\hat{R}_\theta(s_t, a_t)\Big)\Big]$$
$$\leq \frac{T}{K}(K\cdot C_r\cdot\epsilon)\mathbb{E}_\tau\mathbb{E}_{\bar{\tau}|\tau}\Big[\mathbb{E}_{\mathcal{I}\sim\rho_T(\cdot)}\Big|R_{ep}(\tau) - \sum_{t\in\mathcal{I}}\hat{R}_\theta(\bar{s}_t, a_t)\Big|\Big]$$
$$\leq (T\cdot C_r\cdot\epsilon)\sqrt{\frac{1}{(1-C_p^l\cdot\epsilon)^T}\mathcal{L}^w_{RRD}}$$

Finally, by combining these three inequalities together, we obtain:

$$\mathcal{L}_{RRD} \leq \frac{1}{(1-C_p^l\cdot\epsilon)^T}\mathcal{L}^w_{RRD} + (2\cdot T\cdot C_r\cdot\epsilon)\sqrt{\frac{1}{(1-C_p^l\cdot\epsilon)^T}\mathcal{L}^w_{RRD}} + \frac{T^2}{K}(C_r\cdot\epsilon)^2 \tag{C.2}$$

In short, by combining (C.1) and (C.2), we have: for a given reward function θ,

$$\mathcal{L}^w_{RRD}(\theta) \leq A_1(\epsilon)\mathcal{L}_{RRD}(\theta) + B_1(\epsilon)\sqrt{\mathcal{L}_{RRD}(\theta)} + C_1(\epsilon)$$
$$\mathcal{L}_{RRD}(\theta) \leq A_2(\epsilon)\mathcal{L}^w_{RRD}(\theta) + B_2(\epsilon)\sqrt{\mathcal{L}^w_{RRD}(\theta)} + C_2(\epsilon)$$

where the coefficients $A_1(\epsilon) = (1+C_p^u\cdot\epsilon)^T$, $B_1(\epsilon) = (2\cdot T\cdot C_r\cdot\epsilon)\sqrt{(1+C_p^u\cdot\epsilon)^T}$, and $C_1(\epsilon) = \frac{T^2}{K}(C_r\cdot\epsilon)^2$. In addition, the coefficients $A_2(\epsilon) = \frac{1}{(1-C_p^l\cdot\epsilon)^T}$, $B_2(\epsilon) = (2\cdot T\cdot C_r\cdot\epsilon)\sqrt{\frac{1}{(1-C_p^l\cdot\epsilon)^T}}$, and $C_2(\epsilon) = \frac{T^2}{K}(C_r\cdot\epsilon)^2$.

Let's denote by: $\bar{\theta}^* \in \text{argmin}_\theta\, \mathcal{L}^w_{RRD}(\theta)$ and $\theta^* \in \text{argmin}_\pi\, \mathcal{L}_{RRD}(\theta)$. We are going to bound $\mathcal{L}_{RRD}(\bar{\theta}^*)$ in comparison with $\mathcal{L}_{RRD}(\theta^*)$ as follows:

$$\begin{aligned}
\mathcal{L}_{RRD}(\bar{\theta}^*) &\leq A_2(\epsilon)\mathcal{L}^w_{RRD}(\bar{\theta}^*) + B_2(\epsilon)\sqrt{\mathcal{L}^w_{RRD}(\bar{\theta}^*)} + C_2(\epsilon) \\
&\leq A_2(\epsilon)\mathcal{L}^w_{RRD}(\theta^*) + B_2(\epsilon)\sqrt{\mathcal{L}^w_{RRD}(\theta^*)} + C_2(\epsilon) \\
&\leq A_2(\epsilon)\Big[A_1(\epsilon)\mathcal{L}_{RRD}(\theta^*) + B_1(\epsilon)\sqrt{\mathcal{L}_{RRD}(\theta^*)} + C_1(\epsilon)\Big] \\
&\qquad + B_2(\epsilon)\sqrt{A_1(\epsilon)\mathcal{L}_{RRD}(\theta^*) + B_1(\epsilon)\sqrt{\mathcal{L}_{RRD}(\theta^*)} + C_1(\epsilon)} + C_2(\epsilon)
\end{aligned}$$

which concludes our proof.

Proof of Theorem 3: Optimality Bounds on Policy Learning

We first define the policy learning objective in three different scenarios:

– Without reward error and state error:

$$J(\hat{R}_{\theta^*}, \pi) = \mathbb{E}_\tau \sum_t \gamma^t \hat{R}_{\theta^*}(s_t, a_t)$$

where $s_0 \sim P_0$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, $a_t \sim \pi(\cdot \mid s_t)$

– Without state error:

$$J(\hat{R}_{\bar{\theta}^*}, \pi) = \mathbb{E}_\tau \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(s_t, a_t)$$

where $s_0 \sim P_0$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, $a_t \sim \pi(\cdot \mid s_t)$

– With both reward and state errors:

$$J^w(\hat{R}_{\bar{\theta}^*}, \pi) = \mathbb{E}_{\bar{\tau}} \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(\bar{s}_t, a_t)$$

where $s_0 \sim P_0$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, $a_t \sim \pi(\cdot \mid \bar{s}_t)$, $\bar{s}_t \sim (f_w \circ q)(\cdot \mid s_{t-1}, a_{t-1})$

We are going to compare these three objective functions for a given policy π. First, we have:

$$|J(\hat{R}_{\bar{\theta}^*}, \pi) - M(\epsilon) J(\hat{R}_{\theta^*}, \pi)| \leq \mathbb{E}_\tau \sum_t \gamma^t |\hat{R}_{\bar{\theta}^*}(s, a) - M(\epsilon) \hat{R}_{\theta^*}(s, a)| \leq \frac{\Delta(\epsilon)}{1 - \gamma}$$

Second, we have:

$$\begin{aligned} J^w(\hat{R}_{\bar{\theta}^*}, \pi) &= \mathbb{E}_{\bar{\tau}} \Big[\sum_t \gamma^t [\hat{R}_{\bar{\theta}^*}(\bar{s}_t, a_t) - \hat{R}_{\bar{\theta}^*}(s_t, a_t)] + \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(s_t, a_t) \Big] \\ &\leq \frac{C_r \cdot \epsilon}{1 - \gamma} + \mathbb{E}_{\bar{\tau}} \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(s_t, a_t) \end{aligned}$$

We use a similar idea as in reward redistribution:

$$(1 - C_p^l \cdot \epsilon)^T \leq \frac{\bar{P}(\tau)}{P(\tau)} \leq (1 + C_p^u \cdot \epsilon)^T$$

Therefore,

$$\mathbb{E}_{\bar{\tau}} \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(s_t, a_t) = \sum_\tau \bar{P}(\tau) \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(s_t, a_t) \leq (1 + C_p^u \cdot \epsilon)^T J(\hat{R}_{\bar{\theta}^*}, \pi)$$
$$\implies J^w(\hat{R}_{\bar{\theta}^*}, \pi) \leq \frac{C_r \cdot \epsilon}{1-\gamma} + (1 + C_p^u \cdot \epsilon)^T J(\hat{R}_{\bar{\theta}^*}, \pi)$$

On the other hand,

$$J(\hat{R}_{\bar{\theta}^*}, \pi) = \mathbb{E}_\tau \mathbb{E}_{\bar{\tau}|\tau} \Big[\sum_t \gamma^t [\hat{R}_{\bar{\theta}^*}(s_t, a_t) - \hat{R}_{\bar{\theta}^*}(\bar{s}_t, a_t)] + \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(\bar{s}_t, a_t) \Big]$$
$$\leq \frac{C_r \cdot \epsilon}{1-\gamma} + \mathbb{E}_\tau \mathbb{E}_{\bar{\tau}|\tau} \sum_t \gamma^t \hat{R}_{\bar{\theta}^*}(\bar{s}_t, a_t)$$

Note that:

$$\frac{P(\bar{\tau})}{\bar{P}(\bar{\tau})} = \prod_t \frac{\pi(a_t \mid s_t)}{\pi(a_t \mid \bar{s}_t)} \leq \frac{1}{(1 - C_p^l \cdot \epsilon)^T}$$

Therefore,

$$J(\hat{R}_{\bar{\theta}^*}, \pi) \leq \frac{C_r \cdot \epsilon}{1-\gamma} + \frac{1}{(1 - C_p^l \cdot \epsilon)^T} J^w(\hat{R}_{\bar{\theta}^*}, \pi)$$

As a result, we obtain the final bound:

$$J^w(\hat{R}_{\bar{\theta}^*}, \pi) \leq \frac{C_r \cdot \epsilon}{1-\gamma} + (1 + C_p^u \cdot \epsilon)^T \Big[M(\epsilon) J(R, \pi) + \frac{\Delta(\epsilon)}{1-\gamma} \Big]$$
$$J(R, \pi) \leq \frac{1}{M(\epsilon)} \Big[\frac{\Delta(\epsilon)}{1-\gamma} + \frac{C_r \cdot \epsilon}{1-\gamma} + \frac{1}{(1 - C_p^l \cdot \epsilon)^T} J^w(\hat{R}_{\bar{\theta}^*}, \pi) \Big]$$

Now, let's denote by $\pi^* \in \text{argmax}_\pi J(R, \pi)$ and $\bar{\pi}^* \in \text{argmax}_\pi J^w(\hat{R}_{\bar{\theta}^*}, \pi)$, we are going to bound:

$$J(R, \bar{\pi}^*) \geq \frac{1}{(1 + C_p^u \cdot \epsilon)^T \cdot M(\epsilon)} \Big[J^w(\hat{R}_{\bar{\theta}^*}, \bar{\pi}^*) - \frac{C_r \cdot \epsilon}{1-\gamma} \cdot \frac{\Delta(\epsilon)}{1-\gamma} \Big]$$
$$\geq \frac{1}{(1 + C_p^u \cdot \epsilon)^T \cdot M(\epsilon)} \Big[J^w(\hat{R}_{\bar{\theta}^*}, \pi^*) - \frac{C_r \cdot \epsilon}{1-\gamma} \cdot \frac{\Delta(\epsilon)}{1-\gamma} \Big]$$
$$\geq \frac{1}{(1 + C_p^u \cdot \epsilon)^T \cdot M(\epsilon)}$$
$$\cdot \Big[\Big[M(\epsilon) \cdot (1 - C_p^l \cdot \epsilon)^T \Big[J(R, \pi^*) - \frac{\Delta(\epsilon) + C_r \cdot \epsilon}{M(\epsilon) \cdot (1-\gamma)} \Big] \Big] - \frac{C_r \cdot \epsilon}{1-\gamma} \cdot \frac{\Delta(\epsilon)}{1-\gamma} \Big]$$

which concludes our proof.

REFERENCES CITED

[1] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. *Advances in neural information processing systems 32* (2019).

[2] Ahn, I., and Park, J. Drug scheduling of cancer chemotherapy based on natural actor-critic approach. *BioSystems 106*, 2-3 (2011), 121–129.

[3] Akoglu, L., Tong, H., and Koutra, D. Graph-based anomaly detection and description: A survey, 2014.

[4] Albarran, S. E., and Clempner, J. B. A stackelberg security markov game based on partial information for strategic decision making against unexpected attacks. *Engineering Applications of Artificial Intelligence 81* (2019), 408 – 419.

[5] Amos, B., and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. *arXiv preprint arXiv:1703.00443* (2017).

[6] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent, 2016.

[7] Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. Rudder: Return decomposition for delayed rewards, 2019.

[8] Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., and Hochreiter, S. RUDDER: return decomposition for delayed rewards. *CoRR abs/1806.07857* (2018).

[9] Asoh, H., Akaho, M. S. S., Kamishima, T., Hasida, K., Aramaki, E., and Kohro, T. An application of inverse reinforcement learning to medical records of diabetes treatment. In *ECMLPKDD2013 workshop on reinforcement learning with generalized feedback* (2013).

[10] Bengio, Y. Gradient-Based Optimization of Hyperparameters. *Neural Computation 12*, 8 (08 2000), 1889–1900.

[11] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases* (2013), Springer, pp. 387–402.

[12] Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[13] Bojchevski, A., and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations* (2018).

[14] Bojchevski, A., and Günnemann, S. Adversarial attacks on node embeddings via graph poisoning, 2019.

[15] Bothe, M. K., Dickens, L., Reichel, K., Tellmann, A., Ellger, B., Westphal, M., and Faisal, A. A. The use of reinforcement learning algorithms to meet the challenges of an artificial pancreas. *Expert review of medical devices 10*, 5 (2013), 661–673.

[16] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

[17] Brown, A. L., Camerer, C. F., and Lovallo, D. To review or not to review? limited strategic thinking at the movie box office. *American Economic Journal: Microeconomics 4*, 2 (May 2012), 1–26.

[18] Butler, A. R., Nguyen, T. H., and Sinha, A. Countering attacker data manipulation in security games. In *Decision and Game Theory for Security* (Cham, 2021), B. Bošanský, C. Gonzalez, S. Rass, and A. Sinha, Eds., Springer International Publishing, pp. 59–79.

[19] Camerer, C. F., Ho, T.-H., and Chong, J.-K. A Cognitive Hierarchy Model of Games*. *The Quarterly Journal of Economics 119*, 3 (08 2004), 861–898.

[20] Chen, J., Chen, L., Chen, Y., Zhao, M., Yu, S., Xuan, Q., and Yang, X. Ga-based q-attack on community detection. *IEEE Transactions on Computational Social Systems 6*, 3 (2019), 491–503.

[21] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems 34* (2021), 15084–15097.

[22] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[23] Chen, Y., Nadji, Y., Kountouras, A., Monrose, F., Perdisci, R., Antonakakis, M., and Vasiloglou, N. Practical attacks against graph-based clustering. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (oct 2017), ACM.

[24] Chu, T., Wang, J., and Chen, J. An adaptive online learning framework for practical breast cancer diagnosis. In *Medical imaging 2016: Computer-aided diagnosis* (2016), vol. 9785, SPIE, pp. 537–548.

[25] Corbett, A. T., and Anderson, J. R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction 4* (1994), 253–278.

[26] Daskalaki, E., Diem, P., and Mougiakakou, S. G. Personalized tuning of a reinforcement learning control algorithm for glucose regulation. In *2013 35th Annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (2013), IEEE, pp. 3487–3490.

[27] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum, 2017.

[28] Donti, P., Amos, B., and Kolter, J. Z. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems* (2017), pp. 5484–5494.

[29] Efroni, Y., Merlis, N., and Mannor, S. Reinforcement learning with trajectory feedback. In *AAAI Conference on Artificial Intelligence* (2020).

[30] Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Tambe, M., Lemieux, A., et al. Deploying paws: Field optimization of the protection assistant for wildlife security. In *AAAI* (2016), vol. 16, pp. 3966–3973.

[31] Fang, F., Stone, P., and Tambe, M. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Proceedings of the 24th International Conference on Artificial Intelligence* (2015), IJCAI'15, AAAI Press, p. 2589–2595.

[32] Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364* (2021).

[33] Gan, J., Guo, Q., Tran-Thanh, L., An, B., and Wooldridge, M. Manipulating a learning defender and ways to counteract. In *NIPS-19* (2019).

[34] Gan, J., Xu, H., Guo, Q., Tran-Thanh, L., Rabinovich, Z., and Wooldridge, M. Imitative follower deception in stackelberg games. In *EC '19* (2019).

[35] Gangwani, T., Zhou, Y., and Peng, J. Learning guidance rewards with trajectory-space smoothing. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2020), NIPS'20, Curran Associates Inc.

[36] Gaweda, A. E., Muezzinoglu, M. K., Aronoff, G. R., Jacobs, A. A., Zurada, J. M., and Brier, M. E. Individualization of pharmacological anemia management using reinforcement learning. *Neural Networks 18*, 5-6 (2005), 826–834.

[37] Gaweda, A. E., Muezzinoglu, M. K., Jacobs, A. A., Aronoff, G. R., and Brier, M. E. Model predictive control with reinforcement learning for drug delivery in renal anemia management. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society* (2006), IEEE, pp. 5177–5180.

[38] Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' brew: Industrial scale data poisoning via gradient matching, 2020.

[39] Geiping, J., Fowl, L., Somepalli, G., Goldblum, M., Moeller, M., and Goldstein, T. What doesn't kill you makes you robust(er): How to adversarially train against data poisoning, 2022.

[40] Ghesu, F.-C., Georgescu, B., Zheng, Y., Grbic, S., Maier, A., Hornegger, J., and Comaniciu, D. Multi-scale deep reinforcement learning for real-time 3d-landmark detection in ct scans. *IEEE transactions on pattern analysis and machine intelligence 41*, 1 (2017), 176–189.

[41] Ghosh, A., Heffernan, N., and Lan, A. S. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (2020), pp. 2330–2339.

[42] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2014.

[43] Grefenstette, E., Amos, B., Yarats, D., Htut, P. M., Molchanov, A., Meier, F., Kiela, D., Cho, K., and Chintala, S. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727* (2019).

[44] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks, 2016.

[45] Guo, Q., An, B., Bosansky, B., and Kiekintveld, C. Comparing strategic secrecy and Stackelberg commitment in security games. In *IJCAI* (2017).

[46] Hassani, A., et al. Reinforcement learning based control of tumor growth with chemotherapy. In *2010 International Conference on System Science and Engineering* (2010), IEEE, pp. 185–189.

[47] Hochberg, I., Feraru, G., Kozdoba, M., Mannor, S., Tennenholtz, M., and Yom-Tov, E. A reinforcement learning system to encourage physical activity in diabetes patients. *arXiv preprint arXiv:1605.04070* (2016).

[48] Hochreiter, S., and Schmidhuber, J. Lstm can solve hard long time lag problems. In *Proceedings of the 9th International Conference on Neural Information Processing Systems* (Cambridge, MA, USA, 1996), NIPS'96, MIT Press, p. 473–479.

[49] Hortaçsu, A., Luco, F., Puller, S. L., and Zhu, D. Does strategic ability affect efficiency? evidence from electricity markets. *AER 109*, 12 (December 2019), 4302–42.

[50] Hu, Y., Wang, W., Jia, H., Wang, Y., Chen, Y., Hao, J., Wu, F., and Fan, C. Learning to utilize shaping rewards: a new approach of reward shaping. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2020), NIPS'20, Curran Associates Inc.

[51] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence* (2011), pp. 43–58.

[52] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. Adversarial machine learning. In *AISec* (2011).

[53] Huang, W. R., Geiping, J., Fowl, L., Taylor, G., and Goldstein, T. Metapoison: Practical general-purpose clean-label data poisoning. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 12080–12091.

[54] Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018), IEEE, pp. 19–35.

[55] Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems 34* (2021), 1273–1286.

[56] Kao, H.-C., Tang, K.-F., and Chang, E. Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence* (2018), vol. 32.

[57] Kar, D., Ford, B., Gholami, S., Fang, F., Plumptre, A., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., and Nsubaga, M. Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In *AAMAS '17* (2017).

[58] Karypis, G., and Kumar, V. Metis—a software package for partitioning unstructured graphs, partitioning meshes and computing fill-reducing ordering of sparse matrices.

[59] Killian, J. A., Wilder, B., Sharma, A., Choudhary, V., Dilkina, B., and Tambe, M. Learning to prescribe interventions for tuberculosis patients using digital adherence data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 2430–2438.

[60] Kingma, D. P. Auto-encoding variational bayes. In *ICLR* (2014).

[61] Kinsey, S., Wolf, J., Saligram, N., Ramesan, V., Walavalkar, M., Jaswal, N., Ramalingam, S., Sinha, A., and Nguyen, T. Building a personalized messaging system for health intervention in underprivileged regions using reinforcement learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence* (2023), pp. 6022–6030.

[62] Kinsey, S. E., Tuck, W. W., Sinha, A., and Nguyen, T. H. An exploration of poisoning attacks on data-based decision making. In *Decision and Game Theory for Security* (Cham, 2023), F. Fang, H. Xu, and Y. Hayel, Eds., Springer International Publishing, pp. 231–252.

[63] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks, 2017.

[64] Klicpera, J., Bojchevski, A., and Günnemann, S. Personalized embedding propagation: Combining neural networks on graphs with personalized pagerank. *CoRR abs/1810.05997* (2018).

[65] Krantz, S. G., and Parks, H. R. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[66] Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. In *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[67] Li, J., Liao, J., Wu, R., Chen, L., Dan, J., Meng, C., Zheng, Z., and Wang, W. Guard: Graph universal adversarial defense, 2022.

[68] Liu, X., Si, S., Zhu, X., Li, Y., and Hsieh, C.-J. A unified framework for data poisoning attack to graph-based semi-supervised learning, 2019.

[69] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, 2019.

[70] Lowd, D., and Meek, C. Adversarial learning. In *ACM SIGKDD* (2005).

[71] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks, 2017.

[72] Markowitz, H. Portfolio selection. *The Journal of Finance 7*, 1 (1952), 77–91.

[73] Martín-Guerrero, J. D., Gomez, F., Soria-Olivas, E., Schmidhuber, J., Climente-Martí, M., and Jiménez-Torres, N. V. A reinforcement learning approach for individualizing erythropoietin dosages in hemodialysis patients. *Expert Systems with Applications 36*, 6 (2009), 9737–9742.

[74] Mate, A., Madaan, L., Taneja, A., Madhiwalla, N., Verma, S., Singh, G., Hegde, A., Varakantham, P., and Tambe, M. Field study in deploying restless multi-armed bandits: Assisting non-profits in improving maternal and child health, 2021.

[75] McKelvey, R. D., and Palfrey, T. R. Quantal response equilibria for normal form games. In *Games and economic behavior* (1995).

[76] Minervini, P., Demeester, T., Rocktäschel, T., and Riedel, S. Adversarial sets for regularising neural link predictors, 2017.

[77] Miranda, J. J., Kinra, S., Casas, J. P., Davey Smith, G., and Ebrahim, S. Non-communicable diseases in low-and middle-income countries: context, determinants and health policy. *Tropical Medicine & International Health 13*, 10 (2008), 1225–1234.

[78] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature 518*, 7540 (2015), 529–533.

[79] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Los Alamitos, CA, USA, jul 2017), IEEE Computer Society, pp. 5425–5434.

[80] Moore, B. L., Pyeatt, L. D., Kulkarni, V., Panousis, P., Padrez, K., and Doufas, A. G. Reinforcement learning for closed-loop propofol anesthesia: a study in human volunteers. *The journal of machine learning research 15*, 1 (2014), 655–696.

[81] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks, 2015.

[82] Mukhopadhyay, A., and Vorobeychik, Y. Prioritized allocation of emergency responders based on a continuous-time incident prediction model. In *International Conference on Autonomous Agents and MultiAgent Systems* (2017).

[83] Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (2017).

[84] Muñoz-González, L., Pfitzner, B., Russo, M., Carnerero-Cano, J., and Lupu, E. C. Poisoning attacks with generative adversarial nets, 2019.

[85] Nguyen, T. H., Butler, A., and Xu, H. Tackling imitative attacker deception in repeated bayesian stackelberg security games. In *European Conference on Artificial Intelligence* (2020).

[86] Nguyen, T. H., and Sinha, A. The art of manipulation: Threat of multi-step manipulative attacks in security games, 2022.

[87] Nguyen, T. H., Sinha, A., Gholami, S., Plumptre, A., Joppa, L., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Critchlow, R., et al. Capture: A new predictive anti-poaching tool for wildlife protection. In *AAMAS '16* (2016), pp. 767–775.

[88] Nguyen, T. H., Sinha, A., and He, H. Partial adversarial behavior deception in security games. In *IJCAI* (2020).

[89] Nguyen, T. H., Vu, N., Yadav, A., and Nguyen, U. Decoding the imitation security game: Handling attacker imitative behavior deception. In *24th European Conference on Artificial Intelligence* (2020).

[90] Nguyen, T. H., Wang, Y., Sinha, A., and Wellman, M. P. Deception in finitely repeated security games. In *AAAI-19* (2019).

[91] Nguyen, T. H., Yang, R., Azaria, A., Kraus, S., and Tambe, M. Analyzing the effectiveness of adversary modeling in security games. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence* (2013), AAAI'13, AAAI Press, p. 718–724.

[92] Noori, A., Sadrnia, M. A., et al. Glucose level control using temporal difference methods. In *2017 Iranian Conference on Electrical Engineering (ICEE)* (2017), IEEE, pp. 895–900.

[93] Organization, W. H. *Noncommunicable diseases country profiles 2018*. World Health Organization, 2018.

[94] Padmanabhan, R., Meskin, N., and Haddad, W. M. Reinforcement learning-based control of drug dosing for cancer chemotherapy treatment. *Mathematical biosciences 293* (2017), 11–20.

[95] Pandey, S., and Karypis, G. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837* (2019).

[96] Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[97] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)* (2016), IEEE, pp. 372–387.

[98] Parbhoo, S., Bogojeska, J., Zazzi, M., Roth, V., and Doshi-Velez, F. Combining kernel and model based learning for hiv therapy selection. *AMIA Summits on Translational Science Proceedings 2017* (2017), 239.

[99] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction, 2017.

[100] Patil, V. P., Hofmarcher, M., Dinu, M.-C., Dorfer, M., Blies, P. M., Brandstetter, J., Arjona-Medina, J. A., and Hochreiter, S. Align-rudder: Learning from few demonstrations by reward redistribution, 2022.

[101] Peng, B., Shen, W., Tang, P., and Zuo, S. Learning optimal strategies to commit to. In *33th AAAI Conference on Artificial Intelligence* (2019).

[102] Perozzi, B., Al-Rfou, R., and Skiena, S. DeepWalk. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (aug 2014), ACM.

[103] Perrault, A., Wilder, B., Ewing, E., Mate, A., Dilkina, B., and Tambe, M. Decision-focused learning of adversary behavior in security games. *CoRR abs/1903.00958* (2019).

[104] Pfammatter, A., Spring, B., Saligram, N., Davé, R., Gowda, A., Blais, L., Arora, M., Ranjani, H., Ganda, O., Hedeker, D., et al. mhealth intervention to improve diabetes risk behaviors in india: a prospective, parallel group cohort study. *Journal of medical Internet research 18*, 8 (2016), e207.

[105] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., and Sohl-Dickstein, J. Deep knowledge tracing. *Advances in neural information processing systems 28* (2015).

[106] Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., and Kraus, S. Deployed armor protection: The application of a game theoretic model for security at the los angeles international airport. pp. 125–132.

[107] Pita, J., John, R., Maheswaran, R., Tambe, M., and Kraus, S. A robust approach to addressing human adversaries in security games. In *ECAI 2012*. IOS Press, 2012, pp. 660–665.

[108] Price, R. A useful theorem for nonlinear devices having gaussian inputs. *IEEE Trans. Inf. Theory 4* (1958).

[109] Quandl. WIKI various end-of-day data, 2021.

[110] Rabinovich, Z., Jiang, A. X., Jain, M., and Xu, H. Information disclosure as a means to security. In *AAMAS '15* (2015), pp. 645–653.

[111] Rajeswar, S., Ibrahim, C., Surya, N., Golemo, F., Vazquez, D., Courville, A., and Pinheiro, P. O. Haptics-based curiosity for sparse-reward tasks. In *5th Annual Conference on Robot Learning* (2021).

[112] Ranjani, H., Nitika, S., Anjana, R. M., Ramalingam, S., Mohan, V., Saligram, N., et al. Impact of noncommunicable disease text messages delivered via an app in preventing and managing lifestyle diseases: Results of the "myarogya" worksite-based effectiveness study from india. *Journal of Diabetology 11*, 2 (2020), 90.

[113] Ren, Z., Guo, R., Zhou, Y., and Peng, J. Learning long-term reward redistribution via randomized return decomposition. *CoRR abs/2111.13485* (2021).

[114] Ren, Z., Guo, R., Zhou, Y., and Peng, J. Learning long-term reward redistribution via randomized return decomposition, 2022.

[115] Rozsa, A., Rudd, E. M., and Boult, T. E. Adversarial diversity and hard positive generation, 2016.

[116] Sahba, F., Tizhoosh, H. R., and Salama, M. M. Application of reinforcement learning for segmentation of transrectal ultrasound images. *BMC medical imaging 8* (2008), 1–10.

[117] Saria, S. Individualized sepsis treatment using reinforcement learning. *Nature medicine 24*, 11 (2018), 1641–1642.

[118] Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine 29*, 3 (2008), 93–106.

[119] Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems 31* (2018).

[120] Shah, S., Sinha, A., Varakantham, P., Perrault, A., and Tambe, M. Solving online threat screening games using constrained action space reinforcement learning. *CoRR abs/1911.08799* (2019).

[121] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. Protect: A deployed game theoretic system to protect the ports of the united states. In *AAMAS* (2012).

[122] Sinha, A., Kar, D., and Tambe, M. Learning adversary behavior in security games: A pac model perspective. In *AAMAS '16* (2016).

[123] Sinzinger, E. D., and Moore, B. Sedation of simulated icu patients using reinforcement learning based control. *International Journal on Artificial Intelligence Tools 14*, 01n02 (2005), 137–156.

[124] Song, Y., Ma, C., Wu, X., Gong, L., Bao, L., Zuo, W., Shen, C., Lau, R. W., and Yang, M.-H. Vital: Visual tracking via adversarial learning. In *IEEE CVPR* (2018).

[125] Stewart, G. Perturbation theory for the singular value decomposition, 1990.

[126] Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation 23*, 5 (oct 2019), 828–841.

[127] Su, Y., Liu, Q., Liu, Q., Huang, Z., Yin, Y., Chen, E., Ding, C., Wei, S., and Hu, G. Exercise-enhanced sequential modeling for student performance prediction. *Proceedings of the AAAI Conference on Artificial Intelligence 32*, 1 (Apr. 2018).

[128] Sun, L., Dou, Y., Yang, C., Wang, J., Yu, P. S., He, L., and Li, B. Adversarial attack and defense on graph data: A survey, 2020.

[129] Sun, M., Tang, J., Li, H., Li, B., Xiao, C., Chen, Y., and Song, D. Data poisoning attack against unsupervised node embedding methods, 2018.

[130] Sun, Y., Ma, S., Madaan, R., Bonatti, R., Huang, F., and Kapoor, A. Smart: Self-supervised multi-task pretraining with control transformers. *arXiv preprint arXiv:2301.09816* (2023).

[131] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks, 2013.

[132] Tambe, M. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[133] Tambwekar, P., Dhuliawala, M., Martin, L. J., Mehta, A., Harrison, B., and Riedl, M. O. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (Aug. 2019), IJCAI-2019, International Joint Conferences on Artificial Intelligence Organization.

[134] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. LINE. In *Proceedings of the 24th International Conference on World Wide Web* (may 2015), International World Wide Web Conferences Steering Committee.

[135] Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), IEEE, pp. 5026–5033.

[136] Tseng, W.-C., Wang, T.-H. J., Lin, Y.-C., and Isola, P. Offline multi-agent reinforcement learning with knowledge distillation. *Advances in Neural Information Processing Systems 35* (2022), 226–237.

[137] Udeshi, S., Peng, S., Woo, G., Loh, L., Rawshan, L., and Chattopadhyay, S. Model agnostic defence against backdoor attacks in machine learning. *IEEE Transactions on Reliability 71*, 2 (2022), 880–895.

[138] von Stengel, B. Leadership with commitment to mixed strategies.

[139] Wang, B., Jia, J., Cao, X., and Gong, N. Z. Certified robustness of graph neural networks against adversarial structural perturbation, 2021.

[140] Wang, H., Xie, H., Qiu, L., Yang, Y. R., Zhang, Y., and Greenberg, A. Cope: Traffic engineering in dynamic networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications* (2006), pp. 99–110.

[141] Wang, K., Wilder, B., Perrault, A., and Tambe, M. Automatically learning compact quality-aware surrogates for optimization problems. *Advances in Neural Information Processing Systems 33* (2020), 9586–9596.

[142] Waniek, M., Michalak, T. P., Wooldridge, M. J., and Rahwan, T. Hiding individuals and communities in a social network. *Nature Human Behaviour 2*, 2 (jan 2018), 139–147.

[143] Weber, M., Xu, X., Karlaš, B., Zhang, C., and Li, B. Rab: Provable robustness against backdoor attacks, 2022.

[144] Weng, C.-H., Lee, Y.-T., and Wu, S.-H. B. On the trade-off between adversarial and backdoor robustness. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 11973–11983.

[145] Wilcox, R. *Applying contemporary statistical techniques*. Academic Press, 2002.

[146] Wilder, B., Dilkina, B., and Tambe, M. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization, 2018.

[147] Wilder, B., Ewing, E., Dilkina, B., and Tambe, M. End to end learning and optimization on graphs, 2020.

[148] Wilder, B., Yadav, A., Immorlica, N., Rice, E., and Tambe, M. Uncharted but not uninfluenced: Influence maximization with an uncertain network. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (2017), pp. 1305–1313.

[149] Wright, J. R., and Leyton-Brown, K. Level-0 meta-models for predicting human behavior in games. In *EC '14* (2014), ACM.

[150] Xi, Z., Pang, R., Ji, S., and Wang, T. Graph backdoor, 2020.

[151] Xiao, Y., Li, J., and Su, W. A lightweight metric defence strategy for graph neural networks against poisoning attacks. In *Information and Communications Security* (Cham, 2021), D. Gao, Q. Li, X. Guan, and X. Liao, Eds., Springer International Publishing, pp. 55–72.

[152] Xiong, X., Zhao, S., Van Inwegen, E. G., and Beck, J. E. Going deeper with deep knowledge tracing. *International Educational Data Mining Society* (2016).

[153] Xu, L., Gholami, S., McCarthy, S., Dilkina, B., Plumptre, A., Tambe, M., Singh, R., Nsubuga, M., Mabonga, J., Driciru, M., et al. Stay ahead of poachers: Illegal wildlife poaching prediction and patrol planning under uncertainty with field test evaluations (short version). In *2020 IEEE 36th International Conference on Data Engineering (ICDE)* (2020), IEEE, pp. 1898–1901.

[154] Xue, Y., Davies, I., Fink, D., Wood, C., and Gomes, C. P. Avicaching: A two stage game for bias reduction in citizen science. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (2016), pp. 776–785.

[155] Yadav, A., Chan, H., Jiang, A. X., Xu, H., Rice, E., and Tambe, M. Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In *AAMAS* (2016), vol. 16, pp. 740–748.

[156] Yadav, A., Wilder, B., Rice, E., Petering, R., Craddock, J., Yoshioka-Maxwell, A., Hemler, M., Onasch-Vera, L., Tambe, M., and Woo, D. Bridging the gap between theory and practice in influence maximization: Raising awareness about hiv among homeless youth. In *IJCAI* (2018), pp. 5399–5403.

[157] Yan, B., and Gregory, S. Detecting community structure in networks using edge prediction methods. *Journal of Statistical Mechanics: Theory and Experiment 2012*, 09 (sep 2012), P09008.

[158] Yang, C., Wu, Q., Li, H., and Chen, Y. Generative poisoning attack method against neural networks, 2017.

[159] Yang, R., Kiekintveld, C., Ordonez, F., Tambe, M., and John, R. Improving resource allocation strategy against human adversaries in security games. In *IJCAI* (2011).

[160] Yeung, C.-K., and Yeung, D.-Y. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the fifth annual ACM conference on learning at scale* (2018), pp. 1–10.

[161] Yom-Tov, E., Feraru, G., Kozdoba, M., Mannor, S., Tennenholtz, M., and Hochberg, I. Encouraging physical activity in patients with diabetes: intervention using a reinforcement learning system. *Journal of medical Internet research 19*, 10 (2017), e338.

[162] Yu, C., Dong, Y., Liu, J., and Ren, G. Incorporating causal factors into reinforcement learning for dynamic treatment regimes in hiv. *BMC medical informatics and decision making 19*, 2 (2019), 19–29.

[163] Yu, C., Liu, J., Nemati, S., and Yin, G. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR) 55*, 1 (2021), 1–36.

[164] Yuan, X., He, P., Zhu, Q., and Li, X. Adversarial examples: Attacks and defenses for deep learning, 2017.

[165] Zeng, Y., Chen, S., Park, W., Mao, Z. M., Jin, M., and Jia, R. Adversarial unlearning of backdoors via implicit hypergradient, 2022.

[166] Zhang, J., Wang, Y., and Zhuang, J. Modeling multi-target defender-attacker games with quantal response attack strategies. *Reliability Engineering & System Safety 205* (2021).

[167] Zhang, M., Hu, L., Shi, C., and Wang, X. Adversarial label-flipping attack and defense for graph neural networks. pp. 791–800.

[168] Zhang, W., Guo, H., Ranganathan, P., Patel, J., Rajasekharan, S., Danayak, N., Gupta, M., and Yadav, A. A continual pre-training approach to tele-triaging pregnant women in kenya.

[169] Zhang, X., Zhu, X., and Lessard, L. Online data poisoning attack, 2019.

[170] Zhang, Z., Jia, J., Wang, B., and Gong, N. Z. Backdoor attacks to graph neural networks, 2020.

[171] Zhao, Y., Kosorok, M. R., and Zeng, D. Reinforcement learning design for cancer clinical trials. *Statistics in medicine 28*, 26 (2009), 3294–3315.

[172] Zheng, H., Xiong, H., Chen, J., Ma, H., and Huang, G. Motif-backdoor: Rethinking the backdoor attack on graph neural networks via motifs, 2022.

[173] Zheng, L., Chen, J., Wang, J., He, J., Hu, Y., Chen, Y., Fan, C., Gao, Y., and Zhang, C. Episodic multi-agent reinforcement learning with curiosity-driven exploration, 2021.

[174] Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In *international conference on machine learning* (2022), PMLR, pp. 27042–27059.

[175] Zhuang, J., Bier, V. M., and Alagoz, O. Modeling secrecy and deception in a multi-period attacker-defender signaling game. *European Journal of Operational Research 203* (2010), 409–418.

[176] Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2018), KDD '18, Association for Computing Machinery, p. 2847–2856.

[177] Zügner, D., and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)* (2019).