June 2021

SAFE AND PRACTICAL MACHINE LEARNING

Stephen J. Giguere
University of Massachusetts Amherst

SAFE AND PRACTICAL MACHINE LEARNING

Stephen J. Giguere

SAFE AND PRACTICAL MACHINE LEARNING

A Dissertation Presented

by

STEPHEN GIGUERE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2021

College of Information and Computer Sciences

SAFE AND PRACTICAL MACHINE LEARNING

A Dissertation Presented

by

STEPHEN GIGUERE

Approved as to style and content by:

____

Philip Thomas, Co-chair

____

Yuriy Brun, Co-chair

____

Daniel Sheldon, Member

____

Melinda D. Dyar, Member

____

James Allan, Department Chair
College of Information and Computer Sciences

ABSTRACT

SAFE AND PRACTICAL MACHINE LEARNING

MAY 2021

STEPHEN GIGUERE

B.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

M.Sc., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS, AMHERST

Directed by: Professor Philip Thomas and Professor Yuriy Brun

As increasingly sensitive decision making problems become automated using models trained by machine learning algorithms, it is important for machine learning researchers to design training algorithms that provide assurance that the models they produce will be well behaved. While significant progress has been made toward designing safe machine learning algorithms, there are several obstacles that prevent these strategies from being useful in practice. In this defense, I will highlight two of these challenges, and provide methods and results demonstrating that they can be overcome.

First, for many applications, the user must be able to easily specify general and potentially complex definitions of unsafe behavior. While most existing safe machine learning algorithms make strong assumptions about how unsafe behavior is defined, I will describe a flexible interface that allows the user to specify their definitions in a straightforward way at training time, and that is general enough to enforce a wide range of commonly used definitions.

Second, users often require guarantees to hold even when a trained model is deployed into an environment that differs from the training environment. In these settings, the safety guarantees provided by existing methods are no longer valid when the environment changes, presenting significant risk. I will consider two instances of this problem. In the first instance, I will provide algorithms with safety guarantees that hold when the differences between the training and deployment environments are caused by a change in the probability of encountering certain classes of observations. These algorithms are particularly useful in social applications, where the distribution of protected attributes, such as race or sex, may change over time. Next, I will provide algorithms with safety guarantees that hold in more general settings, in which the differences between the training and deployment environments are more challenging to describe. In both settings, I will present experiments showing that the guarantees provided by these algorithms are valid in practice, even when these changes are made antagonistically.

TABLE OF CONTENTS

Page

CHAPTER

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

As increasingly sensitive real-world decision making problems become automated using predictive models trained by machine learning algorithms, the need for techniques that assess the safety of these models is similarly growing. Consider, for example, self-driving cars, machines that guide medical policy and practice, and general purpose robotic workers, all of which have the potential to revolutionize our lives for the better. [36, 51, 28]. However, the positive impacts of these applications come with an increased need for safety measures. For example, in the applications mentioned above, failure of a machine learning model might cause physical injury or significant damages. In other applications, such as predicting or assessing criminal activity [2, 8], using facial recognition software for travel authorization [22], or automating job recruiting strategies [11], deploying poorly-behaved models carries significant risk of causing social injustice. Because of the variety of ways in which adverse effects can manifest, it is important to consider the behavior of the models trained by machine learning algorithms based on definitions that encompass physical safety, fairness, and more.

Unfortunately, the task of designing algorithms that consistently produce well-behaved or safe models is nontrivial due to several challenges that must be considered. First, as the previous examples illustrate, the manner in which unsafe behavior is defined tends to be application specific, introducing a tradeoff between how flexible a given algorithm can be with regard to supporting various safety definitions, and how difficult it is to design algorithms that enforce them. In addition, many real-world safety applications require high confidence that a model will be safe, meaning that these algorithms must provide some form of guarantee on safe behavior for them to be useful.

Finally, the rapid adoption of machine learning techniques across many industries and disciplines means that it is often unrealistic to assume that a practitioner seeking to automate a particular task will have significant expertise in machine learning. Consequently, safe machine learning algorithms that require significant amounts of data analysis in order to properly tune parameters, specify inputs, or define constraints, are of limited use to many practitioners.

Recent work has produced many safe machine learning algorithms that address these challenges to varying degrees. For example, algorithms based on chance constraints are able to offer some assurance that learned models will behave safety, but often require extensive knowledge of the task to use effectively [4]. On the other hand, algorithms based on soft constrained programming can produce models that are empirically safe, but they do not offer guarantees of safe behavior and may require detailed data analysis in order to properly set certain parameters [4]. These approaches and others will be described in more detail in Section 1.2.2. In this dissertation, we propose algorithms designed according to the Seldonian framework for machine learning [54], which provides a formulation for reasoning about these considerations, as well as principles for designing machine learning algorithms that provide high-confidence safety guarantees.

Despite recent progress, existing safe machine learning algorithms remain limited by several practical challenges that have not yet been fully addressed. First, we claim that while some algorithms provide reasonable flexibility in defining safety, they do not yet provide the flexibility needed for users to easily enforce the types of safety constraints that arise in many problem settings. Secondly, we claim that existing safe algorithms tend to ignore the possibility that the environment used to train a model might differ from the environment it is eventually deployed in. We will discuss this problem in detail in Chapter 3 and Chapter 4, but in summary, differences between the training and deployment environments cause many safety guarantees provided by safe algorithms to be unreliable. As a result of these challenges, we propose that while existing safe machine learning algorithms provide significant progress towards the goal of providing practical tools for preventing undesirable behavior, further developments are necessary for these tools to be useful in many settings.

In this dissertation, we present strategies for overcoming these practical challenges. First, we provide background on how safety constraints can be modeled in the machine learning setting, and discuss the Seldonian machine learning framework and related principles that guide the development of safe machine learning algorithms. Given this background, we identify two central problems that must be overcome in order to enforce general, user-defined safety constraints in practical settings. The first challenge is that the algorithm designer must provide an *interface* that allows safety to be defined in a general way. To address this, we identify a general class of safety specifications and provide algorithms for parsing and enforcing them. The second challenge is that in many practical applications, it is important to ensure that a model will behave safely once it is deployed, even when the data that is encountered in the future is drawn from a different distribution than was encountered during training. We refer to this setting by stating that *distribution shift* has occurred between training and deploying the model. In particular, this uncertainty in the distribution of data after deployment means that safety guarantees provided by Seldonian algorithms and others may not apply once the model is deployed. To address this, we present two solutions, based on distinct assumptions about what is known about the distribution shift when the model is trained. First, we address the case in which little is known about how the distribution will change besides the fact that the size of the change will be bounded with respect to some distance measure. In particular, we will provide algorithms that satisfy high-probability safety guarantees provided that the *Kullback-Liebler divergence* (KLD) between the training and deployed data distributions are bounded. These bounds are extremely general, and can therefore be applied in many practical applications. Nonetheless, in other applications, the user might know that the training and deployment distributions differ in particular, well-understood ways. For example, if a classifier is trained on data for students, it may be known that the future distribution may be the same as in training, except that the relative proportion of male and female students might differ. In such cases, the fact that the differences between the training and deployed data distributions can be explained by changes in a single variable—in this case, sex—can be exploited to enforce safety guarantees based on information about how the distribution of that variable will change after deployment. We present safe machine learning algorithms based on this approach in Chapter 3, and demonstrate that the safety guarantees they provide hold with high probability.

1.1 Background: Seldonian Machine Learning

In light of the potentially adverse consequences to deploying machine learning-trained models, it is reasonable to consider how to define safety formally. This entails specifying the object that we should consider the safety of, how safety should be quantified for that object, and what assurances of safety are needed. For example, given a machine learning algorithm that outputs some trained model, one might require the model to not discriminate based on a sensitive feature such as race, and might require assurance that the model behaves safely on some held-out validation set. However, this formulation fails when the safety constraints are impossible to satisfy, and provides no guarantees on how safe the trained model will be when applied to new, unseen data. The *Seldonian machine learning framework* [54] provides a formulation for reasoning about these choices in a consistent manner.

Importantly, the Seldonian machine learning framework formulates safety as a property of the machine learning algorithm itself, rather than a property of any particular trained model. It assumes that safety can be quantified using a scalar function, called a *constraint objective*, such that positive values indicate unsafe behavior. Furthermore, according to this framework, a safe algorithm should allow its user to specify what constitutes unsafe behavior; this definition should not be hard-coded into the algorithm. This requirement is largely at odds with existing approaches to creating algorithms with safety guarantees, which typically have a definition of safety ingrained in the algorithm's design, with some exceptions [1]. To be considered safe under this framework, a learning algorithm must satisfy a set of *behavioral constraints*, which limit the frequency with which the algorithm returns models that produce unsafe behavior.

To make to make these concepts formal, let θ be some predictive model. For example, in classification, θ might map feature vectors X to labels Y, or it might map states S to actions A in the reinforcement learning setting. Regardless of the problem setting, we generally evaluate θ according to some loss function, ℓ. Conventional machine learning algorithms are typically designed to return a model that (approximately) minimizes ℓ. Specifically, an algorithm, a, uses input data, D, to output a model, $a(D)$, that satisfies,

$$\ell(a(D)) \approx \min_{\theta\in\Theta} \ell(\theta).$$

To reason about safety within this general setting, the Seldonian machine learning framework assumes one or more functions g are provided by the user, which are used to assess whether or not a model behaves safely according to the user's application-specific requirements. In this dissertation, we assume a single definition for notational convenience, but our results generalize readily to the case where multiple definitions have been provided. Concretely, the Seldonian machine learning framework assumes that g measures the prevalence of unsafe outcomes when using a given model . Specifically, the framework assumes that $g(\theta) = 0$ represents a threshold for safe behavior, so that θ is considered safe enough to deploy as long as $g(\theta) \leq 0$.

Given g, one might seek to design an algorithm that is guaranteed to return safe models. While it may be possible to design algorithms that achieve this goal for specific applications and specific definitions of safety, it is most helpful to have algorithms that do not assume that g is known until the algorithm is run. Unfortunately, it is generally impossible to ensure safety with absolute certainty without making strong assumptions about how g will be defined. Furthermore, g often depends on quantities such as expected values that, in practice, can only be estimated. Consequently, it is usually impractical to determine with absolute certainty whether or not $g(\theta) \leq 0$ for any θ.

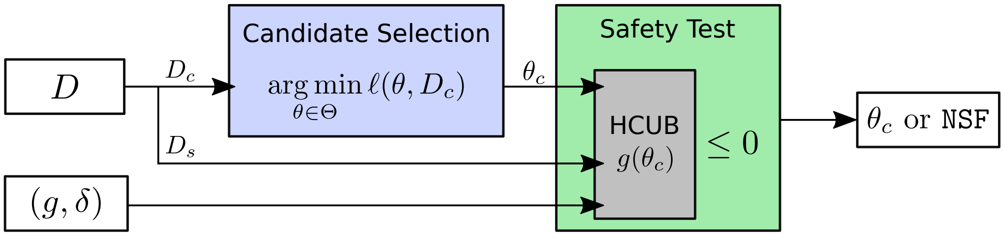

Figure 1.1. An illustration of a prototype for designing algorithms that satisfy the requirements of the Seldonian machine learning framework. Given a definition of unsafe behavior, g, and a tolerance for how often the algorithm can return a solution that is unsafe, δ, these algorithms satisfy the behavioral constraints in (1.1) by applying a *safety test* that determines whether a candidate solution, θ_c, is safe with high confidence. If the safety test is passed, the model is returned, whereas if the test is failed, the algorithm returns `NO_SOLUTION_FOUND` as a safe default. The safety test itself is based on computing a high-confidence upper bound on the value of $g(\theta_c)$, and to ensure it is statistically independent from the process used to train the candidate, separate partitions of the input data, D, are used for each component of the algorithm.

Therefore, the Seldonian framework proposes that safety constraints should be *probabilistic*, with the user being able to specify an upper bound on the probability that the algorithm produces a model that is unsafe. Concretely, the Seldonian framework defines safe algorithms to be those that satisfy a *behavioral constraint* for each definition g, defined by,

$$\Pr\big(g\big(a(D)\big) \leq 0\big) \geq 1-\delta, \tag{1.1}$$

where g and δ are provided by the user when the algorithm, a, is executed. In particular, we highlight that the random quantity used to define the probability in (1.1) is the input data D, which will generally be defined as a set of i.i.d. observations from from some distribution that is relevant to the user's application.

Importantly, the Seldonian framework defines a set of requirements for algorithms to be considered safe, but does not specify how they should be constructed, or how g should be defined. However, recent works thus far have identified one successful strategy [54, 44], which is illustrated in Figure 1.1. At a high level, a set of data is held out during training and used to perform

Algorithm 1 SeldonianPrototype(D, g, δ)

```
$D_c, D_s \leftarrow$ Partition($D$)
$\{$Select a candidate$\}$
$\theta_c \leftarrow$ TrainCandidate($D_c, g, \delta$)
$\{$Perform the safety test using $\theta_c\}$
$u \leftarrow$ HighConfidenceUpperBound($g, D_s, \delta$)
if $u \leq 0$ then
    return $\theta_c$
else
    return NO_SOLUTION_FOUND
end if
```

a *safety test* to determine if the resulting model is safe to deploy. The first split of data is used to train a candidate solution, θ_c, in a step referred to as *candidate selection*. Once a candidate solution is found, the safety test is applied. To perform the safety test, the held out data is used to compute a high-confidence upper bound the constraint objective, $g(\theta_c)$; if the value of the high-confidence upper bound is positive, then θ_c is possibly unsafe and `NO_SOLUTION_FOUND` is returned, where $g(\texttt{NO\_SOLUTION\_FOUND}) := 0$ by definition. If the safety test is passed, then the candidate model, θ_c, is returned by the algorithm. In this way, the above strategy can be seen as a method of censoring the output of the learning algorithm—returning `NO_SOLUTION_FOUND` instead—to ensure that the models it produces are unsafe with tolerable frequencies.

To simplify the steps involved with upper bounding $g(\theta_c)$, initial Seldonian algorithms made the simplifying assumption that g was defined as an expected value that can be estimated without bias using the held out data. We explain this assumption and illustrate it's limitations in Chapter 2, where we propose a new formulation that is much more flexible.

Algorithm 1 provides pseudocode for a prototypical Seldonian algorithm. In that algorithm, `Partition` splits the input data, D, into a set to be used for training, D_c, and a set to be used for performing the safety test, D_s. Then, a candidate model, θ_c, is trained using `TrainCandidate`. While the details of this step can have a significant impact on both the quality of the solution returned and the likelihood of the algorithm returning `NO_SOLUTION_FOUND`, they do not affect

Algorithm 2 SeldonianRegression($D \coloneqq \{(X_i, Y_i)\}_{i=1}^n$, g, δ)

```
Assume: $g(\theta) \coloneqq \mathbf{E}[h(X,\theta)] - \tau$, where $h(x,\theta) \in [0,1]$
{Partition $D$}
$D_c \leftarrow \{(X_i, Y_i)\}_{i=1}^{n/2}$
$D_s \leftarrow \{(X_i, Y_i)\}_{i=n/2+1}^{n}$
{Select a candidate that minimizes MSE}
$\theta_c \leftarrow \arg\min_{\theta \in \Theta} \frac{1}{|D_c|} \sum_{(X,Y) \in D_c} \big(\theta(X) - Y\big)^2$
{Perform the safety test using $\theta_c$ using Hoeffding's inequality}
$u \leftarrow \frac{1}{|D_s|} \sum_{(X,Y) \in D_s} h(X, \theta_c) + \sqrt{\frac{\log(1/\delta)}{2|D_s|}} - \tau$
if $u \leq 0$ then
    return $\theta_c$
else
    return NO_SOLUTION_FOUND
end if
```

whether or not the algorithm satisfies the conditions to be a Seldonian algorithm, and are therefore not elaborated on in this dissertation [54]. Finally, a safety test is performed on θ_c, which involves computing a $(1-\delta)$-probability upper bound on $g(\theta_c)$ using `HighConfidenceUpperBound`, and using the result to determine whether the algorithm should return θ_c or `NO_SOLUTION_FOUND`.

To make this process concrete, consider a regression task in which the goal is to train a linear model, θ, that accurately predicts a real-valued response, $Y \in \mathcal{R}$, associated with a set of real valued features, $X \in \mathcal{R}^d$, given a set of n identically and independently distributed (i.i.d.) labeled observations, $D \coloneqq \{(X_i, Y_i)\}_{i=1}^n$. Next, assume the user would like to ensure that, with confidence at least $1-\delta$, the model will not cause some negative event to occur for more than $100\tau\%$ of the observations. For example, to be risk-sensitive, they may want to ensure that the model will not over- or under-estimate the true response by more than some amount. As a result, we assume the user defines $g(\theta) = \mathbf{E}[h(X,\theta)] - \tau$, where $h(X,\theta)$ returns one if the model causes the negative event to occur for input X, and zero if not. To design a Seldonian algorithm for this setting using the prototype described in Algorithm 1, we might first decide to partition the training data evenly into D_c and D_s. Next, if Θ denotes the set of all linear models, we might define `TrainCandidate` to minimize the mean squared error (MSE) of the model's predictions on D_c:

$$\theta_c = \arg\min_{\theta \in \Theta} \frac{1}{|D_c|} \sum_{(X,Y) \in D_c} \big(\theta(X) - Y\big)^2.$$

Finally, to construct a safety test, we might leverage the fact that $h(X, \theta) \in \{0, 1\}$ and use D_s to compute a high-confidence upper bound on $g(\theta_c)$ based on Hoeffding's inequality [32], which satisfies,

$$\Pr\big(g(\theta_c) \leq U_{\texttt{Hoeff}}(\theta_c, D_s)\big) \geq 1{-}\delta,$$

where

$$U_{\texttt{Hoeff}}(\theta_c, D_s) := \frac{1}{|D_s|} \sum_{(X,Y) \in D_s} h(X, \theta_c) + \sqrt{\frac{\log(1/\delta)}{2|D_s|}} - \tau.$$

Combining these steps, Algorithm 2 defines a Seldonian regression algorithm for minimizing MSE that provides high-confidence guarantees based on the safety definitions described above.

1.1.1 Advantages and Disadvantages of Seldonian Machine Learning

Algorithms designed according to the Seldonian framework offer users several advantages for ensuring safety compared to prior approaches. In particular, they provide the user with high-confidence safety guarantees without requiring that they perform significant data analysis. For example, algorithms based on hard constraints require the user to encode their definition of undesirable behavior as a constraint on the set of model parameters. If, for example, the user would like to train a neural network that depends on many weights, then they must perform extensive data analysis to determine which weights produce models that are safe. Similarly, approaches based on soft constraints require the user to set a tradeoff parameter that balances safety and model performance, which might require significant effort to set properly. In contrast, Seldonian algorithms provide high-confidence safety guarantees given the user's safety definition directly. In addition, these algorithms allow the user to control the maximum probability with which the algorithm produces an unsafe output, and are therefore useful for a wide range of problem settings, ranging from those where safety considerations are relaxed, to those where ensuring safety is critical. Finally, these algorithms provide high-confidence guarantees, which make them desirable for safety-critical applications compared to approaches that are safe empirically, but do not provide formal assurances.

On the other hand, Seldonian algorithms have some drawbacks. First, since the Seldonian machine learning framework places the burden of establishing high-confidence guarantees of safety on the algorithm designer, these algorithms can be challenging to design. In particular, the high-confidence safety guarantees specified by (1.1) can be difficult to establish in general settings. The prototypical Seldonian algorithm presented in Algorithm 1 achieves this by using a held-out portion of the training data to apply a safety test, but this strategy results in less data being used for training a candidate model. As a result, these algorithms have lower data efficiency than safety-agnostic alternatives. A related consideration when designing Seldonian algorithms is that, to ensure the validity of the high-confidence safety guarantees, it can be challenging to evaluate the safety of multiple models without access to significant amounts of data. In Algorithm 1, the safety test is applied to a single candidate model for this reason. However, since the safety test is only applied to a single model, the candidate selection step must be carefully designed so that the model that is selected is likely to pass the safety test [54]. In addition to these considerations, the designer of a Seldonian algorithm must provide a strategy to allow the user to specify their definition of undesirable behavior, g. For Seldonian algorithms that assume simple definitions of safety, this interface might be straightforward, but for Seldonian algorithms designed to enforce more complex definitions, the interface may require significant effort to implement. Finally, for Seldonian algorithms that offer users significant flexibility when specifying g, the algorithm must account for the possibility that the user's definition is unsatisfiable. In Algorithm 1, this is resolved by allowing the algorithm to return `NO_SOLUTION_FOUND`. However, for some applications, it may not be acceptable for the algorithm to return `NO_SOLUTION_FOUND`. Therefore, in these applications, it may be advantageous to use algorithms specifically designed to enforce specific, application-specific safety definitions instead of using general-purpose Seldonian algorithms.

Despite the considerations listed above, we use the Seldonian framework, and in particular algorithms designed according to Algorithm 1, as the foundation of our contributions for several reasons. First, since we seek to present safe machine learning algorithms for use in practical applications, we find that the advantages that Seldonian algorithms offer users outweigh the additional burden place on the algorithm developer. Specifically, while these algorithms are in many ways more challenging to design than safety-agnostic machine learning algorithms or those designed to avoid specific definitions of undesirable behavior, we find that they are considerably more practical for safety-critical applications, especially for users that are not experts in machine learning. In addition, Seldonian algorithms designed according to Algorithm 1 can be easily extended, which make them a desirable starting point for the contributions we propose in this dissertation. For example, to design Seldonian algorithms that provide high-confidence safety guarantees for general definitions of undesirable behavior, one can simply modify the strategy for computing high-confidence upper bounds on $g(\theta)$ without modifying other steps of the algorithm. Consequently, by modifying specific components of the Seldonian prototype in Algorithm 1, we are able to propose Seldonian algorithms that offer various benefits such as robustness to distribution shift, without having to redesign other components of the algorithm.

1.2 Related Work

To provide context for our proposed contributions, we briefly discuss related work.

First, we discuss alternative formulations that have been proposed to ensure that machine learning algorithms produce models that are safe. In particular, we discuss various ways that safety has been defined in prior literature, ranging from formulations designed to ensure the physical safety of automated systems, as well as formulations based on avoiding undesirable behavior in more general settings such as social applications. Throughout this section, we will highlight the ways in which these formulations differ from the framework used by Seldonian machine learning algorithms.

Next, we will describe potential strategies that might be used to design algorithms that achieve the safety guarantees provided by Seldonian algorithms. Here, we will focus our discussion on general algorithmic approaches instead of specific implementations. For example, we will describe approaches based on enforcing hard constraints, chance constraints, multi-objective optimization, and others, and discuss whether or not these designs are suitable for establishing the guarantees provided by Seldonian machine learning algorithms.

After describing existing work on safety in automated systems, we will then discuss existing work that relates to the contributions we propose in this dissertation. Specifically, we will consider existing work on enforcing general definitions of safety and fairness, and comment on how these approaches relate to or inspire the strategies we propose. In addition, we will discuss existing work on ensuring that machine learning models are robust to changes in distribution, both in the setting of general machine learning and in the setting of ensuring that safety guarantees remain valid after deployment.

1.2.1 Perspectives on Safety in Machine Learning

The goal of avoiding undesirable behavior on the part of automated decision makers has been considered for many years, even preceding the widespread adoption of machine learning as a tool to produce them. Historically, the notion of safety has been associated with the ability of an automated decision maker to avoid behaviors that cause damage or harm in a physical sense. These approaches might consider an automated controller that is interacting with some environment, in which the controller is considered safe if it can be shown to avoid dangerous environment states during its operation. For example, throughout early work on designing ship navigation systems, a major priority was to ensure that ship collisions did not occur during transit [34]. To achieve this goal, these systems were based on manually-designed heuristics [35] or expert systems [34] that could be easily evaluated and verified by domain experts. As the complexity of these systems increased, techniques for verifying the safety of these controllers advanced, often leveraging *formal verification* to ensure that safety requirements were met [55]. While the widespread adoption of machine learning algorithms and the use of models with large numbers of parameters have presented many challenges when applying existing safety verification techniques to ensure safety, this early work on safety in automated systems laid the groundwork for several qualitative safety principles that remain relevant [46]. Mohseni et al., [45] outlines three general principles that guide how failure cases might be dealt with for automated systems, based on existing work on autonomous vehicle control [56]:

Inherently Safe Design: Autonomous systems should be designed to be guaranteed to be safe—that is, it is inherently impossible for the system to cause an unsafe outcome to occur.

Safe Fail: Autonomous systems should detect instances in which they are failing or likely to fail, and revert to a safe default behavior. For example, if a self-driving car is close to driving off a road, it should notify the driver and revert control of the vehicle to them.

Safety Margins: The performance of an autonomous system during training or development should be larger than the minimum required to carry out the task by some margin, so that if performance degrades after the system is deployed, minimum performance requirements are likely to still be met.

The Seldonian framework we leverage in this dissertation is general enough to support most of these principles. While inherently safe design is impractical to enforce in most applications of machine learning [45], the Seldonian framework allows the user to place arbitrarily strict requirements on the probability of safe behavior by setting the δ parameter in (1.1) to be close to 0. In addition, the property that existing Seldonian algorithms are allowed to return NO_SOLUTION_FOUND allows failure cases to be delegated to safe default procedures, satisfying the principle of safe fail during training. Finally, existing Seldonian algorithms support safety definitions based on minimum performance requirements, allowing the user to precisely control the margin by which performance can degrade without causing the system to fail.

While these principles can be used to develop safe machine learning algorithms for particular applications, the rapid adoption of machine learning across many different application areas has expanded the range of potential safety requirements and introduced a larger emphasis on *quantitative* definitions of safety. Instead of considering automated decision makers interacting in some physical environment, models trained using machine learning are commonly deployed to solve classification problems, regression problems, unsupervised learning problems, and more. Consequently, the decision of whether or not it is safe to deploy a model trained using machine learning is often based on criteria besides the occurrence of dangerous states. For example, safety might be based on whether or not a model's predictions are correct for some subset of observations, whether or not the model's performance is expected to meet some minimum requirement once deployed.

An important class of applications that illustrate this are social applications, in which safety refers to whether or not a decision maker is fair. For example, consider the task of predicting whether or not to approve a person's application for a loan. To ensure that a predictive model does not unfairly discriminate based on the identified gender of the applicant, the user might want to ensure that the accuracy of the model is similar on average for different genders. While this safety constraint is not as complex as others that might be defined, it presents an important change from safety requirements considered previously, which are based on a single expected value. Instead of a single unknown quantity that must be estimated, definitions such as these might contain many unknown terms, which are combined in general, nonlinear ways. This pattern is particularly evident in existing work on ensuring that models trained using machine learning are fair when making predictions for applications with risk of adverse societal impact. Verma and Rubin [57], for example, provide an overview of twenty definitions of undesirable behavior that have been proposed for fairness-related classification problems. Many of these definitions, such as *group* *fairness* [20, 49, 65], *conditional statistical parity* [12], *predictive parity* [49, 9], *predictive equality* [9, 12], and several others [30, 62, 39, 37], are based on ensuring that various statistics such as model accuracy, false-positive rate, or probability of predicting a positive label, are similar for observations of different subgroups of the population. Consequently, these definitions, like the example cited above, require computing the absolute value of the difference between two conditional expected values. However, many fairness definitions are more complex. The principle of *treatment equality*, for example, states that the probability that the model makes a false positive prediction divided by the probability it makes a false negative prediction should be similar between subgroups of the population [3]. As a result, this definition depends nonlinearly on four conditional expected values. Similarly, the principle of *disparate impact*, particularly the 80% *Rule* assembled by the California Fair Employment Practice Commission in 1971 for enforcing fairness in hiring decisions made by employers, states that the probability of the model predicting a positive label for any subgroup should never be less than 80% as large as the probability of predicting a positive label for any other subgroup [27, 9, 63]. Finally, principles such as *fairness through awareness* require that, given some metric that measures a distance between two individuals and a second metric that measures differences in the model's predictions for those individuals, similar individuals should result in similar distributions over labels given the model [20].

Motivated by such applications, we consider safety in the general sense defined in (1.1), which is flexible enough to capture all of the aforementioned safety requirements, and is able to be incorporated into larger systems in a way that supports the safety principles laid out by early work in safety.

Finally, the adoption of machine learning as a strategy for training models has exposed an important distinction when considering safety. As opposed to the process of manually designing a controller, the process of training a model is usually based on a set of training observations, which are typically considered to be random samples from some underlying distribution. As a result, the output of a machine learning algorithm is itself random, so that it becomes reasonable to assess the safety of a training algorithm, instead of the safety of a particular model it outputs. For example, while traditionally a controller might be considered unsafe if it causes dangerous states to occur, an algorithm for training a controller might be considered unsafe if it has a high likelihood of producing a model that visits unsafe states, given random training data. The perspective of measuring the safety of an algorithm is more general than ensuring the safety of a particular model, since constraints can be defined on the algorithm that cannot be computed given a single output model. Importantly, we note that the constraints that characterize a Seldonian algorithm—that is, the behavioral constraints expressed in (1.1)—are constraints on the machine learning algorithm, and are therefore more general than many of the safety constraints that have been considered in prior work.

1.2.2 Potential Strategies for Achieving Safety

Having outlined existing work on defining safety in the context of machine learning, we now consider potential methods for enforcing these goals. At a high level, the task of designing an algorithm that meets safety requirements with high-confidence can be seen as a particular type of constrained optimization problem. Consequently, the variety of potential strategies reflects the breadth of existing work on constrained optimization.

First, we might consider approaches that are based on enforcing *hard constraints*. At a high level, if it is possible to specify the set of predictive models that satisfy the user's safety requirements, then any algorithm that searches exclusively over this set of models will, with high-confidence, return a safe model. Depending on the choice of model, this search can be performed using various optimization algorithms, such as the simplex algorithm [13]. Unfortunately, since machine learning algorithms are generally trained on finite amounts of data, it is typically infeasible to construct a feasible set of models to search over without requiring significant domain knowledge on the part of the user. In addition, since the feasible sets are defined as constraints on the parameter space of the model, it can be challenging for users to encode general safety constraints in this form.

A reasonable alternative to enforcing hard constraints is to introduce soft constraints on the objective function used to guide the search for a model. For example, if the user seeks an accurate classifier that is safe according to function g that quantifies unsafe behavior, they might select a model by maximizing

$$\texttt{Accuracy}(\theta) - \lambda g(\theta),$$

where $\lambda \geq 0$ is a weight that determines the importance of the safety requirement. This approach has the advantage of being straightforward to implement and flexible enough to handle general definitions of safety, but has several drawbacks that make it ineffective when designing algorithms. First, the parameter λ can have a dramatic effect on the safety of the model that is selected. In particular, λ determines the tradeoff between the original objective and the function g, so that the choice of suitable value varies based on the application, and can be difficult to find in practice [54]. Furthermore, while algorithms based on soft constraints are capable of producing models that are empirically safe, they do not provide the high-confidence guarantees of safety required by many applications. In contrast, algorithms designed according to the Seldonian framework, including those proposed in this dissertation, do not require the user to specify a tradeoff parameter such as λ, and are able to ensure that with high confidence, the model produced by the algorithm will be safe.

A somewhat similar class of algorithms that might be used to design machine learning algorithms with high-confidence safety guarantees are based on *multi-objective methods*. These methods are designed to produce solutions that are in some sense optimal when there are multiple, potentially conflicting, objectives that should be optimized. This makes such methods appropriate for considering safety in machine learning where the user's primary objective, such as achieving an accurate predictive model, is often at odds with the safety constraints imposed by the application. Unfortunately, these methods are not able to provide high-confidence guarantees of safety. In particular, instead of producing a single model, these methods are usually based on computing the *Pareto frontier* of the various objectives, which is the set of candidate solutions with the property that any changes to the solution will cause one or more of the objectives to decrease. As a result, these methods find a set of solutions that represent reasonable tradeoffs between the objectives, but do not guide the user on which one to select. In addition, a solution that lies on the Pareto frontier is not guaranteed with high confidence to be safe. Therefore, to select a solution and establish high-confidence safety guarantees, the user must perform significant additional data analysis. In contrast, existing Seldonian algorithms and those proposed in this dissertation produce a single model (or NO_SOLUTION_FOUND) that is safe with high confidence.

While methods based on hard constraints are challenging to use without significant expert knowledge, and multi-objective methods and those based on soft constraints do directly provide a trained model that is safe with high-confidence, algorithms based on *chance-constrained programming* are more promising. Chance-constrained programs are a class of optimization problems in which an objective function is optimized subject to a set of constraints on the probability that a set of real-valued random variables are below zero. Formally, if $\ell : \Theta \rightarrow \mathcal{R}$ is some loss function to be minimized, then a chance-constrained program is expressed by,

$$\begin{gathered}\arg\min_{\theta\in\Theta} \ell(\theta)\\ s.t.\ \forall i \in \{1, ..., n\}, \quad \Pr(g_i(\theta, W_i) \leq 0) \geq 1{-}\delta_i,\end{gathered}$$

where each g_i is a deterministic real-valued function, each W_i is a random variable, and each $\delta_i \in (0, 1)$. While this formulation is close to that of (1.1), standard chance-constrained programs make several assumptions that distinguish them from the methods used in Seldonian algorithms. Most notably, standard chance-constrained programs assume that the distribution of each W_i is known, while Seldonian algorithms do not make this assumption. Due to this assumption, chance-constrained programs can require significant expert knowledge on the part of the user. For example, if W_i represents a vector of features describing a person, and g_i assesses whether a model θ is fair on average, then the user of an algorithm based on chance-constrained programming must know the exact joint distribution of all user features, which is impractical in many settings. While standard chance-constrained programs are therefore not identical to the problem solved by Seldonian algorithms, there are several variants that are more similar. In particular, *scenario approximation methods* can be applied to solve chance-constrained programs when the distribution of each W_i is unknown, but can be approximated using a finite set of independent samples. Existing Seldonian algorithms can be seen as a particular instance of this problem setting.

1.2.3 Machine Learning with General Safety Definitions

As described in Section 1.2.1, the growing adoption of machine learning algorithms in diverse applications has greatly expanded the range of safety considerations that must be considered. Whereas early approaches for ensuring safety focused on ensuring that automated decision makers did not interact with the environment in unsafe ways, later applications produced an increase in safety requirements based on the performance of predictive models, and eventually led to the need to enforce general, application-specific safety constraints. While these safety requirements are not necessarily mutually exclusive, this development has led to increased interest in producing algorithms capable of enforcing general safety constraints. In this section, we discuss work on enforcing such constraints, and describe how they relate to the methods proposed in this dissertation, and in particular those proposed in Chapter 2.

First, we consider safety definitions based on ensuring that a trained model does not enter into a dangerous or unsafe state when interacting with an environment. A straightforward way to quantify this notion is to define a binary function, h, that acts as an indicator function for whether or not θ produced an unsafe outcome for a given observation, and base the safety definition on the expected value of h. For example, in [54], we considered a problem in which the goal was to train an agent, θ, for controlling an autonomous insulin pump for diabetic patients. While the goal of training the agent was to optimally moderate the patient's blood sugar levels, there was considerable risk of adverse health effects if the agent caused the patient to become hypoglycemic. Consequently, we applied a Seldonian machine learning algorithm to train the agent, with the safety constraint that the trained agent would not, with high-confidence, cause the patient to become hypoglycemic more frequently than some minimum tolerance, τ. To quantify this safety constraint, we let X denote a random variable representing a vector of features describing a patient, and defined $h(\theta, X)$ to be equal to 1 if θ caused the patient to become hypoglycemic, and 0 otherwise. Then, we defined $g(\theta) = \mathbf{E}[h(\theta, X)] - \tau$, so that $g(\theta) \leq 0$ as long as the probability of θ causing hypoglycemia is no more than the tolerance, τ. The approach illustrated by this example is general, and can be used to express constraints based on avoiding dangerous outcomes in many safety-critical applications.

In contrast to detecting the occurrence of unsafe states, many machine learning applications require assurance that the trained model's performance during training will be representative of what will be encountered once the model is deployed. Because these constraints are often designed to ensure that the trained model generalizes from the training data to the deployment environment, there is considerable overlap between these approaches and the techniques for ensuring *model robustness*, which we address in the next section. Nonetheless, safety constraints based on model performance have been extensively studied in prior work. For example, techniques such as augmenting the dataset with synthetic variations of existing samples, injecting noise, and applying multi-task learning, have been proposed as standard methods for ensuring that performance during training will be consistent with what can be expected after the model is deployed [25]. Other approaches seek to detect when the model is being evaluated on a sample that was not typical based on the training data, and alert the user. For example, methods have been proposed to directly estimate the model's certainty in a prediction during classification [16, 31]. In addition, algorithms have been proposed that leverage an *ensemble* of predictors to ensure that deployment performance does not degrade due to overfitting during training [58]. While these approaches seek to ensure that the model's observed performance can be trusted, other methods have proposed strategies to directly quantify the extent to which performance might drop after deployment. Thomas et al. proposed a batch reinforcement learning algorithm that provides high-confidence guarantees that the performance of the trained agent will be at least equal to come user-defined baseline [53]. Consequently, their algorithm is able to produce new reinforcement learning policies that, with high confidence, will achieve improved performance compared to an existing policy. A growing body of work on establishing *generalization bounds* has been successful in quantifying the degree to which performance of a trained model might differ from its performance after deployment [7]. While these results do not typically include explicit algorithms that provide guarantees on performance, it is straightforward to incorporate them into algorithms that can provide such results. Finally, *risk-sensitive* methods have been proposed that mitigate potential losses in performance by optimizing quantities besides expected performance. For example, algorithms based on *conditional value-at-risk (CVaR)* seek to optimize the mean of a lower quantile of the performance distribution, and therefore capture the notion of optimizing the worst-case performance of the model.

To handle potentially complex definitions such as those for fairness applications, existing work has focused on avoiding undesirable behavior based on either particular definitions of safety, or specific classes of definitions. For example, Zafar et al., [63] propose a classifier that simultaneously satisfies the principles of disparate impact and disparate treatment. Agarwal et al. propose a classification algorithm that achieves fairness with respect to safety constraints that are defined as linear functions of expected values or conditional expected values [1], such as demographic parity [19, 6] or equalized odds [30]. While these contributions are effective in many settings, they are not Seldonian and therefore do not provide high-confidence guarantees of safety, and they are not general enough to support many complex definitions such as disparate impact or treatment equality.

1.2.4 Robust Safety Guarantees

A central objective of this dissertation is to design algorithms that provide high-confidence safety guarantees that continue to hold once the trained model is deployed, even if the deployment environment is different from the training environment. In particular, we consider the setting in which the probability distribution over observations might change after the model is trained. The problem of performing machine learning using data from one distribution with the goal of achieving high performance on observations drawn from a different distribution has been studied in existing work. There, it is often referred to as the problem of *distribution shift* or *domain adaptation* [47, 38], and categorized as a sub-problem in the area of *transfer learning* [60]. Nonetheless, the goals and contributions proposed in existing work are somewhat different from those presented in this dissertation, as described below.

Existing work on distribution shift is usually further categorized based on certain assumptions that describe specific details about how the training and deployment distributions differ. To illustrate, let $X \in \mathcal{X}$ be a random variable representing a vector of features, let $Y \in \mathcal{Y}$ be a random variable representing some response that the user would like to predict, and assume that X and Y are drawn from the training distribution. Now, let $X' \in \mathcal{X}$ and $Y' \in \mathcal{Y}$ represent the features and response variables when drawn from the deployment distribution. The most general form of distribution shift simply assumes that there exists some pair, $(x, y) \in \mathcal{X} \times \mathcal{Y}$, such that the joint probability of (x, y) is different before and after deployment:

$$\exists (x, y) \in \mathcal{X} \times \mathcal{Y} \quad \text{s.t.} \quad \Pr(X{=}x, Y{=}y) \neq \Pr(X'{=}x, Y'{=}y).$$

However, in many problem settings, it is reasonable for the user to assume that the distribution may change in particular, known ways after deployment. For example, in an image recognition system, the distribution of pixel data might shift if different hardware is used to capture a scene before and after the model is deployed, but the correct label associated with the image remains unchanged. This setting, which is variously called *covariate shift* or *frequency feature bias* [60, 42], is defined by the following pair of assumptions:

$$\exists x \in \mathcal{X} \quad \text{s.t.} \quad \Pr(X{=}x) \neq \Pr(X'{=}x), \quad \text{and}$$
$$\forall (x,y) \in \mathcal{X} \times \mathcal{Y}, \quad \Pr(Y{=}y|X{=}x) = \Pr(Y'{=}y|X'{=}x).$$

Similarly, in some applications it is reasonable to assume that the probability of encountering any set of features will not change after deployment, but associated responses might differ. For example, when classifying text, the frequency of particular words might be the same before and after deployment, but the intended meaning of those words might change [60]. This type of shift is often called *label shift* or *context feature bias* [60, 42], and is formalized by the following assumptions:

$$\exists y \in \mathcal{Y} \quad \text{s.t.} \quad \Pr(Y{=}y) \neq \Pr(Y'{=}y), \quad \text{and}$$
$$\forall (x,y) \in \mathcal{X} \times \mathcal{Y}, \quad \Pr(X{=}x|Y{=}y) = \Pr(X'{=}x|Y'{=}y).$$

While existing work on distribution shift, covariate shift and label shift are helpful for providing strategies to account for differences between training and deployment environments, they are typically not applicable in the context of ensuring that a learned model will be safe after deployment. In particular, we seek robust safety guarantees based on general definitions of safety, which often depend on random variables besides X and Y. For example, let $S \in \mathcal{S}$ be a random variable representing some quantity that is required to define safety, such as an indicator for some dangerous event or some protected attribute such as an individual's race, and let $S' \in \mathcal{S}$ represent that quantity when observed after deployment. In Chapter 4, we consider the problem of establishing robust safety guarantees under the general distribution shift assumptions given by,

$$\exists (x,y,s) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{S} \quad \text{s.t.} \quad \Pr(X{=}x, Y{=}y, S{=}s) \neq \Pr(X'{=}x, Y'{=}y, S'{=}s).$$

In addition, in Chapter3 we consider a setting that is similar to covariate shift and label shift, with the modification that the variable that causes the distribution shift is not constrained to be X or Y. In particular, let $T \in \mathcal{T}$ be a random variable representing the quantity that might have a different distribution after deployment, and let $T' \in \mathcal{T}$ represent the same quantity after deployment. Drawing on terminology from many social applications, we refer to $t \in \mathcal{T}$ as a demographic attribute and propose methods to establish safety guarantees that hold under *demographic shift*, which is formalized by the following assumptions:

$$\exists t \in \mathcal{T} \quad \text{s.t.} \quad \Pr(T{=}t) \neq \Pr(T'{=}t), \quad \text{and}$$
$$\forall (x,y,t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{T}, \quad \Pr(X{=}x, Y{=}y|T{=}t) = \Pr(X'{=}x, Y'{=}y|T'{=}t).$$

While many methods proposed in existing work cannot be directly applied to the general distribution shift and demographic shift problems we address in this dissertation, they are helpful for providing strategies that could be extended to these settings. In the remainder of this section, we discuss some of these approaches and describe how they might be adapted to provide safety guarantees for general safety definitions in the settings described above.

First, many algorithms proposed in existing work are based on the idea of learning a representation of predictive features that allow them to generalize well from the training to the deployment environment. For example, strategies have been proposed for either explicitly or implicitly regularizing the optimization used to train models to ensure that they are robust to variations in the distribution of data. Methods such as augmenting training data with synthetically-generated variations or antagonistic examples have been shown to improve the generalization of trained models [25]. Techniques based on explicitly regularizing the training objective using unlabeled data from the deployment distribution have also been applied effectively [64]. Unfortunately, these approaches are difficult to adapt to the setting addressed in this dissertation because proving high-confidence bounds on the safety of a regularized model is challenging.

A third class of methods we consider leverages various amounts of information about the deployment environment during training to improve performance after training. One approach in this class assumes that the change in distribution before and after deployment can be addressed by finding an appropriate transformation of either the features or response variables. For example, under the assumption of covariate shift, these methods seek a transformation, $t : \mathcal{X} \rightarrow \mathcal{X}$ such that $\Pr(X{=}x) \approx \Pr(X'{=}t(x))$. Consequently, performance in the deployment environment can be improved by training on samples from the pre-deployment environment after transforming them using t. Such methods often learn this transformation using limited access to data from the deployment distribution and some assumptions on the form of the transformation, such as assuming that it is linear [23, 24, 26]. Other approaches avoid assuming that the transformation has a particular form by iteratively assigning predicted labels to unlabeled data from the deployment environment [5]. While approaches such as these can be shown to empirically improve performance on the deployment distribution, they are difficult to apply to our problem setting for two reasons. First, they are based on improving performance of the trained model after deployment, and it is not clear how to translate these methods to apply to general, potentially nonlinear definitions of safety. Secondly, the effects of the transformations used by these methods are often challenging to analyze theoretically and rely on assumptions that are difficult to validate, so that it is difficult to produce high-confidence bounds on quantities that depend on the transformed observations.

Nonetheless, some strategies for transforming input samples to account for distribution shift, particularly those based on reweighting observations when estimating expected values, are closely related to our proposed strategies. Intuitively, these approaches account for the impact of distribution shift by reweighting the contribution of each training observation according to the relative probability of encountering that observation before and after deployment. Under general distribution shift, methods based on this strategy have been shown to be effective at improving performance after deployment in many problems, such as classification [61, 33], density estimation [17], and regression [33]. Although these approaches are similar to those mentioned above in that they seek to improve performance on the deployed distribution, in Chapter 3 we show that the same strategy of reweighting training samples can be used to compute high-confidence intervals describing possible values of various safety-related quantities. For example, Lipton et al. proposed a strategy for adjusting classifiers to correct for covariate shift which is in many ways similar to the robust

Seldonian algorithms we propose in Chapter 3 [42]. In particular, they use a reweighting scheme to compute intervals on the confusion rates of a classifier, and use these intervals to determine when to apply a correction step to account for distribution shift. In contrast, we apply a reweighting scheme to compute high-confidence intervals describing general definitions of safety, and use these intervals to decide whether or not to return a trained model.

1.3 Problem Statement and Outline

The overall goal of designing safe machine learning algorithms is to solve the same tasks as those that are solvable with conventional ML approaches while avoiding adverse consequences, with as few qualitative and quantitative differences as possible from those conventional approaches. Thus, the initial Seldonian algorithms that were proposed were not wholly sufficient, and there are many open questions that remain regarding how to design Seldonian algorithms that can achieve the goal of ensuring the safety of models used in real world, potentially high-risk applications. In this dissertation, we will identify two avenues for improving Seldonian algorithm design to make the resulting algorithms more useful in practical settings.

First, we address the limiting assumption that g is defined as an expected value. We will demonstrate that, even after employing several computational tricks, this assumption is simply too limiting to allow the growing collection of real-world fairness and safety requirements to be enforced in existing Seldonian algorithms. To address this limitation, we will propose a mathematical formulation for g that is significantly more expressive, as well as an algorithm for computing valid, high-probability upper bounds on $g(\theta)$. Ultimately, the formulation and algorithm we propose will allow users to provide safety definitions as text, with few limitations, and will support all of the computational steps required to enforce those definitions in Seldonian algorithms.

Next, we will address a major limitation of existing Seldonian algorithms, which is that they assume that the data used for training and for enforcing safety comes from the exact data distribution that will be encountered once the trained model is deployed. This assumption is not new to the design of machine learning algorithms: it is well understood that the observed performance of a trained model may not match its deployed performance due to various factors, such as overfitting and differences in the distribution of data between these settings. Qualitatively, it is often assumed that the training-time and deployment-time performance will differ, with larger differences being observed the more different the training and deployment environments are. However, while it might be permissible to have performance change due to train-deployment differences, this presents a particular problem for Seldonian algorithms. In short, any safety guarantees proven to hold on the training distribution do not necessarily apply to the model after deployment if the training and deployment distributions are not identical. Thus, while a small shift between training and deployment might cause a small change in performance, the safety guarantees provided by existing Seldonian algorithms are extremely brittle: any change in data distribution causes the guarantees provided during training to be useless for ensuring the safety of the model once it is deployed. Because of the severity of this issue, we will propose two strategies for achieving safety guarantees that are robust to shift between the training and deployment settings, which differ based on assumptions about how the user is able to describe the possible shift that might be observed.

To summarize, we will propose the following contributions:

- A mathematical formulation, set of algorithms, and implementation that allow a user to define safety using text input representing an equation for g, and that accomplish all computational requirements to be used in Seldonian algorithms.

- An extension of this mathematical formulation for g, as well as algorithms that allow the user to define safety constraints that are robust to changes between the training and deployment environments, provided that change can be described by a change in the marginal probability distribution of some demographic variable, T.

- Algorithms that allow the user to define safety constraints that are robust to changes between the training and deployment environments, provided that change can be described by constraints on the *Kullback-Leibler divergence* between the probability distributions during training and deployment.

1.4 Outline

In the following three chapters, we will discuss our contributions in detail. Within each chapter, we will begin with a high-level introduction and motivation for our contribution, followed by a discussion of any relevant background material. Then, we will propose concrete steps to implement our contributions. Finally, we will evaluate our contributions by describing several hypotheses for each solution, proposing experiments to test them, and including results and discussion.

CHAPTER 2

CONTRIBUTION: A FLEXIBLE INTERFACE FOR DEFINING SAFETY

In this chapter, we will propose a solution that overcomes the limitation that existing Seldonian algorithms are not adaptable enough to enforce many real-world definitions of safety. Note that while there are many strategies that might be used to design machine learning algorithms that are safe, we use the Seldonian framework as the basis of our contributions in this chapter for several reasons. First, as described in Chapter 1, algorithms designed according to the Seldonian machine learning framework offer several desirable advantages for practical applications. In particular, they do not require extensive data analysis to define safety constraints, and they provide the user with high-confidence safety guarantees, which make these algorithms suitable for safety-critical applications. In addition, prior Seldonian algorithms, such as those designed according to the prototype described in Algorithm 1, serve as a useful starting point for the contributions in this chapter. Specifically, these algorithms are modular, in the sense that the component of the algorithms responsible for training models is separate from the component that is used to establish high-confidence safety guarantees. Consequently, to extend these methods to provide high-confidence safety guarantees based on complex definitions of safety, we simply modify the component of the algorithm that computes high-confidence upper bounds on the prevalence of unsafe behavior when using a particular model

As described in Chapter 1, under the Seldonian framework for machine learning, and algorithm, a, is considered safe if it satisfies the following behavioral constraint,

$$\Pr\big(g\big(a(D)\big) \leq 0\big) \geq 1{-}\delta,$$

where g quantifies the prevalence of unsafe outcomes when using the model's output for the user's application, and $\delta \in [0,1]$ is a user-specified tolerance. However, prior Seldonian algorithms that were proposed made the simplifying assumption that g is defined by the user based on a single expected value that could be estimated without bias using the data available to the algorithm during training. This assumption, although limiting, was made to simplify the process of computing the upper bound on $g(\theta)$ required by the safety test that formed the basis of those initial algorithms..

To make this assumption concrete, let Z be a random variable representing a single observation. For example, in the standard classification environment, each Z might be a feature vector and label pair, whereas in the offline contextual bandit setting each Z might be a tuple containing a context, action, and an observed reward. By using Z instead of application specific variables, we ensure that the formulation we provide in this chapter is general enough to apply to most machine learning problem settings. With Z defined above, we can now formally state the assumption that initial Seldonian algorithms were based on, namely that g is of the form,

$$g(\theta) \coloneqq \mathbf{E}\left[H \,|\, C\right] - \tau, \tag{2.1}$$

where $H = h(Z, \theta)$ defines some *observable* based on the data and model, $C = c(Z, \theta)$ is a Boolean-valued *condition*, and τ is an optional tolerance parameter used to calibrate g.

We will explain several limitations of this assumption below, but first, we briefly illustrate some of the types of safety definitions that *can* be expressed as an instance of (2.1). To provide a concrete setting for these examples, consider the binary classification task of predicting whether or not to approve a loan application, Y, given a feature vector describing the applicant, X, and additional information about the applicant's sex, S. In particular, S might be used during training to assess whether the model is fair, but is often not used for prediction. For example, S could represent a protected attribute such as race or sex which is not used for prediction in applications such as resume filtering. In this setting, Z consists of a single (X, Y, S) tuple, and (2.1) is suitable for ensuring that, for example, certain confusion rates of the model are small. For example, to ensure that the false-positive rate of θ for applicants that identify as female is below a tolerance, τ, one might set $h(Z, \theta) = \theta(X)$ and $c(Z, \theta)$ to be the indicator event for $Y = 1$ and $S{=}\texttt{female}$. Similar constructions can be used to enforce that other confusion rates, such as positive rates, false negative rates, and others, are bounded.

Furthermore, by relying on certain computational tricks, other, more complex definitions can be enforced as well. For example, if the user wishes to enforce that the false-positive rates for individuals identifying as males is similar to the false-positive rate for individuals who identify as female, one can specify a pair of safety constraints, each of the form given in (2.1), that together capture this definition. However, this is only possible because the difference between two false-positive rates is a linear operation on the individual positive-rate terms. Furthermore, strategies for computing a high-confidence upper bound for this definition of g have drawbacks. For example, a suitable bound can be obtained by pairing up random male samples with random female samples, but this process severely reduces the data efficiency of the algorithm. Thus, in this augmented classification problem, we see that (2.1) can be used to enforce tolerances on certain error or confusion rates, and with suitable tricks, can be used to enforce more complex constraints provided they are linear functions of these rates.

Unfortunately, it is straightforward to find a large variety of safety definitions that practitioners would like to use in practice, which cannot be represented by (2.1). Returning to the above classification problem, consider, for example, safety constraints based on *disparate impact*, a principle for assessing unfair behavior that is codified in United States labor laws [27, 9, 63]. In the context of classification, the principle of disparate impact states that a model is unfair with respect to a protected attribute—sex, for example—if the model's false-positive rate for one value of the attribute is less than 80% of that for any other value of the attribute. Assuming binary sex, this can be represented as a safety definition of the Seldonian framework by defining,

$$g_{\texttt{DI}}(\theta) = 0.8 - \min\left\{\frac{\mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{female}]}{\mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{male}]}, \frac{\mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{male}]}{\mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{female}]}\right\}.$$

Because this definition includes ratios, it is clear from inspection that it cannot be represented by any choice of H and C that is consistent with (2.1) and can be sampled without bias. Furthermore, there are many other definitions of safety that similarly cannot be represented by (2.1), and this collection continues to grow as research on quantifying safety—particularly in the realm of fairness for social applications—progresses [57]. Thus, if Seldonian algorithms are to be effective tools for enforcing these notions of safety, we propose that they must support these types of general, nonlinear definitions, and do so without requiring significant machine learning knowledge on the part of the user, and without significantly degrading the data efficiency of the training process.

To this end, in this chapter, we propose a new formulation for g to replace (2.1), which has the desirable properties that it is significantly more expressive but supports the computational requirements to be used in Seldonian algorithms. To ensure that this formulation is straightforward to use, we propose a method that uses parsing algorithms to allow safety constraints to be specified using text inputs to the algorithm, which closely resemble the mathematical notation defining g. In this dissertation, we refer to the collection of a mathematical formulation for g, a procedure that allows the user to specify their desired definition, and an algorithm for computing upper bounds, as an *interface for defining safety*. In our experiments, we show that the interface proposed in this chapter makes significant advances towards the goal of providing Seldonian algorithms that are useful in practical settings, and which enhance the state of the art by allowing a wide range of safety constraints to be enforced.

In the rest of this chapter, we will first discuss existing work that relates to our proposed interface. Then, we will describe our interface in detail. This discussion will consist of three parts: 1) a discussion of our proposed mathematical formulation and an illustration of how it supports more sophisticated definitions than (2.1), 2) a proposal for how to parse text input into a computationally convenient representation for g, and 3) an algorithm for using this representation to compute the high-confidence upper bounds required to perform the safety test used by existing Seldonian algorithms. Finally, we include an evaluation that demonstrates the ability of our interface to produce valid, high-confidence upper bounds on $g(\theta)$ for several definitions.

2.1 Overview of our Proposed Interface

In this section, we discuss the details of the interface for defining safety, which supports enforcing complex, practical definitions of safety within the context of existing Seldonian algorithm design, as exemplified by Algorithm 1. As stated in Chapter 1, an algorithm a is *Seldonian* if it satisfies the following constraints on the probability that it returns an unsafe solution:

$$\Pr\big(g\big(a(D)\big) \leq 0\big) \geq 1{-}\delta,$$

where g quantifies the user's definition of undesirable behavior and δ is their tolerance for how often a can return solutions that are unsafe. Consequently, our interface consists of three parts, namely a mathematical formulation for defining g, an algorithm for parsing this definition from text input, and an algorithm for using the parsed representation to compute high-confidence upper bounds that can be incorporated into Seldonian algorithms designed according to Algorithm 1. We discuss each of these components below.

2.1.1 A Mathematical Formulation for g

To support general, nonlinear definitions of safety, our interface assumes that g can be written in the following form:

$$g(\theta) \coloneqq f(\phi_1(\theta), ..., \phi_k(\theta)), \tag{2.2}$$

where $f : \mathbb{R}^k \to \mathbb{R}$ is a potentially nonlinear and non-smooth function defined by the user, and each $\phi_j(\theta)$ denotes a *parameter* of the data given the model, θ. In particular, we assume each parameter is defined by $\phi_j(\theta) \coloneqq \mathbf{E}[h_j(Z,\theta)|c_j(Z,\theta)]$, where each h_j is a functional that describes an *observable* of Z given the model and each $c_j : \mathcal{Z} \times \Theta \to \{\texttt{True}, \texttt{False}\}$ is some event.

It is straightforward to show that (2.2) is at least as general as (2.1), since the formulation in (2.1) can be recovered by setting $f(v_1) = v_1 - \tau$ and $\phi_1(\theta) = \mathbf{E}[H|C]$. However, since the expression defined by f is allowed to be nonlinear, it follows that (2.2) can support complex definitions of safety that cannot easily be written in the form of (2.1), such as disparate impact. Concretely, if one defines,

$$f(v_1, v_2) = 0.8 - \min\left\{\frac{v_1}{v_2}, \frac{v_2}{v_1}\right\},$$

and defines two parameters, ϕ_1 and ϕ_2 by,

$$\begin{aligned} \phi_1(\theta) &= \mathbf{E}\big[h_1(Z,\theta)\,\big|\,c_1(Z,\theta)\big], \quad \text{with} \\ h_1(Z,\theta) &= \theta(X), \quad \text{and} \\ c_1(Z,\theta) &= \mathbb{I}[Y{=}0 \wedge S{=}\texttt{female}], \end{aligned}$$

and

$$\begin{aligned} \phi_2(\theta) &= \mathbf{E}\big[h_2(Z,\theta)\,\big|\,c_2(Z,\theta)\big], \quad \text{with} \\ h_2(Z,\theta) &= \theta(X), \\ c_2(Z,\theta) &= \mathbb{I}[Y{=}0 \wedge S{=}\texttt{male}], \end{aligned}$$

where $\mathbb{I}[c]$ is 1 if $c = \texttt{True}$ and 0 otherwise, then simple substitution into (2.2) reproduces the definition of disparate impact.

$$
\begin{array}{rcl}
\boldsymbol{expr} & := & \mathit{term} \\
 & | & \mathit{expr} + \mathit{term} \\
 & | & \mathit{expr} - \mathit{term} \\
\boldsymbol{term} & := & \mathit{unary} \\
 & | & \mathit{term} * \mathit{unary} \\
 & | & \mathit{term} \,/\, \mathit{unary} \\
\boldsymbol{primary} & := & \mathit{exp\_value} \\
 & | & \texttt{number} \\
 & | & (\,\mathit{expr}\,)
\end{array}
\qquad
\begin{array}{rcl}
\boldsymbol{unary} & := & \mathit{primary} \\
 & | & |\,\mathit{expr}\,| \\
 & | & +\,\mathit{unary} \\
 & | & -\,\mathit{unary} \\
 & | & \max(\,\mathit{exprs}\,) \\
 & | & \min(\,\mathit{exprs}\,) \\
\boldsymbol{exprs} & := & \mathit{expr} \\
 & | & \mathit{expr}\,,\,\mathit{exprs} \\
\boldsymbol{exp\_value} & := & \texttt{string}
\end{array}
$$

Figure 2.1. Production rules for the expression grammar used to parse input strings into a tree-structured representation with leaf nodes representing either numerical values or strings that will eventually be interpreted as parameters.

2.2 An Algorithm for Parsing Safety Definitions

To allow safety definitions to be provided by the user as text input, we define a pair of grammars appropriate for writing mathematical expressions of expected values, which will be interpreted as the expression defining g using existing implementations for LR parsers.

In particular, our *expression grammar* supports the composition of the unary operations of negation and absolute value, the binary operations of addition, subtraction, multiplication, division, and the variadic operations of taking the maximum or minimum of a set of terms. The terminal symbols of this grammar are constant numerical values, and strings representing conditional expected values corresponding to the various parameters, $\phi_j(\theta)$. The production rules of the expression grammar are used to construct a tree structure representing the mathematical expression, f, that combines them. The full list of production rules for the expression grammar used to parse the representation for f are given in Figure 2.1. Given this grammar, we use a standard LR parser to construct a parse tree representing f, with leaf nodes given by either numerical constants or strings representing each parameter, $\phi_j(\theta)$.

While the expression grammar allows the definition of f to be parsed into a tree-structured representation, it does not parse the individual conditional expected values representing the various

Algorithm 3 ParseDefinition($safety\_str$)

```
$expression\_tree \leftarrow$ LRParse($safety\_str, expression\_grammar$)    {Fig. 2.1}
for each leaf node, $n$, in $expression\_tree$ do
    if $n$ represents an expected value then
        $n.subtree \leftarrow$ LRParse($n.text, parameter\_grammar$)    {Fig. 2.2}
    else
        Set $n.operation$ and $n.operands$ based on the text for $n$.
    end if
end for
return $expression\_tree$
```

parameters. To parse the parameters, we define a separate *parameter grammar* that is used to parse the text representing each parameter into a suitable representation. In the parameter grammar, the terminal symbols are either numerical constants or strings that are constrained to correspond to named variables that are appropriate to the problem setting. For example, in classification, the named variables might be set to include Y to denote labels during classification, X to represent the features, and Y_p to represent the model's predicted labels. The production rules for the parameter grammar are similar to those in the expression grammar, with the addition of logical operations that are necessary to define the Boolean event for any conditional expectation. The full list of production rules for the parameter grammar are shown in Figure 2.2. As with the expression grammar, we propose to parse the parameters into appropriate data structures using a standard LR parser.

Given the expression and parameter grammars, as well as the user's text input defining g, our algorithm for parsing the user's input begins by parsing f using an LR parser based on the expression grammar, producing a tree structure with nodes that capture operations, and leaf nodes that are either numerical constants or strings representing parameters. Next, we separately parse each leaf node that represents a parameter using an LR parser based on the parameter grammar. If parsing errors occur in either of these operations, or if any parameter references variables that are not in the list of pre-defined named variables, these errors are reported to the user. Pseudocode for our parsing algorithm is given in Algorithm 3. In the next section, we present an algorithm

$$\begin{array}{rcl}
\boldsymbol{exp\_value} & := & \mathrm{E}[\ expr\ ] \\
 & | & \mathrm{E}[\ expr \mid log\_expr\ ] \\
 & | & \mathrm{E}[\ log\_expr\ ] \\
 & | & \mathrm{E}[\ log\_expr \mid log\_expr\ ] \\
\boldsymbol{expr} & := & term \\
 & | & expr\ +\ term \\
 & | & expr\ -\ term \\
\boldsymbol{term} & := & unary \\
 & | & term\ *\ unary \\
 & | & term\ /\ unary \\
\boldsymbol{unary} & := & primary \\
 & | & |\ expr\ | \\
 & | & +\, unary \\
 & | & -\, unary
\end{array}$$

$$\begin{array}{rcl}
\boldsymbol{primary} & := & \texttt{string} \\
 & | & \texttt{number} \\
 & | & (\ expr\ ) \\
\boldsymbol{log\_expr} & := & log\_term \\
 & | & log\_expr \,||\, log\_term \\
\boldsymbol{log\_term} & := & log\_unary \\
 & | & log\_unary\ ,\ log\_term \\
\boldsymbol{log\_unary} & := & comp \\
 & | & \sim log\_unary \\
 & | & (\, log\_expr\,) \\
\boldsymbol{comp} & := & expr\ \leq\ expr \\
 & | & expr\ =\ expr \\
 & | & expr \geq expr
\end{array}$$

Figure 2.2. Production rules for the parameter grammar used to parse input strings representing parameters into tree-structured data structures that support evaluation and computation of confidence intervals. Note that when defining expected values of logical expressions (bool_expr), logical values are interpreted as floating point values according to `true` → 1.0 and `false` → 0.0 for syntactic convenience. In addition, if A and B are two logical expressions, the text "A, B" is interpreted as the logical conjunction of A and B to make the grammar's syntax more similar to the standard notation for defining expected values, such as "E[X|A,B]".

that leverages the tree structures for f and for each parameter to construct high-confidence upper bounds on $g(\theta)$ using available data.

2.3 Bounding $g(\theta)$ Using the Parsed Computation Tree

In order to use the available data to compute high-confidence upper bounds on the parsed definition for g, we introduce an algorithm with two parts. First, the available data and the objects representing the parameters are used to construct high-probability confidence intervals for the value of each parameter, $\phi_j(\theta)$. In particular, this step uses the parsed representation of each parameter to generate a set of independent and identically distributed samples of the conditional expected values, and then applies standard concentration inequalities. The confidence intervals describing the set of parameters are constructed so that they hold jointly with probability at least $1{-}\delta$, where δ is defined by the user. Then, using *interval arithmetic* [14], the intervals on each parameter are recursively propagated through the parse tree representing f, to compute a $1{-}\delta$-probability confidence interval for $g(\theta)$.

To make this process concrete, we first assume a set of n, i.i.d. samples of Z are available, denoted $D = \{Z_i\}_{i=1}^n$, which forms the input data for the algorithm. To compute a high-confidence upper bound on $g(\theta)$, we first produce a set of samples that can be used to provide unbiased estimates of each parameter. For parameters j, given by $\phi_j(\theta) := \mathbf{E}[h_j(Z,\theta)|c_j(Z,\theta)]$, we compute a set of samples $H_i^{(j)} := h_j(Z_i,\theta)$ and corresponding values of the condition, $C_i^{(j)} := c_j(Z_i,\theta)$. These samples are computed using D and the object representing ϕ_j, which is built using the parameter parser described in the previous section. Specifically, since each $H_i^{(j)}$ and corresponding condition $C_i^{(j)}$ might be defined as expressions, the parse trees for each term are used to recursively compute them. For example, if $D = \{(X_i, Y_i)\}_{i=1}^n$ and $\phi_j(\theta) = \mathbf{E}[Y - \theta(X)|Y > 0]$, then the set of predictions, $\{\theta(X_i)\}_{i=1}^n$, are subtracted from the corresponding set of true responses, $\{Y_i\}_{i=1}^n$, to produce the set $\{H_i^{(j)} := Y_i - \theta(X_i)\}_{i=1}^n$. Similarly, the set of true responses, $\{Y_i\}_{i=1}^n$ are each compared to the constant 0 to compute the set of conditions, $\{C_i^{(j)}\}_{i=1}^n$. The full set of rules used to combine sets of samples is given in Figure 2.3. Finally, we note that if at any point, a node of the parse trees representing the parameter attempt to combine sets of samples that have incompatible sizes, the user is alerted.

Once the sets $\{H_i^{(j)}\}_{i=1}^n$ and $\{C_i^{(j)}\}_{i=1}^n$ have been constructed, we define the set of samples used to bound $\phi_j(\theta)$ to be the subset of $\{H_i^{(j)}\}_{i=1}^n$ for which the condition evaluates to `true`:

$$S_j(D) \coloneqq \left\{ H_i^{(j)} : C_i^{(j)} = \texttt{true}, \quad \text{for } i = 1, ..., n \right\}.$$

This process results in a set of unbiased estimates of $\phi_j(\theta)$. Using these samples, we bound $\phi_j(\theta)$ using any appropriate confidence interval, such as Hoeffding's inequality or inversion of the Student's t-test as done in prior Seldonian algorithms [54, 44].

Once confidence intervals have been computed for each parameter, they are recursively propagated through the nodes of the parse tree representing f. In particular, for any node representing a mathematical operation, an interval on the value of that node's subtree is be computed using interval arithmetic. Specifically, interval arithmetic provides a set of rules for computing intervals on the output of a mathematical operation, given intervals for the operands [14]. As a simple example, if $x \in [a, b]$ and $y \in [c, d]$, it follows that $x + y \in [a + c, b + d]$. The rules for other operations can become more complex, but are straightforward to derive. The full set of interval arithmetic rules uses in our interface is provided in Figure 2.4. By applying these rules recursively at each node of the computation tree for f, our proposed algorithm outputs a confidence interval that contains the value of $g(\theta)$ with high probability. Psuedocode for our procedures for bounding each parameter and recursively combining them to produce a high-confidence upper bound on $g(\theta)$ is provided in Algorithms 4 and 5.

Importantly, the interval on $g(\theta)$ is valid as long as the each parameter is actually within its corresponding confidence interval. Therefore, the high-confidence upper bound on $g(\theta)$ holds with

$$
\begin{array}{rcl}
-\boldsymbol{\mathcal{X}} & \rightarrow & \{-x_j\}_{j=1}^m \\
|\boldsymbol{\mathcal{X}}| & \rightarrow & \{|x_j|\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j + y_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j - y_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j * y_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} / \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j / y_j\}_{j=1}^m
\end{array}
\qquad
\begin{array}{rcl}
\sim \boldsymbol{\mathcal{X}} & \rightarrow & \{\sim x_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} \wedge \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j \wedge y_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} \vee \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j \vee y_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} \leq \boldsymbol{c} & \rightarrow & \{x_j \leq c\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} \leq \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j \leq y_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} = \boldsymbol{c} & \rightarrow & \{x_j = c\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j = y_j\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} \geq \boldsymbol{c} & \rightarrow & \{x_j \geq c\}_{j=1}^m \\
\boldsymbol{\mathcal{X}} \geq \boldsymbol{\mathcal{Y}} & \rightarrow & \{x_j \geq y_i\}_{j=1}^m
\end{array}
$$

Figure 2.3. Rules for combining sets of samples when generating unbiased estimates of parameters. All variables in calligraphic notation are assumed to be sets of m samples, and c denotes any constant value. Rules in the leftmost column accept real-valued sample sets and output real-valued sample sets, while rules in the rightmost column accept either real-valued or Boolean-valued rules, depending on the operation, and produce Boolean-valued sample sets. The production rules of the parameter grammar ensure that the condition term defining any conditional expectations is Boolean-valued.

at least the probability that all of the confidence intervals on the parameters are simultaneously valid. If the confidence interval for each $\phi_j(\theta)$ is computed to hold with probability $1{-}\delta_j$, the union bound states that the confidence intervals on the set of parameters holds with probability at least $1{-}\delta_1{-}...{-}\delta_k$. Therefore, the bound on $g(\theta)$ can be made to hold with probability at least $1{-}\delta$ as long as the confidence interval on each parameter is computed to hold with probability at least $1{-}\delta/k$.

2.4 Seldonian Machine Learning Algorithms using the New Interface

As shown in Algorithm 1, given a definition g defined according to (2.1), existing Seldonian algorithms proceed by first training a candidate model, θ_c. Next, they construct a high-confidence upper bound on $g(\theta_c)$. If the value of the high-confidence upper bound on $g(\theta_c)$ is below zero with the required probability, then it follows that θ_c is likely to be safe, and the candidate is returned.

$$
\begin{array}{rcl}
\boldsymbol{-v_1} & \rightarrow & [-b_1, -a_1] \\
\boldsymbol{|v_1|} & \rightarrow & \begin{cases} [a_1, b_1] & \text{if } \; a_1 \geq 0 \\ \big[0, \max\{-a_1, b_1\}\big] & \text{if } \; 0 \in [a_1, b_1] \\ \big[-b_1, -a_1\big] & \text{if } \; b_1 \geq 0 \end{cases} \\
\boldsymbol{v_1 + v_2} & \rightarrow & [a_1{+}a_2, b_1{+}b_2] \\
\boldsymbol{v_1 - v_2} & \rightarrow & [a_1{-}b_2, b_1{-}a_2] \\
\boldsymbol{v_1 * v_2} & \rightarrow & \text{range}\{a_1b_1, a_1b_2, b_1a_2, b_1b_2\} \\
\boldsymbol{v_1 / v_2} & \rightarrow & \begin{cases} [-\infty, \infty] & \text{if } \; 0 \in [a_2, b_2] \\ \text{range}\{a_1/b_1, a_1/b_2, b_1/a_2, b_1/b_2\} & \text{else} \end{cases} \\
\boldsymbol{\min\{v_1, ..., v_k\}} & \rightarrow & \big[\min\{a_1, ..., a_k\}, \min\{b_1, ..., b_k\}\big] \\
\boldsymbol{\max\{v_1, ..., v_k\}} & \rightarrow & \big[\max\{a_1, ..., a_k\}, \max\{b_1, ..., b_k\}\big]
\end{array}
$$

Figure 2.4. Rules for combining intervals on values. In these rules, we assume that each value v_j is bounded within some interval, $[a_j, b_j]$. In addition, we assume that $\text{range}\{...\}$ returns an interval with endpoints given by the minimum and maximum values of its operands—that is, $\text{range}\, v_1, ..., v_k \coloneqq [\min\{v1, ..., v_k\}, \max\{v_1, ..., v_k\}]$.

However, if the value of the high-confidence upper bound is above zero, then θ_c may not be safe with the required confidence, and the algorithm instead returns NO_SOLUTION_FOUND.

To produce Seldonian algorithms that leverage the interface proposed in this chapter, we simply replace the procedure used to compute a high-confidence upper bound on $g(\theta_c)$ using the algorithms provided above. In particular, pseudocode for Seldonian algorithms that use our interface is given in Algorithm 6.

2.5 Results and Evaluation

In this section, we seek to assess whether or not the proposed interface is effective at producing high-confidence upper bounds on the prevalence of unsafe behavior.

First, we might consider whether or not the interface is able to produce high-confidence bounds on $g(\theta)$ that are tighter or otherwise preferable to those produced by existing methods. However, direct comparison to existing methods is difficult, because those strategies, including those used in

Algorithm 4 BoundParameter(n, $data$, $bound\_type$, δ)

```
$\mathcal{H} \leftarrow$ apply sample set rules to $n.expression$ given variable sample sets in $data$.   {Fig. 2.3.}
if $n.condition$ exists then
    $\mathcal{C} \leftarrow$ apply sample set rules to $n.condition$ given variable sample sets in $data$.   {Fig. 2.3.}
    $\mathcal{H} \leftarrow \left\{\mathcal{H}_j \in \mathcal{H} : \mathcal{C}_j = \texttt{true} \quad \forall j \in 1, ..., m\right\}$.
end if
$m \leftarrow |\mathcal{H}|$
if $bound\_type$ is Hoeffding then
    Assume each $h \in \mathcal{H}$ is in $[a, b]$
    $\tilde{\mu} \leftarrow \frac{1}{m}\sum_{j=1}^{m}\mathcal{H}_j$
    return $\left[\tilde{\mu} - (b - a)\sqrt{\frac{\log 2/\delta}{2m}}, \quad \tilde{\mu} + (b - a)\sqrt{\frac{\log 2/\delta}{2m}}\right]$
else if $bound\_type$ is t-test then
    $\tilde{\mu} \leftarrow \frac{1}{m}\sum_{j=1}^{m}\mathcal{H}_j$
    $\tilde{\sigma} \leftarrow \sqrt{\frac{\left(\sum_{j=1}^{m}\mathcal{H}_j - \tilde{\mu}\right)^2}{m-1}}$
    return $\left[\tilde{\mu} - \frac{t_{1-\delta/2, m-1}\tilde{\sigma}}{\sqrt{m-1}}, \quad \tilde{\mu} + \frac{t_{1-\delta/2, m-1}\tilde{\sigma}}{\sqrt{m-1}}\right]$
end if
```

Algorithm 5 FlexibleHighConfidenceUpperBound($expression\_tree$, δ, $data$)

```
$k \leftarrow$ number of parameters in $expression\_tree$.
for Each node $n$ in $expression\_tree$ that represents a parameter do
    $parameter\_intervals[n] \leftarrow$ Boundparameter($n, data, bound\_type, \delta/k$)
end for
$[a, b] \leftarrow$ Apply interval arithmetic rules to $expression\_tree$ given $parameter\_intervals$ {Fig. 2.4.}
return b
```

prior Seldonian algorithms, are not flexible enough to produce bounds for the more complex definitions of g that the proposed interface supports. Furthermore, when g is defined to be compatible with the assumptions made by previous methods, the proposed interface is identical to existing methods. Therefore, it is not informative to compare the results of our interface to those of existing methods for those definitions.

However, it is informative to ask whether or not the high-confidence upper bounds produced by our interface are empirically valid—that is, do they return values that, with the required probability, are larger than the true value of $g(\theta)$. To answer this, we designed a set of experiments to

Algorithm 6 FlexibleSeldonian(D, $safety\_str$, δ)

```
$D_c, D_s \leftarrow$ Partition($D$)
$expression\_tree \rightarrow$ ParseDefinition($safety\_str$)
{Select a candidate}
$\theta_c \leftarrow$ TrainCandidate($D_c, expression\_tree, \delta$)
{Perform the safety test using $\theta_c$}
$u \leftarrow$ FlexibleHighConfidenceUpperBound($expression\_tree, \delta, D_s$)
if $u \leq 0$ then
    return $\theta_c$
else
    return NO_SOLUTION_FOUND
end if
```

determine, for various definitions of safety, whether our proposed algorithms successfully return high-confidence upper bounds on the true value of $g(\theta)$, for a model θ. In the following sections, we first explain the details of our experimental design, and then describe our procedure for reporting results. Finally we provide the results of our experiments and discuss them.

2.5.1 Experimental Design

To assess whether or not our proposed interface produces valid high-confidence upper bounds on the prevalence of unsafe behavior, it is necessary to design an experiment where 1) we are able to generate many datasets sampled from the same underlying distribution in order to assess the probability that the bounds fail, and 2) the true value of $g(\theta)$ can be computed exactly. If, for example, a single input dataset is used to evaluate the high-confidence upper bounds, such results would not provide enough information to assess the distribution of values that the high-confidence upper bounds take, or to compute $g(\theta)$ and determine whether or not the high-confidence upper bounds are valid with the required probability. Therefore, we instead designed a series of experiments based on uniformly sampling datasets from a fixed underlying population with replacement. Using this design, it is possible to generate any number of randomly sampled datasets, allowing the distribution of the values of the high-confidence upper bound to be evaluated. In addition, because the population and the sampling distributions are known exactly, it is possible to use this oracle knowledge to compute $g(\theta)$ exactly.

First, we defined the population, denoted $\mathcal{D}_{\text{pop}}$, based on a dataset provided by ProPublica [2], which contains various features describing $5{,}278$ individuals who were convicted of crimes, including their sex and race. This dataset was originally used to provide evidence that Correctional Offender Management Profiling for Alternative Sanctions (COMPAS), a system developed by Northpointe to assess risk of criminal recidivism, exhibited unfair bias against offenders from racial minority groups when classifying offenders into high- or low-risk groups [2]. Consequently, while the datasets used to evaluate our bounds are synthetic due to being randomly sampled from this population, the underlying population is representative of real-world data. Given this data, we defined $\mathcal{D}_{\text{pop}} := \{(X_i, Y_i, R_i)\}_{i=1}^{n}$, where $n = 5{,}278$, each $X_i \in \mathcal{X}$ is a vector of features describing an offender, each $Y_i \in \{0, 1\}$ is a binary indicator for whether or not they eventually reoffended, and each $R_i \in \{\texttt{white}, \texttt{non-white}\}$ indicated whether or not the offender's race was Caucasian.

Next, to assess the validity of our bounds, we trained a model, $\theta : \mathcal{X} \rightarrow \{0, 1\}$, to predict whether or not each individual would reoffend. Since the choice of θ does not impact whether or not the high-confidence upper bounds produced by our interface are valid, we trained θ using logistic regression on all samples in the population, $\mathcal{D}_{\texttt{pop}}$.

Finally, since the central goal of our experiments is to assess whether or not the proposed interface produces high-confidence upper bounds on $g(\theta)$ that are valid for real-world definitions of fairness, we selected five definitions of g that could not be easily supported by existing work, and conducted separate experiments for each. In particular, we performed separate experiments using the following definitions of unfair behavior:

Demographic Parity: The principle of demographic parity states that the rate that a model predicts a positive outcome should not vary significantly between different subgroups of a population [20, 6]. In our experiments, we defined these subgroups based on the self-identified sex of each student, and specified the maximum tolerable difference between positive rates to be 0.1. This leads to the following definition of $g(\theta)$:

$$g_{\text{DP}}(\theta) \coloneqq \Big| \, \mathbf{E}[\theta(X)|R{=}\texttt{white}] - \mathbf{E}[\theta(X)|R{=}\texttt{non-white}] \, \Big| - 0.1.$$

Disparate Impact: The principle of disparate impact also measures discrepancy between the positive rates of a model across different subgroups, but does so by examining their relative size [27, 9, 63]. Specifically, the principle states that a model is unfair if its positive rate for one subgroup is less than a certain percentage of the positive rate for any other subgroup. In our experiments, we set the minimum tolerable ratio to be 0.8, corresponding to the widely-used 80%-rule:

$$g_{\text{DI}}(\theta) \coloneqq 0.8 - \min\left\{ \frac{\mathbf{E}[\theta(X)|R{=}\texttt{white}]}{\mathbf{E}[\theta(X)|R{=}\texttt{non-white}]}, \frac{\mathbf{E}[\theta(X)|R{=}\texttt{non-white}]}{\mathbf{E}[\theta(X)|R{=}\texttt{white}]} \right\}.$$

Equal Opportunity: Instead of measuring discrepancy based on positive rates, the principle of equal opportunity states that different subgroups should have similar false-negative rates [30, 9]. This definition is be appropriate when the frequency of a negative true label is expected to be different between subgroups, but the user would like to ensure that the model does not make certain errors more often for one subgroup compared to another. We set the maximum tolerable discrepancy between false-negative rates to be 0.1, leading to the following definition:

$$g_{\text{EOp}}(\theta) \coloneqq \Big| \, \mathbf{E}[\theta(X)|Y{=}0, R{=}\texttt{white}] - \mathbf{E}[\theta(X)|Y{=}0, R{=}\texttt{non-white}] \, \Big| - 0.1.$$

Predictive Equality: Like equal opportunity, predictive equality states that a model should not make certain errors more often for one subgroup than another. However, predictive equality states that discrepancy should be measured using false-positive rates [9, 12]. We set the maximum tolerable difference between false-positive rates to be 0.1, leading to the following definition:

$$g_{\text{PE}}(\theta) := \left| \mathbf{E}[1{-}\theta(X)|Y{=}1, R{=}\texttt{white}] - \mathbf{E}[1{-}\theta(X)|Y{=}1, R{=}\texttt{non-white}] \right| - 0.1.$$

Equalized Odds: The principle of equalized odds can be seen as combining the principles of equal opportunity and predictive equality, and states that discrepancy should be measured using both false-positive and false-negative rates [30]. We set the maximum tolerable discrepancy according to equalized odds to be 0.1, leading to the following definition:

$$\begin{aligned} g_{\text{EOd}}(\theta) := \; & \left| \mathbf{E}[\theta(X)|Y{=}0, R{=}\texttt{white}] - \mathbf{E}[\theta(X)|Y{=}0, R{=}\texttt{non-white}] \right| + \\ & \left| \mathbf{E}[1{-}\theta(X)|Y{=}1, R{=}\texttt{white}] - \mathbf{E}[1{-}\theta(X)|Y{=}1, R{=}\texttt{non-white}] \right| - 0.1. \end{aligned}$$

Given the population, $\mathcal{D}_{\text{pop}}$, a trained model, θ, we conducted an experiment based on the following procedure, for each definition of g defined above. First, to estimate the distribution of values returned by our interface when evaluated using datasets randomly sampled from the population, we repeatedly generated datasets by uniformly sampling $1{,}000$ observations from $\mathcal{D}_{\text{pop}}$ with replacement. For each dataset, we used Algorithm 5 to parse a string representation of g and compute a high-confidence upper bound on $g(\theta)$ given that dataset. In each case, we set $\delta = 0.05$, so that the high-confidence upper bounds would be computed to be valid for at least 95% of all randomly sampled datasets.

After recording the value of the high-confidence upper bound produced by Algorithm 5 for all $1{,}000$ randomly sampled datasets, we then computed the true value of $g(\theta)$ using $\mathcal{D}_{\text{pop}}$ and the oracle knowledge that the randomly sampled datasets were generated by uniformly sampling from $\mathcal{D}_{\text{pop}}$ with replacement.

Then, given our oracle knowledge of the underlying population, and our oracle knowledge that the datasets were generated using random uniform sampling with replacement, we computed the exact, true value of $g(\theta)$ for each definition of g. To determine whether our computed high-confidence upper bounds hold with the required probability, we then compared the value of each bound with the true value of $g(\theta)$ to estimate the frequency with which they were valid.

For each definition of g, this procedure produced $1{,}000$ independent evaluations of the high-confidence upper bounds produced by our interface, as well as the true value of $g(\theta)$. Consequently, to assess whether or not the high-confidence upper bounds held with the required probability, it sufficed to determine whether or not at least 950 of the evaluated high-confidence upper bounds were larger than the value of $g(\theta)$.

2.5.2 Reporting

Results for each experiment are presented using violin plots showing the distribution of the computed high-confidence upper bounds, for each definition of g. For each definition, we show results obtained when the parameters are bounded using Hoeffding's inequality, as well as concentration inequalities based on inversion of the Student's t-test. Each violin plot is separated into two regions: the red region shows the distribution of the upper 95% of values of the high-confidence upper bound, while the green region shows the distribution of the lower 5%. In addition, a horizontal, dashed black line is included within each figure, showing the exact value of $g(\theta)$. Consequently, if the dashed black line crosses any violin plot at or below the green region of that plot, it indicates that the high-confidence upper bound is empirically valid, as computed upper bounds were larger than the true value of $g(\theta)$ for at least 95% of the randomly sampled datasets. However, if the dashed black line crosses through the red region of the violin plots, it indicates that the high-confidence upper bounds did not hold with the required probability.

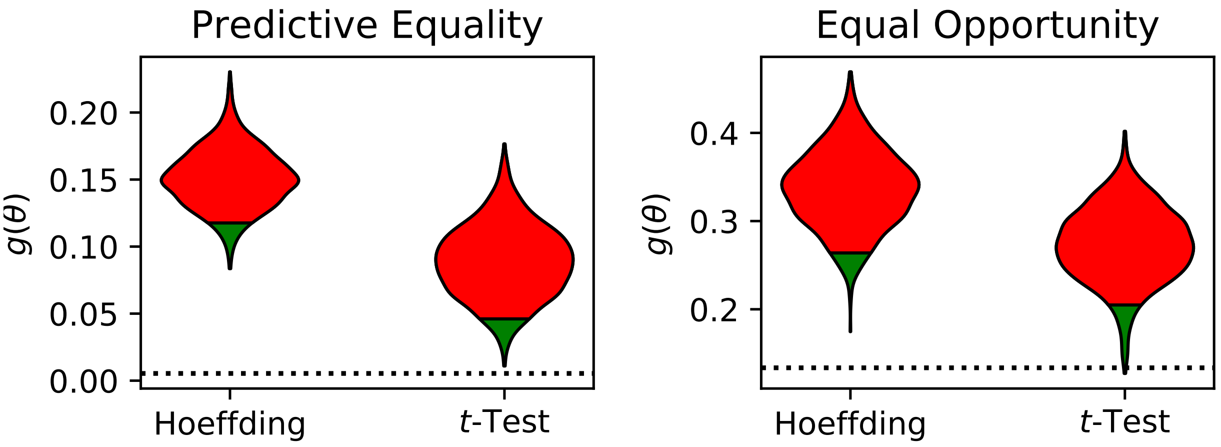

Figure 2.5. Results evaluating the validity of our algorithms for computing high-confidence upper bounds on $g(\theta)$ when g is defined according to the principles of predictive equality and equal opportunity.

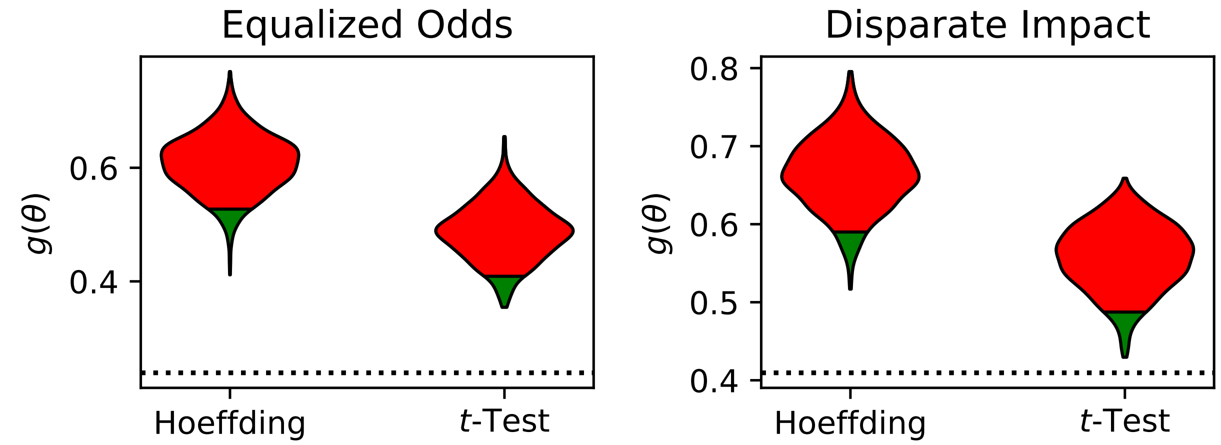

Figure 2.6. Results evaluating the validity of our algorithms for computing high-confidence upper bounds on $g(\theta)$ when g is defined according to the principles of equalized odds and disparate impact.

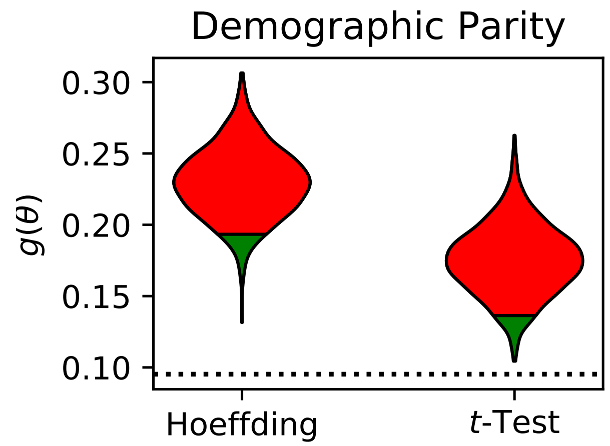

Figure 2.7. Results evaluating the validity of our algorithms for computing high-confidence upper bounds on $g(\theta)$ when g is defined according to the principle of demographic parity.

2.5.3 Results and Discussion

The outcome of our experiments verify our theoretical results, and show that for practical definitions of undesirable behavior, the proposed interface produces high-confidence upper bounds that are valid. In every experiment, at least 95% of the computed values for the high-confidence upper bounds are larger than the true values of $g(\theta)$. Importantly, these high-confidence upper bounds were specified using text input, making them easy to use, and were based on definitions of undesirable behavior that could not be easily represented—or in some cases, could not be represented at all—using existing methods. Nonetheless, there are several patterns in these results that are illustrative to discuss.

First, we note that for every definition of g, the high-confidence upper bounds produced using concentration inequalities obtained by inversion of the Student's t-test are tighter—that is, return values closer to the true value of $g(\theta)$—than those returned using Hoeffding's inequality. This is due to the fact that Hoeffding's inequality, which is based on the worst-case analysis of the variance of a bounded random variable, is generally looser than bounds that use empirical estimates of variance. Secondly, we note that in some cases, such as when g is defined according to the principle of equalized odds, both high-confidence upper bounds are shown to be loose. There are at least two sources of this looseness. First, in some cases, looseness can be introduced due to looseness in the concentration inequalities used. For example, confidence intervals constructed using Hoeffding's inequality are based on worst-case assumptions on the variance of the distribution of a given set of samples, and can therefore be loose when the true variance of the distribution is small. However, a second form of looseness is introduced in the use of interval arithmetic. In particular, when applying interval arithmetic, our algorithms consider the possibility that value of each parameter is independent of the other parameters. When this is not the case, the value of the high-confidence upper bounds become larger than necessary. For example, suppose that with high probability, $\phi_1(X, \theta) \in [0, 1]$ and $\phi_2(X, \theta) \in [0, 1]$, and $g(\theta) = f(v_1, v_2) = v_1 - v_2$. If ϕ_1 and ϕ_2 are dependent and happen to satisfy $\phi_1(X) = -\phi_2(X)$, then it is clear that $g(\theta) = 0$. Without knowledge of this dependence, the high-confidence upper bounds produced by our interface would reason that $g(\theta) \leq 2$ with high probability, which is considerably larger than the true value, $g(\theta) = 0$. Consequently, we note that it may be possible to improve the tightness of the high-confidence upper bounds produced by our interface given additional assumptions on the dependence between each parameter. However, since bounds on the dependence between each parameter can be difficult to determine in practice, we find that despite being conservative, the strategy proposed in this section allows the interface to be useful in general problem settings.

2.6 Limitations and Future Work

The interface proposed in this chapter allows many safety constraints to be enforced that could not be represented by prior safe machine learning algorithms. Nonetheless, it can be improved in several ways. In particular, two directions for future progress on enforcing complex definitions of safety are to expand our proposed interface to enforce definitions that it currently cannot represent, and to improve the tightness of the bounds produced when the parameters defining safety might be correlated.

First, we note that while our proposed interface is sufficient for representing many definitions of safety that have been proposed, it can be made significantly more general. In some cases, this can be achieved by allowing the user to specify parameters that are not single expected values, provided appropriate strategies for computing confidence intervals can be derived. For example, consider a safety definition that involves the conditional value at risk (CVaR) of a real-valued function of an observation, Z. While this parameter can technically be represented in our interface, it might be preferable to treat this parameter as an explicit special case, and apply confidence intervals that are specifically designed to be tight for CVaR [52]. Similarly, the interface could be extended to depend on the entropy of the probability distribution of some function of Z, using confidence intervals on entropy [15]. In addition to extensions based on recognizing specific parameters and applying appropriate confidence intervals, our interface can also be extended by allowing the user to recursively nest certain parameters. For example, if the user would like to define safety based on the variance of a function f of the observation Z, they might define a parameter,

$$\phi(Z,\theta) = \mathbf{E}\left[(f(Z) - \mathbf{E}[f(Z)])^2\right].$$

However, because this parameter is defined as an expected value that itself depends on another expected value, it cannot be represented using the interface proposed in this chapter. Therefore, another direction for improving the proposals of this chapter is to extend the interface to allow parameters, such as expected values, to depend on the value of other parameters.

Finally, the interface we propose in this chapter may produce confidence intervals on $g(\theta)$ that are loose if any of the parameters that define safety are correlated. For example, consider the following definition of safety, where $f : \mathcal{Z} \to \mathcal{R}$:

$$g(\theta) = \phi_1(\theta) + \phi_2(\theta), \quad \text{where}$$
$$\phi_1(\theta) = \mathbf{E}[f(Z)] \quad \text{and} \quad \phi_2(\theta) = \mathbf{E}[-f(Z)].$$

Our proposed interface assumes that $\phi_1(\theta)$ and $\phi_2(\theta)$ are independent, and will therefore produce a non-empty confidence interval on $g(\theta)$. However, it is clear from inspection that $g(\theta) = 0$, since $\phi_2(\theta) = -\phi_1(\theta)$. As shown by this example, if the parameters that define safety are correlated, then the proposed interface is prone to producing confidence intervals on $g(\theta)$ that are looser than necessary. As a result, it would be helpful to detect cases when parameters are correlated, and account for this when computing confidence intervals on $g(\theta)$.

CHAPTER 3

CONTRIBUTION: SELDONIAN ALGORITHMS FOR DEMOGRAPHIC SHIFT

As stated in Chapter 1, existing safe machine learning algorithms make several assumptions that, while reasonable in some cases, are often not satisfied in practice. A notable example is the assumption that once deployed, the model will be evaluated on data that has the same distribution as the data that was collected to train the model in the first place. While this assumption is useful because it allows estimates of the model's performance or safety to be informative about how the model will behave once it is deployed, it is often violated in practice. For example, if a model must decide whether or not to accept a student's college application, it is likely that the distribution of applicants will change between training and deployment, as various trends cause demographics such as race or gender to shift over time. As a result of such shift, safety guarantees that are established during training may no longer apply once the model is deployed.

The example above illustrates a common form of *distribution shift* that is worth addressing directly. In particular, this type of shift occurs when the difference between the training and deployment distributions can be explained entirely by a shift in the marginal distribution of a single random variable, such as race or gender. Drawing on the terminology from social applications, we refer to this variable as a *demographic attribute*, and refer to this specific type of shift between the training and deployment environments as *demographic shift*.

In this chapter, we propose Seldonian algorithms that use information about the distribution of the demographic attribute before and after deployment to provide high-probability safety guarantees that hold after the model is deployed. First, we propose algorithms that achieve this goal provided the exact frequencies of each demographic before and after deployment are known exactly. Building on these, we propose additional algorithms that provide robust safety guarantees when the frequency of the demographic after deployment is unknown, but is known to be in some interval or region.

To illustrate our contributions, we consider an augmented classification problem, in which the task is to predict the label associated with a given feature vector, with additional features provided during training that are used to evaluate safety but are not used for prediction. In particular, in this chapter, we use a concrete problem setting instead of simply referring to data instances abstractly using Z, as in Chapters 2 and 4, to emphasize the difference between random variables that are used for prediction, those used to define safety, and those used to characterize demographic shift. We describe this setting in detail in the following section, but note that our proposed algorithms generalize to other problem settings with only minor modifications.

In the following sections, we present these Seldonian algorithms and provide results showing that the safety guarantees that they provide are useful in practice. While there are many strategies for achieving safe machine learning, we base our contributions on prior Seldonian algorithms for several reasons. First, as described in Chapter 1, Seldonian algorithms provide several advantages over alternative approaches for users that are not experts in machine learning. Since the goal of our dissertation is to provide machine learning algorithms that provide safety assurances in practical settings, we base our contributions on prior Seldonian algorithms in order to leverage these advantages. In addition, prior strategies for designing Seldonian algorithms, such as the strategy described in Algorithm 1, are modular. In particular, to extend these approaches to design algorithms that provide high-confidence guarantees of safety under various forms of distribution shift, we can simply modify the strategy used to compute high-confidence upper bounds on the prevalence of unsafe behavior when using a chosen model, without changing the strategy used to train a candidate model.

In the rest of this chapter, we first provide background needed to define distribution shift formally in the context of classification. Next, we propose Seldonian algorithms that leverage different assumptions about what is known during training about the demographic shift that will be encountered. Finally, we evaluate our algorithms and provide results verifying that the safety guarantees they provide are valid in practice, while prior safe algorithms do not.

3.1 Background

3.1.1 Safety-Augmented Classification

To define a concrete problem setting to illustrate our contributions, we begin with the standard classification problem setting, in which each data *instance* consists of a set of *features* and an associated *label*. However, when considering the safety or fairness of a classifier, it is useful to extend this setting to augment each instance with an additional *safety attribute* that contains information needed to assess the classifier's behavior. This information is often not used for prediction, but is assumed to be available during training to determine whether or not the classifier is safe. For example, depending on applicable laws, information describing the sex or race of a job applicant might be protected, preventing it from being used to make hiring decisions. Nonetheless, this attribute might be required to assess whether or not a predictive model for filtering resumes is biased against applicants of certain races or sexes. We refer to this problem setting as *safety-augmented classification*.

To describe this setting formally, we denote features by $X \in \mathcal{X}$, labels by $Y \in \mathcal{Y}$, and the safety attribute by $S \in \mathcal{S}$, and assume that (X, Y, S) is sampled from some joint probability distribution defined over $\mathcal{X} \times \mathcal{Y} \times \mathcal{S}$.

In the naïve classification setting, the safety attribute is typically ignored, and the user's goal is simply to accurately predict the label associated with X when its true label is unknown. These predictions are generated using a *model*, $\theta : \mathcal{X} \to \mathcal{Y}$. The quality of θ is then measured using a loss function, such as expected classification error. To obtain an accurate classifier, one typically selects a training algorithm, a, designed to minimize the chosen loss, and supplies it with a dataset consisting of n observations sampled independently from the joint distribution—that is, $D = \{(X_i, Y_i, S_i)\}_{i=1}^{n}$, where $\Pr(X_i, Y_i, S_i) \coloneqq \Pr(X, Y, S)$ for all $i \in \{1, ..., n\}$. To assess the safety of an algorithm, we follow the Seldonian framework described in Section 1.1 and assume an auxiliary function, g, that accepts a model and is calibrated so that $g(\theta) > 0$ if θ exhibits unsafe behavior. In contrast to the model's predictions, g typically depends in some way on the safety attribute, S. For example, to assess whether or not a classifier used for resume filtering is biased based on the race of the applicant, g might measure the difference in the model's expected accuracy for individuals of one race compared to another, even if the model's predictions do not explicitly depend on race.

To simplify the discussion of our results in this chapter, we assume that g has the same form as assumed by prior Seldonian algorithms [54]. Specifically, if $H = h(X, Y, S, \theta)$ defines some choice of real-valued observable, $C = c(X, Y, S, \theta)$ defines some Boolean condition, and τ represents a tolerance for unsafe behavior, then in this chapter, we assume that g is defined by,

$$g(\theta) \coloneqq \mathbf{E}\big[\, H \,\big|\, C \,\big] - \tau. \tag{3.1}$$

While this simplifying assumption ignores the contributions proposed in Chapter 2, the results we propose here are easily generalized to take advantage of the flexible interface proposed in that chapter. Specifically, in this chapter, we demonstrate how to compute high-confidence upper bounds on $g(\theta)$ that hold when the probability of encountering certain subgroups of the population changes after the model is deployed. To leverage the new interface, we compute these high-confidence upper bounds, as well as analogous high-confidence lower bounds, for each statistic—that is, each $\phi_j(\theta)$ as defined in Chapter 2—and use the resulting confidence intervals in place of those described in Chapter 2, which do not account for demographic shift. We describe this process more thoroughly in Section 3.3. Finally, while we frame our results in the context of safety-augmented classification problems, our results generalize immediately to other contexts, such as regression problems [54] or and offline bandit problems [44].

3.1.2 Demographic Shift in Safety-Augmented Classification

Given a definition for g, the Seldonian framework states that a training algorithm, a, is safe with respect to g if

$$\Pr\Big(g(a(D)) \leq 0\Big) \geq 1{-}\delta, \tag{3.2}$$

for some confidence threshold, $\delta \in [0, 1]$. Training algorithms designed with safety considerations in mind offer significant advantages for many applications. Nonetheless, even the safety-aware algorithms described in Chapter 1.1 can be of limited practical use because of the problem of *demographic shift*. In short, the apparent performance and safety guarantees associated with these algorithms are only valid after deployment if the trained models are evaluated using data sampled from the same distribution that was sampled for training. However, in many social applications certain subgroups of the population might be over- or under-represented in the training data compared to what will be encountered once the model is deployed. For example, individuals identifying as male or female might be equally represented in the dataset used to train a model, but once the model is deployed, it might be considerably more likely to encounter females compared to males. Consequently, any safety guarantees that were provided by the algorithm may fail when the probability of encountering these subgroups changes between training and deployment.

In order to reason about such differences between the training and deployment data distributions, we associate each observation with a *demographic attribute*, drawing on terminology relevant to many social applications. Specifically, we augment each data instance with a random variable representing the demographic attribute, denoted $T \in \mathcal{T}$. The demographic attribute is often distinct from the other variables defining each observation, but it does not need to be. For example, if the fairness of a classifier is based on the race of each individual, but the distribution over races might change after the model is deployed, then the safety attribute, S, and the demographic attribute, T, may be the same.

Given the demographic attribute, we let (X, Y, S, T) represent an instance observed during training, and let (X', Y', S', T') represent an instance encountered once the model is deployed. To formalize the effect of demographic shift, we assume that the marginal distribution of the demographic attribute may change between training and deployment, but that the pre- and post-shift joint distributions over instances are otherwise identical. Concretely, this can be summarized by the following two conditions, which we refer to as the *demographic shift assumptions*:

$$\exists\, t \in \mathcal{T} \quad \text{s.t.} \quad \Pr(T = t) \neq \Pr(T' = t), \quad \text{and,} \tag{3.3}$$

$$\forall\, (x, y, s, t), \quad \Pr(X{=}x, Y{=}y, S{=}s \,|\, T{=}t) = \Pr(X'{=}x, Y'{=}y, S'{=}s \,|\, T'{=}t). \tag{3.4}$$

Because the terms that define safety in (3.1) depend on X, Y, and S, it follows that any guarantees of safety based on g may fail to hold after the model is deployed, which corresponds to replacing these random variables with X', Y' and S'. Formally, if $H' = h(X', Y', S', \theta)$ and $C' = c(X', Y', S', \theta)$, so that

$$g'(\theta) = \mathbf{E}\big[H' \big| C'\big] - \tau \tag{3.5}$$

measures the prevalence of unsafe behavior when θ is used after deployment, then the challenge presented by demographic shift is summarized by the observation that, for any training algorithm a,

$$\Pr\big(\, g(a(D)) \leq 0 \,\big) \geq 1{-}\delta \quad \not\Rightarrow \quad \Pr\big(\, g'(a(D)) \leq 0 \,\big) \geq 1{-}\delta.$$

The goal of this chapter is to propose training algorithms that satisfy the rightmost inequality—that is, algorithms that offer safety guarantees that are valid after demographic shift—and to provide results that demonstrate that these algorithms are effective in practice.

3.2 Seldonian Algorithms for Demographic Shift

To produce Seldonian algorithms with constraints that hold under demographic shift, we propose modifying the prototypical Seldonian algorithm shown in Algorithm 1 by updating the definition of `HighConfidenceUpperBound` to return the worst-case value of the high-confidence upper bound on $g'(\theta_c)$ that might be encountered, given assumptions describing the demographic shift that might be observed. Intuitively, standard Seldonian algorithms compute a high-confidence upper bound on $g(\theta_c)$ that accounts for uncertainty due to the fact that $g(\theta_c)$ must be estimated using finite data, and uses this high-confidence upper bound to perform a safety test. As stated earlier, this formulation assumes that the training data is representative of what will be encountered after the model is deployed. However, if the user is able to provide specific details characterizing the demographic shift that will be observed after deployment, then it is possible to modify the high-confidence upper bound used in the safety test in order to account for this additional uncertainty.

In particular, if the change in the demographic marginal distribution, $\Pr(T)$, is known exactly, then we propose algorithms that reweight observations during training to compute a high-probability upper bound on the value of $g'(\theta_c)$ if θ_c were deployed. Unfortunately, it is often impractical to assume that, during training, the user is able to exactly specify the demographic shift that will be observed after deployment. If the demographic shift is not known then these high-confidence upper bounds described above cannot be computed exactly. To address this, we also propose algorithms based on performing a worst-case optimization of these high-confidence upper bounds to determine the largest value that might be encountered while satisfying the user's assumptions. By using the resulting value in the safety test described in Algorithm 1, we obtain Seldonian algorithms that provide high-confidence guarantees of safety under demographic shift, even when the future distribution over demographics is not known exactly.

In the following sections, we provide extensions to `HighConfidenceUpperBound` from Algorithm 1 that apply when the safety test is based on Hoeffding's inequality [32] or inversion of the Student's t-test [50]. Then, we describe how these techniques can be combined with the proposals in Chapter 2 to handle more general safety definitions. Finally, we introduce a series of experiments to evaluate the effectiveness of our proposed algorithms, and provide results and discussion.

3.2.1 Robustness to Demographic Shift: Hoeffding-based Bounds

To begin, we consider the case in which there is no demographic shift, which forms the basis for our algorithms. Given n i.i.d. samples $\{(H_i, C_i)\}_{i=1}^{n}$ that each satisfy $\Pr(H_i) = \Pr(h(X, Y, S, \theta))$ and $\Pr(H_i) = \Pr(c(X, Y, S, \theta))$, and assuming that $H \in [a, b]$, Hoeffding's inequality produces the following [48]:

$$\Pr\left(\mathbf{E}[H|C] \leq \frac{1}{n_C}\sum_{i\in\mathcal{I}_C} H_i + (b{-}a)\sqrt{\frac{\log(1/\delta)}{2n_C}}\right) \geq 1{-}\delta,$$

where $\mathcal{I}_C$ are the indices of the samples for which $C_i = \texttt{True}$, and $n_C = |\mathcal{I}_C|$. Since $g(\theta) = \mathbf{E}[H|C] - \tau$, it follows that (3.2) holds if an algorithm a only returns models that satisfy,

$$\frac{1}{n_C}\sum_{i\in\mathcal{I}_C} H_i + (b{-}a)\sqrt{\frac{\log(1/\delta)}{2n_C}} - \tau \leq 0.$$

In the next sections, we consider how to leverage this result to provide similar high-confidence upper bounds that hold under demographic shift.

3.2.1.1 Exactly Known Demographic Shift

To provide safety guarantees that hold under demographic shift, we require $g'(\theta) \leq 0$ with high probability, where $g'(\theta) = \mathbf{E}[H'|C'] - \tau$. However, in this setting, H' and C' are defined with respect to the demographic-shifted distribution, for which no samples are available. Therefore, we seek a new random variable, $\hat{H}$, which can be computed from the pre-shift variables, X, Y and S, but that satisfies $\mathbf{E}[\hat{H}|C] = \mathbf{E}[H'|C']$. Using similar techniques to those used in importance sampling, one can show that under the demographic shift assumptions, the following $\hat{H}$ satisfies these requirements:

$$\hat{H} \coloneqq \phi(T)H,$$

where ϕ computes the importance weight associated with T defined by,

$$\phi(t) \coloneqq \frac{\Pr(C|T{=}t)\Pr(T'{=}t)}{\Pr(T{=}t|C)\sum_{t'\in\mathcal{T}}\Pr(C|T{=}t')\Pr(T'{=}t')} \quad \forall t \in \mathcal{T}. \tag{3.6}$$

This result is provided in Theorem 3.2.1.

Theorem 3.2.1. *Assume that* $\Pr(T{=}t) \geq 0$ *for all* $t \in \mathcal{T}$. *If the demographic shift properties hold, then the random variable* $\hat{H} \coloneqq \phi(T)H$ *satisfies* $\mathbf{E}[H'|C'] = \mathbf{E}[\hat{H}|C]$*, where* ϕ *is defined in* (3.6).

Proof. First, we write $\mathbf{E}[H'|C']$ as a sum over expected values conditioned on the value of the demographic attribute by applying the law of total probability [66]:

$$\begin{aligned}\mathbf{E}[H'|C'] &= \sum_{t\in\mathcal{T}}\mathbf{E}[H'|C',T'{=}t]\Pr(T'{=}t|C') \\ &= \sum_{t\in\mathcal{T}}\mathbf{E}[H|C,T{=}t]\Pr(T'{=}t|C'). && \textit{(Using (3.4))}\end{aligned}$$

Here, the second line follows from the second demographic shift assumption, which states that for all $t \in \mathcal{T}$ and $x, y, s \in \mathcal{X}\times\mathcal{Y}\times\mathcal{S}$, $\Pr(X'{=}x, Y'{=}y, S'{=}s|T'{=}t) = \Pr(X{=}x, Y{=}y, S{=}s|T{=}t)$. Next, we multiply each term by $\Pr(T{=}t|C)/\Pr(T{=}t|C) = 1$, reorganize terms, and write the sum over $t \in \mathcal{T}$ as a single expected value:

$$\begin{aligned}\mathbf{E}[H'|C'] &= \sum_{t\in\mathcal{T}}\mathbf{E}[H|C,T{=}t]\left(\frac{\Pr(T{=}t|C)}{\Pr(T{=}t|C)}\right)\Pr(T'{=}t|C') \\ &= \sum_{t\in\mathcal{T}}\mathbf{E}[H|C,T{=}t]\left(\frac{\Pr(T'{=}t|C')}{\Pr(T{=}t|C)}\right)\Pr(T{=}t|C) \\ &= \mathbf{E}\left[\phi(T)H|C\right],\end{aligned}$$

where,

$$\phi(t) = \frac{\Pr(T'{=}t|C')}{\Pr(T{=}t|C)}.$$

Finally, we rewrite $\phi(t)$ to only depend on the post-shift marginal distribution, $\Pr(T'{=}t)$, and the pre-shift conditional distributions, $\Pr(T{=}t|C)$ and $\Pr(C|T{=}t)$, for each $t \in \mathcal{T}$:

$$\begin{aligned}
\phi(t) &= \frac{\Pr(T'{=}t|C')}{\Pr(T{=}t|C)} \\
&= \frac{\Pr(C'|T'{=}t)\Pr(T'{=}t)}{\Pr(T{=}t|C)\Pr(C')} && \textit{(Using Bayes Theorem)} \\
&= \frac{\Pr(C|T{=}t)\Pr(T'{=}t)}{\Pr(T{=}t|C)\Pr(C')} && \textit{(Using (3.4))} \\
&= \frac{\Pr(C|T{=}t)\Pr(T'{=}t)}{\Pr(T{=}t|C)\sum_{t'\in\mathcal{T}}\Pr(C'|T'{=}t')\Pr(T'{=}t')} \\
&= \frac{\Pr(C|T{=}t)\Pr(T'{=}t)}{\Pr(T{=}t|C)\sum_{t'\in\mathcal{T}}\Pr(C|T{=}t')\Pr(T'{=}t').} && \textit{(Using (3.4))}
\end{aligned}$$

□

The variable $\hat{H} \coloneqq \phi(T)H$ has the advantage that, because it is defined solely with respect to pre-shift random variables, it is possible to generate i.i.d. samples of $\hat{H}$ using data available during training. In particular, let $D = \{(X_i, Y_i, S_i, T_i)\}_{i=1}^{n}$ be a set of i.i.d. input observations. A set of i.i.d. observations for $\hat{H}$ is obtained by computing $\{\hat{H}_i\}_{i\in\mathcal{I}_C}$, where each $\hat{H}_i = \phi(T_i)h(X_i, Y_i, S_i, \theta)$.

Using the set of observations, $\{\hat{H}_i\}_{i\in\mathcal{I}_C}$, we can apply Hoeffding's inequality to derive a high-confidence upper bound on $\mathbf{E}[\hat{H}|C]$:

$$\Pr\left(\mathbf{E}[\hat{H}|C] \leq \frac{1}{n_C}\sum_{i\in\mathcal{I}_C}\hat{H}_i + (b'{-}a')\sqrt{\frac{\log(1/\delta)}{2n_C}}\right) \geq 1{-}\delta.$$

However, since $\mathbf{E}[\hat{H}|C] = \mathbf{E}[H'|C']$ by Theorem 3.4.1, this expression also provides a high-confidence upper bound suitable for assessing safety after demographic shift:

$$\Pr\left(\mathbf{E}[H'|C'] \leq \frac{1}{n_C}\sum_{i\in\mathcal{I}_C}\hat{H}_i + (b'{-}a')\sqrt{\frac{\log(1/\delta)}{2n_C}}\right) \geq 1{-}\delta. \tag{3.7}$$

One caveat in the above is that while $H \in [a, b]$, the range of $\hat{H}$ must take into account the possible values of $\phi(T)$. Specifically, we have

$$a' \coloneqq \inf_{t\in\mathcal{T}} a\,\phi(t) \quad \text{and} \quad b' \coloneqq \sup_{t\in\mathcal{T}} b\,\phi(t),$$

which follows from the observation that $\phi(t) \geq 0$ for all $t \in \mathcal{T}$. Using these results, and recalling that $g'(\theta) \coloneqq \mathbf{E}[H'|C'] - \tau$, it follows $g'(\theta) \leq 0$ with high probability if,

$$\frac{1}{n_C}\sum_{i\in\mathcal{I}_C}\hat{H}_i + \left(\sup_{t\in\mathcal{T}} b\,\phi(t)'{-}\inf_{t\in\mathcal{T}} a\,\phi(t)\right)\sqrt{\frac{\log(1/\delta)}{2n_C}} - \tau \leq 0, \tag{3.8}$$

where each $\hat{H}_i$ depends on θ by the definition, $\hat{H}_i \coloneqq \phi(T_i)h(X_i, Y_i, S_i, \theta)$. From (3.8), it is clear that if the pre-shift conditionals, $\Pr(C|T{=}t)$ and $\Pr(T{=}t|C)$, can be accurately computed for all $t \in \mathcal{T}$, and if the post-shift demographic marginals, $\Pr(T'{=}t)$, are known during training, then $g'(\theta)$ can be upper-bounded even when data from the post-shift distribution is unavailable. Consequently, these quantities can be used to construct a Seldonian algorithm that ensures that its output is safe under the specified demographic shift.

3.2.1.2 Bounded Demographic Shift

Unfortunately, it is often impractical to assume that the post-shift marginal distribution is known exactly during training. To address this, we consider the setting in which the user is able to provide a set of probability distributions over $\mathcal{T}$ that contains the unknown future marginal distribution. For notational convenience, let q^* represent the true (unknown) post-shift marginal distribution over demographics, so that $q^*_t := \Pr(T'{=}t)$. If the user provides a set of probability distributions over the elements of $\mathcal{T}$, denoted by $\mathcal{Q}$, that satisfies $q^* \in \mathcal{Q}$, then a high-confidence upper bound on $g'(\theta)$ can be found by computing the worst case value of the high-confidence upper bound attained for any $q \in \mathcal{Q}$.

To begin, we parameterize the high-confidence upper bound in (3.8) to explicitly depend on a possible post-shift demographic distribution, q:

$$\Pr\left(g'(\theta) \leq U_{\texttt{hoeff}}(g, D, \theta; q^*)\right) \geq 1{-}\delta,$$

where

$$\begin{aligned} U_{\texttt{hoeff}}(g, D, \theta; q) \quad = \quad & \frac{1}{n_C} \sum_{i \in \mathcal{I}_C} \phi(T_i; q) h(X_i, Y_i, S_i, \theta) \\ & + \left(\sup_{t \in \mathcal{T}} b\, \phi(t; q) {-} \inf_{t \in \mathcal{T}} a\, \phi(t; q) \right) \sqrt{\frac{\log(1/\delta)}{2 n_C}} - \tau, \end{aligned}$$

and

$$\phi(t; q) := \frac{\Pr(C|T{=}t) q_t}{\Pr(T{=}t|C) \sum_{t' \in \mathcal{T}} \Pr(C|T{=}t') q_{t'}}.$$

While the true post-shift marginal distribution, q^*, is assumed to be unknown, it is clear that if $q^* \in \mathcal{Q}$, then

$$U_{\texttt{hoeff}}(g, D, \theta; q^*) \leq \sup_{q \in \mathcal{Q}} U_{\texttt{hoeff}}(g, D, \theta; q).$$

Computing the supremum of $U_{\texttt{hoeff}}$ for $q \in \mathcal{Q}$ is complicated by the fact that the value of the high-confidence upper bound depends on the infimum and supremum values of q. Consequently, $U_{\texttt{hoeff}}$ might not be a smooth function of q, and its supremum cannot be found in closed form. Therefore, we propose to use a numerical optimizer to approximate the supremum of U_{hoeff} over $q \in \mathcal{Q}$. In our implementations, for example, we applied *simplicial homology optimization* [21], which is proven to converge to global optima of non-smooth functions subject to equality and inequality constraints such as those defined by the condition, $q \in \mathcal{Q}$.

3.2.2 Robustness to Demographic Shift: Bounds based on the Student's t-Test

The strategy proposed above can be summarized as first parameterizing a function that computes the high-confidence upper bound to explicitly depend on the demographic distribution after deployment, and then performing a worst case analysis over the deployment distributions that might be encountered. While the high-confidence upper bounds proposed above are based on Hoeffding's inequality, the same strategy can also be applied to design high-confidence upper bounds under demographic shift that use other inequalities.

For example, given n i.i.d. samples $\{(H_i, C_i)\}_{i=1}^{n}$, let $\mathcal{I}_C$ be the indices of the samples for which $C_i = \texttt{True}$, and let $n_C = |\mathcal{I}_C|$. By inverting the commonly-used Student's t-Test [50] and applying Bessel's correction for sample standard deviations, one can derive the following high-confidence upper bound on the expected value, $\mathbf{E}[H|C]$:

$$\Pr\left(\mathbf{E}[H|C] \leq \frac{1}{n_C}\sum_{i\in\mathcal{I}_C} H_i + \frac{\tilde{\sigma}}{\sqrt{n_C}} t_{1-\delta, n_C-1}\right) \geq 1-\delta, \tag{3.9}$$

where $\tilde{\sigma}$ is the sample standard deviation of the samples of H_i for which $C_i = \texttt{True}$ including Bessel's correction,

$$\tilde{\sigma} := \sqrt{\frac{1}{n_C-1}\sum_{i\in\mathcal{I}_C}\left(H_i - \frac{1}{n_C}\sum_{j\in\mathcal{I}_C} H_j\right)^2},$$

and $t_{1-\delta,n_C-1}$ is the $1-\delta$ quantile of the Student's t distribution with n_C-1 degrees of freedom. Note that while (3.9) only holds exactly if $\frac{1}{n_C}\sum_{i\in\mathcal{I}_C} H_i$ is a normally distributed random variable, the central limit theorem states that the distribution of this quantity converges to a normal distribution as $n_C \to \infty$ regardless of the distribution of each H_i, making this approximation reasonable for large n_C [59].

In the following sections, we show how this high-confidence upper bound can be modified to derive a high-confidence upper bound on the demographic-shifted expected value, $\mathbf{E}[H'|C']$, when the distribution of the demographic variable is known exactly, and when it is known to be in some set.

3.2.2.1 Exactly Known Demographic Shift

In order to derive a high-confidence upper bound on $\mathbf{E}[H'|C']$, where H' and C' represent the observable and condition after demographic shift has occurred, we again use Theorem 3.2.1. In particular, Theorem 3.2.1 states that the expected value using the demographic-shifted variables, H' and C'—that is, $\mathbf{E}[H'|C']$—is equivalent to the importance-weighted expected value $\mathbf{E}[\phi(T)H|C]$, which can be estimated using data available during training. Recall that the weighting function, ϕ, depends on the distribution of the demographic after demographic shift occurs, as well as several quantities that can be estimated during training:

$$\phi(t) := \frac{\Pr(C'|T'{=}t)}{\Pr(C|T{=}t)} = \frac{\Pr(C|T{=}t)\Pr(T'{=}t)}{\Pr(T{=}t|C)\sum_{t'\in\mathcal{T}}\Pr(C|T{=}t')\Pr(T'{=}t')} \quad \forall t \in \mathcal{T}.$$

To obtain a high-confidence upper bound on $\mathbf{E}[H'|C']$, we leverage Theorem 3.2.1 and apply (3.9) to the reweighted observations, $\{\phi(T_i)H_i\}_{i\in\mathcal{I}_C}$. This results in the following:

$$\Pr\left(\mathbf{E}[H'|C'] \leq \frac{1}{n_C}\sum_{i\in\mathcal{I}_C}\phi(T_i)H_i + \frac{\tilde{\sigma}'}{\sqrt{n_C}}t_{1-\delta,n_C-1}\right) \geq 1{-}\delta, \tag{3.10}$$

where $\tilde{\sigma}'$ is the sample standard deviation of the reweighted samples for which $C_i = \texttt{True}$:

$$\tilde{\sigma}' := \sqrt{\frac{1}{n_C{-}1}\sum_{i\in\mathcal{I}_C}\left(\phi(T_i)H_i - \frac{1}{n_C}\sum_{j\in\mathcal{I}_C}\phi(T_i)H_j\right)^2}. \tag{3.11}$$

Using this result, it is straightforward to design a *quasi*-Seldonian algorithm that provides safety guarantees that hold under demographic shift. If we assume that the user would like to ensure that with high probability, an algorithm produces a safe result using $g(\theta) \coloneqq \mathbf{E}[H|C] - \tau$ and subject to demographic shift, it suffices to define the high-confidence upper bound used in the safety test—that is, `HighConfidenceUpperBound` in Algorithm 1—to compute the value of,

$$U_{t\text{-test}}(g, D_s, \delta) \coloneqq \frac{1}{n_C} \sum_{i \in \mathcal{I}_C} \phi(T_i) H_i + \frac{\tilde{\sigma}}{\sqrt{n_C}} t_{1-\delta, n_C - 1} - \tau, \tag{3.12}$$

where $\tilde{\sigma}'$ is the sample standard deviation of the samples of $\hat{H}_i = \phi(T_i)H_i$ defined in (3.11). The resulting quasi-Seldonian algorithm can be used as long as the post-shift marginal probabilities, $\Pr(T'{=}t)$, are known for each $t \in \mathcal{T}$. Note that we refer to this as a quasi-Seldonian algorithm because the high-confidence upper bound in (3.10) is only valid approximately as $n_C \to \infty$.

3.2.2.2 Bounded Demographic Shift

Next, we consider how to extend the high-confidence upper bound in (3.12) to apply when the demographic-shifted marginal distribution, $\Pr(T'{=}t)$, is not known exactly, but is known to lie in some set of distributions, $\mathcal{Q}$. Specifically, if q^* denotes the true post-shift demographic distribution, defined by $q^*_t \coloneqq \Pr(T'{=}t)$ for each $t \in \mathcal{T}$, then we assume that $q^* \in \mathcal{Q}$. Similar to our strategy in Section 3.2.1.2, we accomplish this by parameterizing $U_{t\text{-test}}$ to explicitly depend on the post-shift demographic distribution, and calculate the worst-case value of the upper bound for all demographic distributions in $\mathcal{Q}$.

First, we write the high-confidence upper bound function in (3.12) to explicitly depend on the choice of post-shift demographic distribution, q:

$$U_{t\text{-test}}(g, D, \delta; q) \coloneqq \frac{1}{n_C} \sum_{i \in \mathcal{I}_C} \phi(T_i; q) H_i + \frac{\tilde{\sigma}'_q}{\sqrt{n_C}} t_{1-\delta, n_C - 1} - \tau,$$

where

$$\tilde{\sigma}'_q = \sqrt{\frac{1}{n_C - 1} \sum_{i \in \mathcal{I}_C} \left(\phi(T_i; q) H_i - \frac{1}{n_C} \sum_{i \in \mathcal{I}_C} \phi(T_i; q) H_i \right)^2},$$

and

$$\phi(t;q) \coloneqq \frac{\Pr(C|T{=}t)q_t}{\Pr(T{=}t|C)\sum_{t'\in\mathcal{T}}\Pr(C|T{=}t')q_{t'}}.$$

Note that if the post-shift demographic distribution is known, then $U_{t\text{-test}}(g,D,\delta;q^*)$ is identical to (3.12) for any input dataset, D. However, if q^* is not known, but is contained in $\mathcal{Q}$, then we compute the largest value of $U_{t\text{-test}}(g,D,\delta;q)$ for all $q\in\mathcal{Q}$, which is necessarily at least as large as $U_{t\text{-test}}(g,D,\delta;q^*)$:

$$U_{t\text{-test}}(g,D,\delta;q^*) \leq \sup_{q\in\mathcal{Q}} U_{t\text{-test}}(g,D,\delta;q).$$

As with the comparable high-confidence bounds based on Hoeffding's inequality, we numerically compute this supremum using simplical homology optimization [21].

3.3 Integrating Robust Bounds into Seldonian algorithms

Having described methods for computing high-confidence upper bounds on $g'(\theta)$, which measures the prevalence of unsafe behavior when θ is deployed under demographic shift, we now discuss how these results can be used for the design of Seldonian machine learning algorithms. Revisiting the pseudocode that describes a prototypical Seldonian algorithm, we propose that Seldonian algorithms that account for demographic shift can be obtained by modifying the function `HighConfidenceUpperBound` to be based on the bounds presented in this chapter.

Furthermore, these can be extended in a straightforward way to leverage the general interface proposed in Chapter 2. Specifically, Chapter 2 proposes an interface that supports computing upper bounds on $g(\theta)$ that are based on multiple conditional expected values. To integrate the strategies proposed in this section, we compute the upper bounds presented here, as well as the corresponding lower bounds, for each statistic. For example, for bounds based on Hoeffding's inequality, it is straightforward to apply the union bound to show that,

$$\Pr\left(\inf_{q\in\mathcal{Q}} L_{\texttt{hoeff}}(g,D,\theta;q) \leq g'(\theta) \leq \sup_{q\in\mathcal{Q}} U_{\texttt{hoeff}}(g,D,\theta;q)\right) \geq 1{-}2\delta,$$

where,

$$
\begin{aligned}
L_{\texttt{hoeff}}(g, D, \theta; q) \;\; = \;\; & \frac{1}{n_C} \sum_{i \in \mathcal{I}_C} \phi(T_i; q) h(X_i, Y_i, S_i, \theta) \\
& - \left(\sup_{t \in \mathcal{T}} b\, \phi(t; q) - \inf_{t \in \mathcal{T}} a\, \phi(t; q) \right) \sqrt{\frac{\log(1/\delta)}{2 n_C}} - \tau.
\end{aligned}
$$

Intervals such as these are computed for each statistic that appears in the definition of g. Then, these intervals are used in place of the intervals described in Chapter 2, which do not account for the impact of demographic shift.

3.4 Evaluation and Results

In Section 3.2, we presented high-confidence upper bounds that can be used to design Seldonian algorithms that provide safety guarantees that hold after demographic shift. However, it is reasonable to ask whether or not the resulting algorithms are truly effective. For example, do the guarantees provided by these algorithms hold empirically, and do they require unreasonable amounts of data? In this section, we describe experiments designed to answer these questions.

First, we outline three specific hypotheses we seek to test in order to evaluate whether or not Seldonian algorithms based on the results of this chapter are effective. Next, we describe a set of experiments designed to test these hypotheses. This includes a discussion of the safety definitions we evaluated, our strategy for selecting assumptions on how the demographic distribution might change for each experiment, and a strategy for simulating the effect of demographic shift and evaluating relevant statistics such as accuracy and the value of $g'(\theta)$ for any model, θ. Finally, we provide results and discussion comparing our proposed robust Seldonian algorithms to comparable standard Seldonian algorithms.

3.4.1 Hypotheses

In our experiments, we seek to answer three questions regarding the behavior of our proposed algorithms for overcoming demographic shift compared to existing approaches.

Robustness of safety guarantees The central question we wish to examine in these experiments is whether or not various algorithms, including our proposed methods, are able to provide high-probability guarantees of safe behavior that continue to hold under demographic shift in practice. We expect existing Seldonian algorithms to behave safely when evaluated on the training distribution, but to violate their safety constraints when evaluated on the demographic-shifted distributions. In contrast, we hypothesize that the robust algorithms we propose in this chapter will behave safely after demographic shift.

Model accuracy We also seek to determine whether or not our robust algorithms will achieve comparable accuracy compared standard Seldonian algorithms. Assessing this condition is complicated by the fact that, since the robust algorithms are solving a more constrained problem than the baselines, it is possible that the models trained using these methods will be unable to simultaneously achieve high accuracy and satisfy safety constraints after demographic shift. However, we use our experiments to produce qualitative evidence demonstrating how large the loss in accuracy might be in practice.

Data efficiency Finally, we seek to determine whether or not our robust algorithms will be capable of producing models—that is, to avoid returning `NO_SOLUTION_FOUND`—with reasonable amounts of training data compared to baselines.

3.4.2 Experimental Design

Evaluating the hypotheses listed above is complicated by the fact that, in order to assess the validity of safety guarantees and to assess the data efficiency of our proposed algorithms, we require multiple pairs of datasets sampled from the same underlying distribution, and which exhibit a known amount of demographic shift. If, for example, a single pair of datasets exhibiting demographic shift were used, it could provide evidence about the accuracy of models produced using these algorithms, but would not allow us to assess whether the safety guarantees hold with the required probability, nor to properly assess the amount of data required for such algorithms to reliably return solutions instead of NO_SOLUTION_FOUND. To overcome this difficulty, we developed an experimental design based on simulating training datasets by resampling from a fixed population. Randomly sampling training datasets from the population using a known distribution and precisely controlling the deployment distribution offers several advantages for evaluating the hypotheses listed above. For example, this design ensures that any failures—that is, instances in which the model produced by our algorithms returns unsafe solutions with a larger frequency than specified—can be properly attributed to a failure of the high-confidence upper bounds we propose, instead of being due to violation of the user's assumptions on demographic shift. In addition, because the underlying population, as well as the training and deployment distributions, are known, it is straightforward to compute exact values for $g(\theta)$ and $g'(\theta)$. Therefore, we decided to base our experiments on uniformly sampling training datasets from the population, training a model using either our proposed algorithms or other baselines, and finally evaluating the safety of that model after demographic shift by antagonistically searching for a new distribution over the population that satisfies the user's assumptions, but otherwise maximizes the prevalence of unsafe behavior.

Following this design, there are three components that are required. First, our experiments require a dataset to be used as the population, and a concrete classification problem statement. In Section 3.4.3, we present the problem statement used in our experiments and provide details on the data we used. Next, we require a set of example inputs that a user might provide to our algorithms. These inputs consist of a safety definition, g, as well as a set of marginal demographic distributions, $\mathcal{Q}$, that the user assumes contains the true demographic distribution that will be observed after deployment. Details for how we generated the example user input for each experiment are provided in Section 3.4.4. Finally, our experiments require a strategy for antagonistically selecting a new

Algorithm 7 DemographicShiftTrial($\mathcal{D}_{\text{pop}}$)

```
Specify example user inputs, $g$, $\delta$, and $\mathcal{Q}$                {Sec. 3.4.4}
Uniformly sample $D$ from $\mathcal{D}_{\text{pop}}$
for each training algorithm, $a$ do
    Train $\theta^*$ using $D$, $g$, $\delta$, and $\mathcal{Q}$
    Record values for accuracy and $g(\theta^*_a)$ during training
    Find a distribution $Q^*$ to maximize $g'(\theta^*_a)$ subject to $q^* \in \mathcal{Q}$   {Sec. 3.4.5}
    Record values for accuracy and $g'(\theta^*_a)$ under $Q^*$
end for
```

distribution over the population that has a marginal demographic distribution q^* that satisfies $q^* \in \mathcal{Q}$ and otherwise maximizes the prevalence of unsafe behavior for a given model. This component of our experiments is described in Section 3.4.5.

Given these components, our experiments were conducted as follows. First, for each experiment, we specify a safety constraint and an assumption on how much the demographic distribution might change after model deployment. Then, we repeat several trails, where for each trial, we first generate a training dataset by resampling uniformly from the population with replacement. Then, we apply each algorithm being evaluated to produce a set of trained models. We evaluate our proposed algorithms and compare to standard Seldonian algorithms. After the models are trained, we antagonistically select a new sampling distribution over the population for each model, denoted Q^*, which is chosen to maximize the prevalence of unsafe behavior—that is, to maximize $g'(\theta)$—while satisfying the assumptions on demographic shift. In particular, note that Q^* denotes a chosen distribution over the population after deployment, whereas q^* is a corresponding distribution over the demographic attribute. Finally, we use oracle knowledge of the antagonistic distributions and the population to compute exact values for post-shift accuracy and safety for each model. Psuedocode for a single trial of our experiments is given in Algorithm 7. In the following sections, we describe our methodology in more detail, and provide specific experimental details.

3.4.3 Problem Statement and Notation

In our experiments, we consider the binary classification task of predicting whether or not a student's grade point average (GPA) is above a certain threshold, while avoiding discriminatory behavior based on sex. Furthermore, we consider the setting in which the marginal distribution of the race of the students might change after deployment. If, for example, a model tends to make discriminatory decisions based on sex more often for a minority race compared to a majority race, then the model might make such decisions much more frequently on average if the distribution of races were to change after deployment.

Formally, we assume a training data observation consists of a tuple, (X, Y, S, T), where X is a vector of predictive features, $Y \in \{0, 1\}$ is a binary label representing whether or not the student's GPA was above 3.0, the demographic attribute, T, represents the race of the student, and S describes the sex of the student. Importantly, T and S are not directly used for prediction, and are not assumed to be available once the model is deployed.

Given this formulation, the central goal of each classification algorithm is to learn a model that achieves small classification error,

$$\ell(\theta) = \mathbf{E}\left[\mathbb{I}[\theta(X) \neq Y]\right],$$

where $\mathbb{I}[c]$ is 1 if c is `true` and 0 otherwise. We evaluate the fairness of each model based on whether or not its predictions are discriminatory with respect to the sex of the student. Finally, we consider the setting in which, after the model is deployed, the demographic distribution—that is, the marginal distribution over student race—might change. In summary, our experiments are designed to determine whether or not various algorithms are able to provide high-confidence guarantees that their output models will not discriminate based on sex, even when the distribution of student races might change after deployment.

For our experiments, we used data describing $43{,}303$ students applying to universities. Each feature vector representing a student consisted of their scores on nine entrance exams, and the goal of each classification algorithm was to predict whether or not the student's GPA was above 3.0. In addition, the dataset included information about the race and sex of each student, which we used as the demographic attribute and safety attribute, respectively. Consequently, this data allowed us to assess our hypothesis about whether or not safety constraints based on the sex of each student would continue to hold if the distribution over race were to change after each model is deployed. In the following sections, we describe our methodology for specifying these safety constraints, as well as our assumptions on how the distribution over student races could change.

3.4.4 Specifying User Inputs

In this section we describe the example user inputs that we applied to generate our experiments. During real-world application of our methods, these inputs would be provided by the user, and the methods are not limited by the choices made below.

First, we discuss how we defined undesirable behavior for each experiment. In practice, the choice of safety definition, g, and failure tolerance, δ, is given by the user, so the definitions used in our experiments were chosen to illustrate that our methods work for a variety of problems, and do not influence whether or not the results of our experiments are valid. Therefore, we selected five definitions of fairness that have been studied in existing work, and which leverage the interface proposed in Chapter 2. Specifically, we performed separate experiments for five definitions of fairness, which are similar to those used in our experiments in Chapter 2. While the principles for each definition are the same as those in Chapter 2, in these experiments, we based the protected subgroups for each definition on the sex of each individual, instead of their race. When conducting these experiments, we combined the results described in this chapter with the interface proposed in Chapter 2, in order to show that our methods are able to provide safety guarantees under demographic shift for practical definitions of safety. Details and motivation for selecting these definitions are provided in Section 2.5.1, but the specific definitions are shown in Figure 3.4.4 for completeness.

To complete the problem specification for each experiment, we provided example assumptions that the user might place on the possible demographic distributions that could be encountered in the deployment environment. Specifically, this step requires defining a set $\mathcal{Q}$ that the user assumes will contain the future distribution over the demographic attribute. Because the choice of $\mathcal{Q}$ is considered an input from the user, our strategy for selecting $\mathcal{Q}$ does not impact the validity of our results. As a result, while the algorithms that we propose work well for any demographic assumptions of the form,

$$\mathcal{Q} := \left\{ q \;\text{ s.t } \forall t \in \mathcal{T},\;\; q_t \in [a_t, b_t],\;\; q_t \geq 0,\;\; \text{and } \sum_{t \in \mathcal{T}} q_t = 1 \right\},$$

we generated $\mathcal{Q}$ in our experiments by inflating a region around the true demographic marginal probabilities when sampling uniformly from the population. Intuitively, this strategy can be seen as interpolating between assuming no demographic shift and assuming completely unknown demographic shift, using an interpolation factor, α.

Formally, if $p_t := \Pr(T{=}t)$ is the true probability of encountering demographic t when sampling uniformly from the population, then we set a_t and b_t by computing,

$$a_t = (1{-}\alpha)p_t \qquad \text{and} \qquad b_t = (1{-}\alpha)p_t + \alpha,$$

where $\alpha \in [0, 1]$. We then defined $\mathcal{Q}$ in each experiment based on (3.4.4). Using this scheme, setting $\alpha = 1$ causes the interval for $\Pr(T'{=}t)$ to become $[0, 1]$, while setting $\alpha = 0$ produces the point interval $[p_t, p_t]$. In our experiments, we set $\alpha{=}0.3$ to simulate the case in which the user has some, but not complete, knowledge about the demographic distribution.

Demographic Parity [20, 6]:

$$g_{\text{DP}}(\theta) \coloneqq \Big|\, \mathbf{E}[\theta(X)|S{=}\texttt{female}] - \mathbf{E}[\theta(X)|S{=}\texttt{male}] \,\Big| - 0.1$$

Disparate Impact [27, 9, 63]:

$$g_{\text{DI}}(\theta) \coloneqq 0.8 - \min\left\{ \frac{\mathbf{E}[\theta(X)|S{=}\texttt{female}]}{\mathbf{E}[\theta(X)|S{=}\texttt{male}]}, \frac{\mathbf{E}[\theta(X)|S{=}\texttt{male}]}{\mathbf{E}[\theta(X)|S{=}\texttt{female}]} \right\}$$

Equal Opportunity [30, 9]:

$$g_{\text{EOp}}(\theta) \coloneqq \Big|\, \mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{female}] - \mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{male}] \,\Big| - 0.1$$

Predictive Equality [9, 12]:

$$g_{\text{PE}}(\theta) \coloneqq \Big|\, \mathbf{E}[1{-}\theta(X)|Y{=}1, S{=}\texttt{female}] - \mathbf{E}[1{-}\theta(X)|Y{=}1, S{=}\texttt{male}] \,\Big| - 0.1$$

Equalized Odds [30]:

$$\begin{aligned} g_{\text{EOd}}(\theta) \coloneqq \quad & \Big|\, \mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{female}] - \mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{male}] \,\Big| + \\ & \Big|\, \mathbf{E}[1{-}\theta(X)|Y{=}1, S{=}\texttt{female}] - \mathbf{E}[1{-}\theta(X)|Y{=}1, S{=}\texttt{male}] \,\Big| - 0.1. \end{aligned}$$

Figure 3.1. Exact definitions of fairness used our experiments on evaluating safety guarantees under demographic shift. Details motivating these definitions can be found in Section 3.4.4. These definitions were selected to evaluate whether the algorithms proposed in this chapter were able to provide safety guarantees under demographic shift for a variety of practical definitions. In our experiments, these definitions were specified as text input, and were parsed and bounded using the interface proposed in Chapter 2.

3.4.5 Simulating and Evaluating the Impact of Demographic Shift

Here, we describe our procedure for simulating the impact of demographic shift given the population described above. Intuitively, after generating a training dataset, we antagonistically select a new, non-uniform distribution over the population, which satisfies the user's assumptions on demographic shift—that is, that the marginal distribution over demographics is contained in $\mathcal{Q}$—but otherwise maximizes the prevalence of unsafe behavior. Since the population and sampling distributions are known during evaluation, this oracle knowledge can be used to compute exact values for various statistics, such as expected classification accuracy and the value of $g'(\theta)$ for any model θ.

To make this procedure formal, let the input dataset—that is, the population—be denoted by $\mathcal{D}_{pop}$:

$$\mathcal{D}_{pop} := \{(x_i, y_i, s_i, t_i)\}_{i=1}^{n}.$$

Note that we do not refer to this set using the standard notation for random variables because in the context of our experiments, the population is treated as a fixed, non-random underlying population. To generate a random training dataset, D, consisting of n_0 samples, we sample observations uniformly from $\mathcal{D}_{pop}$ with replacement. Specifically, if P denotes the uniform distribution over the observations in $\mathcal{D}_{pop}$, then training datasets are defined by $D := \{(X_j, Y_j, S_j, T_j)\}_{j=1}^{n_0}$, where each $(X_j, Y_j, S_j, T_j) \sim P$.

Next, we generate a new distribution over the population that satisfies the user's assumptions but otherwise maximizes the prevalence of unsafe behavior for a given model, which we denote by Q. However, to comply with the user's assumptions on demographic shift, Q must be selected carefully. The following theorem provides the conditions that Q must satisfy to achieve this.

Theorem 3.4.1. *Let P denote a uniform distribution over $\mathcal{D}_{pop}$, a population consisting of n observations, $\mathcal{D}_{pop} := \{(x_i, y_i, s_i, t_i)\}_{i=1}^{n}$. Assume that the demographic attribute takes values in some set $\mathcal{T}$, and that each demographic $t \in \mathcal{T}$ occurs at least once in the population. Next, assume that*

q denotes some distribution over $\mathcal{T}$, so that q_t denotes the probability of encountering the demographic $t \in \mathcal{T}$, and that $q \in \mathcal{Q}$. Finally, let $\mathbb{N}_{\mathcal{D}_{pop}}[x, y, s, t]$ denote the number of observations in $\mathcal{D}_{pop}$ that are equal to (x, y, s, t) and let $\mathbb{N}_{\mathcal{D}_{pop}}[t]$ denote the number of observations that have demographic attribute equal to t. It follows that the definition of Q shown below satisfies both of the demographic shift assumptions, and has a marginal distribution over demographics that is contained in $\mathcal{Q}$:

$$Q(X{=}x, Y{=}y, S{=}s, T{=}t) = \frac{\mathbb{N}_{\mathcal{D}_{pop}}[x, y, s, t]}{\mathbb{N}_{\mathcal{D}_{pop}}[t]} q_t.$$

.

Proof. To show this result, we derive an expression for Q that has these properties by construction.

First, we expand the post-shift joint distribution using the laws of conditional probability:

$$Q(X{=}x, Y{=}y, S{=}s, T{=}t) = Q(X{=}x, Y{=}y, S{=}s | T{=}t) Q(T{=}t).$$

Next, we apply the second demographic shift assumption:

$$Q(X{=}x, Y{=}y, S{=}s, T{=}t) = P(X{=}x, Y{=}y, S{=}s | T{=}t) Q(T{=}t).$$

Then, we represent the conditional $P(X, Y, S | T{=}t)$ as a ratio using laws of conditional probability:

$$Q(X{=}x, Y{=}y, S{=}s, T{=}t) = \frac{P(X{=}x, Y{=}y, S{=}s, T{=}t)}{P(T{=}t)} Q(T{=}t).$$

Because P is a uniform distribution over the observations in $\mathcal{D}_{pop}$, it follows that the value of $P(X{=}x, Y{=}y, S{=}s, T{=}t)$ is simply the number of occurrences of (x, y, s, t) in $\mathcal{P}$ divided by the total number of samples in the population, n. Similarly, $P(T{=}t)$ is simply the number of observations that have demographic attribute equal to t, divided by n. Since we assume that each demographic is observed in the population, it follows that $P(T{=}t) > 0$ for all $t \in \mathcal{T}$. Let $\mathbb{N}_{\mathcal{D}_{pop}}[x, y, s, t]$

denote the number of observations in $\mathcal{D}_{pop}$ that are equal to (x, y, s, t) and let $\mathbb{N}_{\mathcal{D}_{pop}}[t]$ denote the number of observations that have demographic attribute equal to t. It follows that for any observation, $(x, y, s, t) \in \mathcal{D}_{pop}$ we have

$$Q(X{=}x, Y{=}y, S{=}s, T{=}t) = \frac{\mathbb{N}_{\mathcal{D}_{pop}}[x, y, s, t]/n}{\mathbb{N}_{\mathcal{D}_{pop}}[t]/n} Q(T{=}t) = \frac{\mathbb{N}_{\mathcal{D}_{pop}}[x, y, s, t]}{\mathbb{N}_{\mathcal{D}_{pop}}[t]} Q(T{=}t).$$

Finally, we define the marginal distribution of Q over demographics to be given by q:

$$Q(X{=}x, Y{=}y, S{=}s, T{=}t) = \frac{\mathbb{N}_{\mathcal{D}_{pop}}[x, y, s, t]}{\mathbb{N}_{\mathcal{D}_{pop}}[t]} q_t.$$

Since Q has the same conditional distribution given the demographic as P by construction, it satisfies the demographic shift assumptions. Furthermore, since the marginal distribution of Q over demographics is defined to be given by q, which satisfies $q \in \mathcal{Q}$, it also satisfies the user's assumptions on demographic shift. □

Theorem 3.4.1 shows that, given any $q \in \mathcal{Q}$, we can construct a distribution over the population which satisfies the demographic shift assumptions. Therefore, to select antagonistically select a distribution to maximize the prevalence of unsafe behavior for a given model, θ, we numerically optimize $g'(\theta)$ over $q \in \mathcal{Q}$ using simplical homology optimization [21] to determine the maximizing marginal distribution, q^*, and then define the distribution over $\mathcal{D}_{\text{pop}}$ using Theorem 3.4.1.

Theorem 3.4.1 can also be used to compute exact values for various statistics of interest during evaluation, such as expected classification accuracy or the value of $g'(\theta)$ for any model, θ. For example, consider estimating the post-shift classification accuracy of a model, θ, given by $\mathbf{E}_Q[\mathbb{I}[\theta(X){=}Y]]$. If $\bar{\mathcal{D}}_{pop}$ denotes the set of unique observations in $\mathcal{D}_{pop}$, then we have

$$\begin{aligned}\mathbf{E}_Q[\mathbb{I}[\theta(X){=}Y]] &= \sum_{(x,y,s,t)\in\bar{\mathcal{D}}_{pop}} \mathbb{I}[\theta(x){=}y] Q(X{=}x, Y{=}y, S{=}s, T{=}t) \\ &= \sum_{(x,y,s,t)\in\bar{\mathcal{D}}_{pop}} \mathbb{I}[\theta(x){=}y] \frac{\mathbb{N}_{\mathcal{D}_{pop}}[x, y, s, t]}{\mathbb{N}_{\mathcal{D}_{pop}}[t]} Q(T{=}t).\end{aligned}$$

Analogous expressions can be used to compute exact values for post-shift model accuracy, as well as the value of $g'(\theta)$ for any model θ.

3.5 Results

In the previous section, we described a set of experiments designed to evaluate several hypotheses about the effectiveness of our proposed Seldonian algorithms. In this section, we describe the results of these experiments, which verify our claims that the algorithms proposed in this chapter are effective at providing safety guarantees that hold under demographic shift, while existing algorithms are not effective.

In our experiments, we evaluated four training algorithms to test our hypotheses. All four algorithms were configured to produce linear models on the features, X, to ensure that no algorithm had an advantage over the others due to using more complex, nonlinear models to generate predictions. First, we include results for the algorithms proposed in this chapter based on Hoeffding's inequality and based on inversion of the Student's t-test. These are denoted by *Seldonian$_{DS}$* and *quasi-Seldonian$_{DS}$*, and shown in green in our figures. Finally, to assess whether or not our algorithms would be more effective under demographic shift than prior Seldonian algorithms, we include results for two comparable standard Seldonian algorithms. In particular, we selected Seldonian algorithms based on Hoeffding's inequality and based on inversion of the Student's t-test, which are referred to as *Seldonian* and *quasi-Seldonian* in our figures and shown in blue [54].

3.5.1 Evaluation and Reporting

In order to test the hypotheses in Section 3.4.1, we record several values produced for each trial of our experiments.

First, to assess whether or not the safety guarantees provided by our algorithms are empirically valid, we recorded, for each trial, whether or not each algorithm resulted in a *safety failure* during training or after deployment. Specifically, a safety failure occurs when any algorithm produces a model—as opposed to NO_SOLUTION_FOUND—that is unsafe. To detect safety violations during training, we computed the true value of $g(\theta)$, where θ is the model returned by the algorithm, and tested whether or not it was above zero. Then, to detect a safety violation after deployment, we computed $g'(\theta)$ for each model, using oracle knowledge of the population and the antagonistic resampling distribution.

Next, to assess our hypothesis regarding the accuracy of models produced by our algorithms, we computed the true accuracy of each trained model before and after deployment. To compute these values, we used oracle knowledge of the population, as well as the sampling distributions that generated each training and deployment dataset. However, the assessment of whether or not our algorithms are effective at producing accurate models under demographic shift constraints is complicated by the fact that our proposed algorithms may return models that achieve lower accuracy than standard Seldonian algorithms for multiple reasons. First, our algorithms might return models with lower performance because they must account for uncertainty in the deployment distribution, which standard methods ignore. Because this performance loss is due to constraints placed by the user—namely that with high confidence the algorithms should return models that are safe after demographic shift—this performance loss does not reflect a shortcoming of our proposed algorithms. However, it is also possible that accurate models exist that are safe under demographic shift, but our algorithms are unable to identify them.

To determine which forms of performance loss are responsible for any decrease in accuracy exhibited by our proposed algorithms, we also estimate the best possible classification accuracy that can be achieved by a linear model 1) without safety constraints, 2) while satisfying safety constraints defined on the training distribution, and 3) while satisfying safety constraints that must hold under demographic shift. Thus, if the accuracy of our models approach these best-case values, it implies that any loss in accuracy when considering the possibility of demographic shift is due to the strictness of the user's constraints, and not the algorithms we propose. Initially, we performed the search for linear models that achieve best-case accuracy using a brute force search over the set of all linear classifiers. However, we found that, even when randomly sampling a large number of linear models, the highest observed accuracy was significantly lower than the average accuracy of standard Seldonian algorithms, indicating that these algorithms were identifying a very small set of high-performing models that were extremely unlikely to be encountered using random search. Consequently, we approximated our best-case values by initializing a search using a model found using standard Seldonian algorithms, and fine-tuning the model to improve accuracy using *covariance matrix adaptation evolution strategy* (CMA-ES) [29]. As a result, the values we report are only approximations to the best-case values for accuracy, but provide a reasonable lower bound on the performance that might be expected under safety constraints, and under safety constraints that are required to hold under demographic shift.

Finally, we recorded statistics to assess the data efficiency of each algorithm. In particular, we sought to determine whether our algorithms would require prohibitive amounts of data to return safe models instead of `NO_SOLUTION_FOUND`, compared to standard Seldonian algorithms. For this, we recorded whether or not each algorithm returned a solution or `NO_SOLUTION_FOUND` for each trial. In addition, we conducted trials using various amounts of training data in order to assess how the probability of returning a model changes as a function of the amount of data provided.

These quantities were recorded for each trial, producing empirical estimates that could be used to assess the validity of the guarantees provided by our algorithms, as well as the average accuracy and data efficiency of the models they produced for each experiment. However, since these estimates are based on a finite number of trials (25 in our experiments) and therefore do not represent the true values, we also computed standard error for each quantity in order to assess this uncertainty.

Concretely, for each experiment, we report five plots. First, we show the acceptance rate of each algorithm as a function of the size of the training dataset. Then, we provide two rows of plots, featuring the results on the training and deployment distributions, respectively. In each row, we first show the average accuracy of the models produced by each algorithm for various training

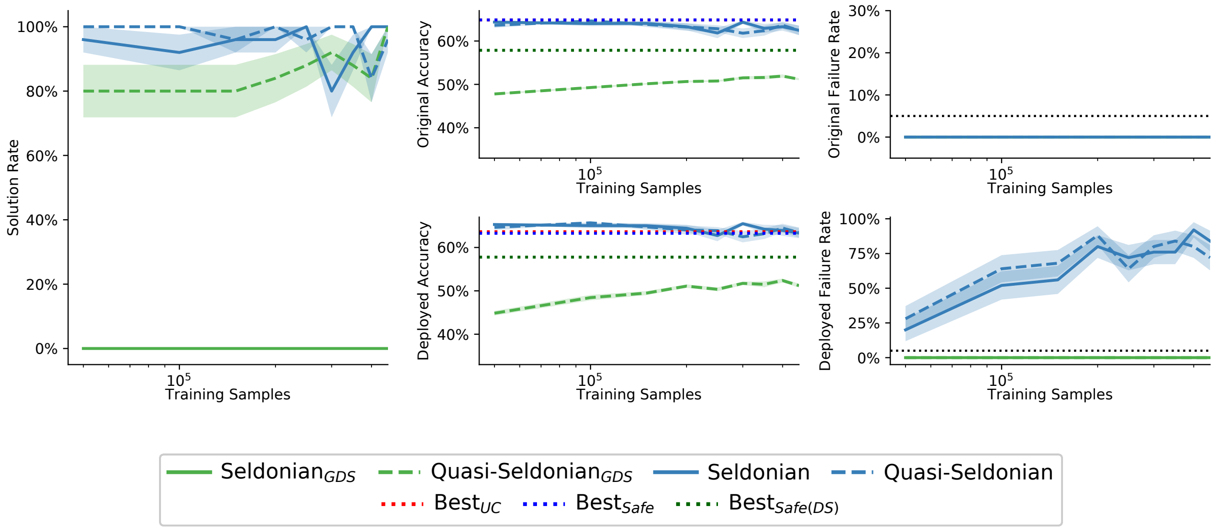

Figure 3.2. Results for experiments enforcing safety constraints based on the principle of disparate impact to preclude discrimination based on student sex when the marginal distribution of student race might change after model deployment. The rightmost column of plots displays the frequency with which each algorithm returns a solution that is unsafe before and after deployment, and demonstrates that the algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after demographic shift, whereas standard Seldonian algorithms (blue) do not. However, empirically, these added safety benefits come at the cost of accuracy (shown in the middle column of plots) and data efficiency (shown in the leftmost plot). Nonetheless, these results show that for safety-critical applications for which ensuring safety after deployment is the primary requirement, our algorithms are effective.

dataset sizes. Next, we show the failure rate for each algorithm as the size of the training dataset increases. In all figures, we show standard error for each statistic using a shaded region around the curve that delineates the average values.

3.5.2 Results

The results of our experiments using each definition of fairness are shown in Figures 3.2 to 3.6.

First, we evaluate our first hypothesis, which stated that when subjected to demographic shift, any safety guarantees provided by baselines would become invalid, while the safety guarantees provided by our proposed algorithms would not. The results of our experiments support this hypothesis. Specifically, when examining the subplots showing the frequency of safety violations—

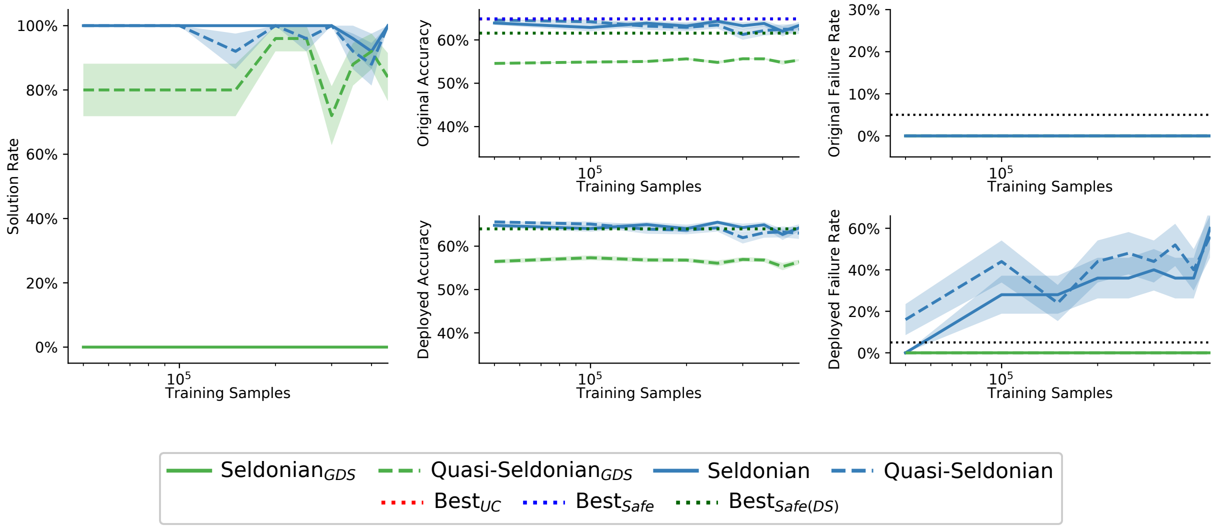

Figure 3.3. Results for experiments enforcing safety constraints based on the principle of demographic parity to preclude discrimination based on student sex when the marginal distribution of student race might change after model deployment. These results demonstrate a similar pattern as shown in Figure 3.2: The algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after deployment, while prior Seldonian algorithms (blue) do not. Interestingly, for this definition, the data efficiency of our quasi-Seldonian algorithm (displayed in the leftmost plot) was comparable to that of standard Seldonian algorithms. These results demonstrate that our proposed algorithms are effective solutions in safety-critical applications that are subject to demographic shift, and demonstrate that these benefits are consistent for a variety of practical safety definitions.

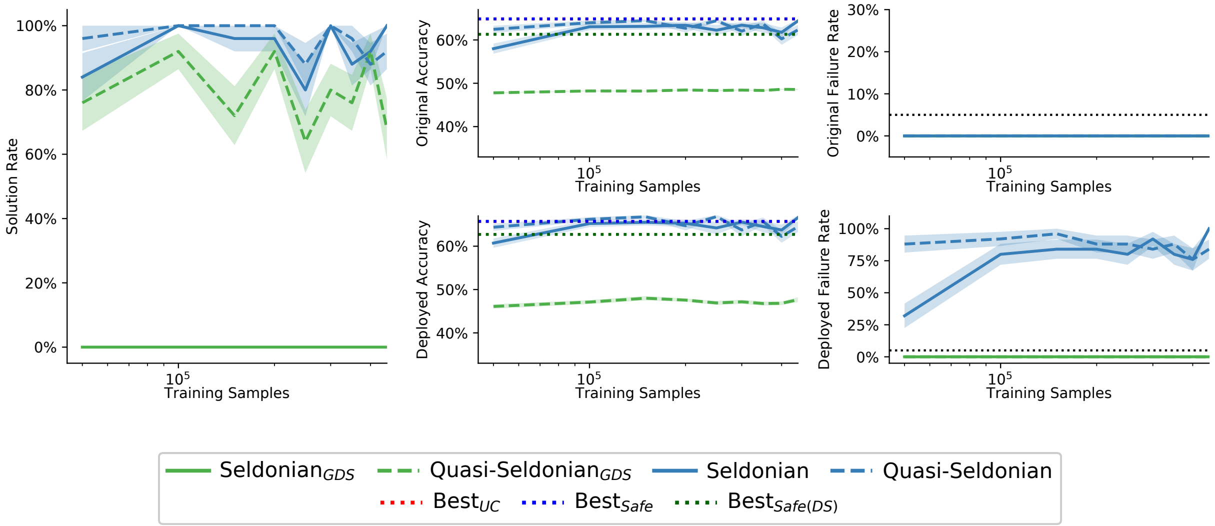

Figure 3.4. Results for experiments enforcing safety constraints based on equal opportunity to preclude discrimination based on student sex when the marginal distribution of student race might change after model deployment. These results demonstrate a similar pattern as shown in Figure 3.2: The algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after deployment, while prior Seldonian algorithms (blue) do not. These results demonstrate that our proposed algorithms are effective solutions in safety-critical applications that are subject to demographic shift, and demonstrate that these benefits are consistent for a variety of practical safety definitions.

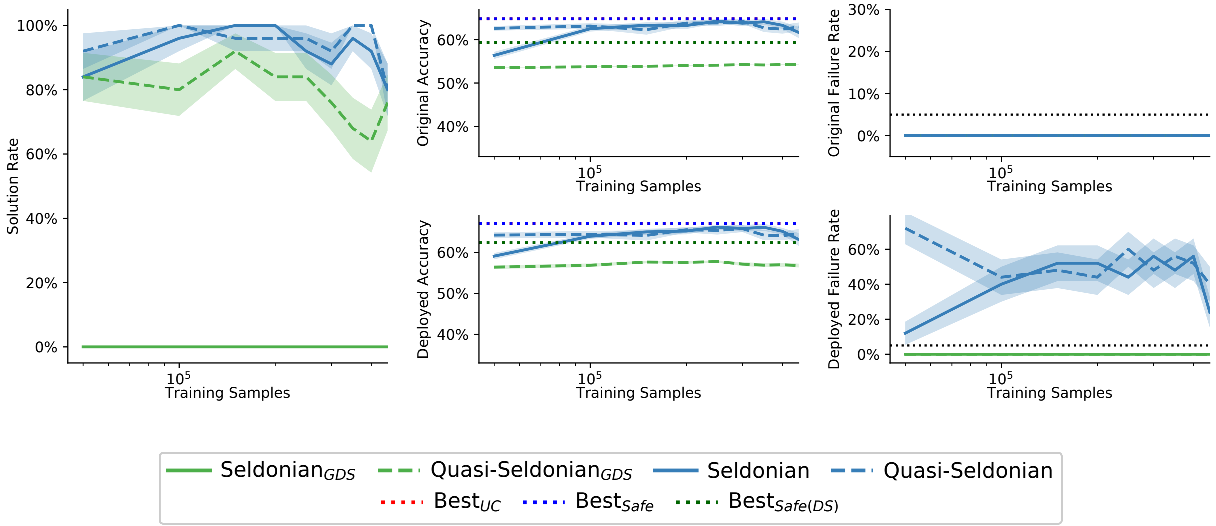

Figure 3.5. Results for experiments enforcing safety constraints based on the principle of equalized odds to preclude discrimination based on student sex when the marginal distribution of student race might change after model deployment. These results demonstrate a similar pattern as shown in Figure 3.2: The algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after deployment, while prior Seldonian algorithms (blue) do not. These results demonstrate that our proposed algorithms are effective solutions in safety-critical applications that are subject to demographic shift, and demonstrate that these benefits are consistent for a variety of practical safety definitions.

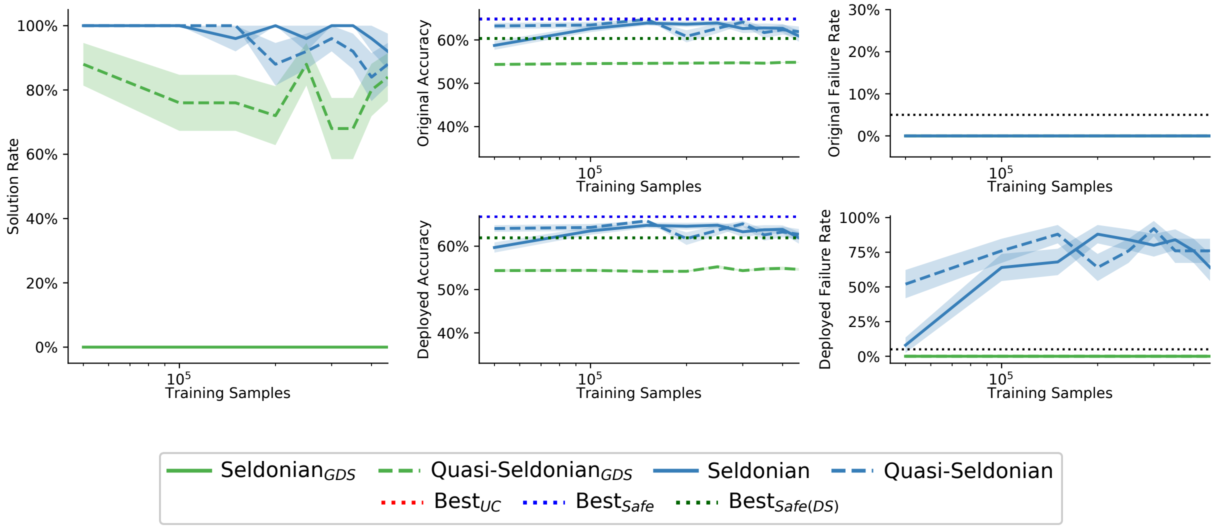

Figure 3.6. Results for experiments enforcing safety constraints based on the principle of predictive equality to preclude discrimination based on student sex when the marginal distribution of student race might change after model deployment. These results demonstrate a similar pattern as shown in Figure 3.2: The algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after deployment, while prior Seldonian algorithms (blue) do not. These results demonstrate that our proposed algorithms are effective solutions in safety-critical applications that are subject to demographic shift, and demonstrate that these benefits are consistent for a variety of practical safety definitions.

that is, the rightmost column of plots in each figure—the models produced by previous training algorithms were unsafe more than 5% of the time, even if they consistently appeared to be safe for the training data distribution. First, we note that standard Seldonian algorithms did not result in safety violations when evaluated on the training distribution. This observation validates that the safety guarantees provided by these algorithms—namely that with high probability they will not produce unsafe models on data drawn from the distribution used for—are valid. However, when evaluated on data from a deployed distribution that is impacted by demographic shift, the models produced by prior Seldonian algorithms are frequently unsafe. On the other hand, the models trained using the algorithms proposed in this chapter were consistently safe for the deployment distribution, verifying our hypothesis.

Next, we consider the accuracy of the models trained by our algorithms. In Section 3.4.1, we hypothesize that the models trained by our proposed algorithms will tend to have lower accuracy than those trained using standard Seldonian algorithms. This hypothesis is supported by our results. Examining the middle column of plots in each figure, we find that standard Seldonian algorithms achieve higher accuracy on average than the models trained using the algorithms proposed in this chapter. This pattern is reasonable, since our proposed algorithms must account for uncertainty about the deployment distribution, which prior Seldonian algorithms ignore. Thus, while prior Seldonian algorithms are able to identify models with higher performance, these models cannot be shown to be safe with high probability under demographic shift.

At first, the loss in accuracy demonstrated by our algorithms suggests that in many problems, there might be a tradeoff between performance and safety under demographic shift. To investigate this in more detail, we compare the accuracy when using each algorithm to our estimated best-case accuracy that can be achieved in various settings. First, we compare the accuracy of standard Seldonian algorithms, which provide high-confidence safety guarantees when evaluated on the training distribution, to the best-case accuracy of linear models that were found to be safe during training using oracle knowledge, which is shown as a dashed blue horizontal line. In each experiment, we find that standard Seldonian algorithms were able to identify models that come close to the best-case accuracy, showing that they not only are effective at providing high-confidence safety guarantees, but that they are also effective at finding accurate models subject to the safety constraints. On the other hand, we find that our proposed algorithms are less effective at achieving the best-case accuracy that can be attained for linear models that are safe under demographic shift, which is shown by a green dashed horizontal line. In most experiments, we observe that there is some cost in accuracy that is incurred when considering safety under demographic shift compared to safety during training, which is evidenced by the fact that the best-case accuracy obtained under demographic shift is generally close to five percentage points lower than the best case accuracy that can be achieved when enforcing safety during training. Nonetheless, our proposed algorithms tend to return models that achieve significantly lower accuracy than the estimated best-case value under demographic shift.

The inability of our proposed algorithms to produce models that achieve the best-case accuracy under safety constraints that depend on demographic shift may be due to uncertainty that results from using finite training data to evaluate our algorithms. Specifically, the best-case accuracy under these safety constraints is computed using oracle knowledge of the data distribution during training and deployment, while our proposed algorithms are evaluated using finite data. As a result, our proposed algorithms cannot perfectly evaluate $g(\theta)$, and must rely on high-confidence upper bounds. However, the fact that $g(\theta)$ cannot be evaluated exactly by our algorithms implies that there will be some models that are safe, but cannot be shown to be safe with high confidence, and therefore cannot be returned. This is particularly problematic for models that achieve $g(\theta)$ that is negative, but is very close to zero. We hypothesize that this effect is the central cause for the decreased accuracy observed for our proposed algorithms compared to standard Seldonian algorithms. For example, in our experiments on enforcing safety guarantees based on predictive equality, we found that the model that achieved best-case accuracy under safety constraints during training resulted in $g(\theta) \approx -0.0126$. Since standard Seldonian algorithms were able to return solutions that closely matched this best-case accuracy, it follows that the high-confidence upper bounds on $g(\theta)$ used by those algorithms typically overestimated the true value of $g(\theta)$ by less than 0.0126. However, the model that achieved best-case accuracy under safety constraints that involve demographic shift resulted in $g(\theta) \approx -0.0021$. Thus, for a robust Seldonian algorithm to return this model, the high-confidence upper bound on $g(\theta)$ would need to be no more than 0.0021 larger than the true value, which is a significantly stricter requirement than was encountered when enforcing constraints that must only hold during training. Therefore, in that experiment, we hypothesize that the models that achieve best-case accuracy are much closer to the boundary of the feasible set of safe models, and that consequently, our proposed algorithms were not able to ensure that they were safe with the required confidence. Consequently, we hypothesize that the inability of our proposed algorithms to return models that achieve best-case accuracy is due to uncertainty that arises when using finite data for training, and that the accuracy of models produced by these algorithms would improve given more training data.

Finally, we evaluate our hypothesis that our proposed algorithms would be less data-efficient than baseline algorithms, in the sense that they require significant amounts of training data to return a solution other than `NO_SOLUTION_FOUND`. Our experiments support this hypothesis. Specifically, the leftmost plot in each figure displays the frequency with which each algorithm returns a solution besides `NO_SOLUTION_FOUND`, for varying amounts of training data. Prior Seldonian algorithms were consistently more data-efficient than the algorithms proposed in this chapter, which can be explained by the observation that the use of importance sampling by our algorithms to account for demographic shift often increases the variance and range of the random variables used to define safety compared to standard Seldonian algorithms. Notably, the algorithms proposed in this chapter that were based on Hoeffding's inequality were unable to return solutions. However, in each experiment, the version of our proposed Seldonian algorithms that were based on inversion of the Student's t-test was eventually able to consistently produce solutions, demonstrating that they are viable tools for ensuring safety in problem settings where the demographic shift might occur.

3.6 Limitations and Future Work

While the algorithms proposed in this chapter are effective at providing safety guarantees that hold under demographic shift, there are several limitations of these methods, as well as directions for improving on these results.

First, the algorithms proposed in this chapter account for the fact that the user may not know the deployed distribution of the demographic variable, but they assume that certain quantities that depend on the training distribution are known. In many cases, these quantities may be easy to estimate accurately using training data, but the generality of our proposed algorithms could be improved by extending them to also account for uncertainty in the demographic distribution during training. Intuitively, this extension might assume that the demographic distribution during training is in some set, $\mathcal{P}$, and assume that the demographic distribution after deployment is in another set, $\mathcal{Q}$. Then, the value of the high-confidence upper and lower bounds on $g(\theta)$ could be optimized over all possible pairs of distributions, $(\mathbf{p}, \mathbf{q})$, where $\mathbf{p} \in \mathcal{P}$ and $\mathbf{q} \in \mathcal{Q}$, where $\mathcal{P}$ and $\mathcal{Q}$ are provided by the user.

Second, it is possible that the algorithms proposed in this chapter can be improved by modifying the strategy used to select candidate solutions. Since the strategy used to select candidate models does not impact the validity of our safety guarantees, we did not emphasize the design of effective candidate selection procedures for demographic shift. However, as demonstrated by our experimental results, the algorithms proposed in this chapter were unable to return solutions that achieved comparable accuracy to those returned by standard Seldonian approaches. It is possible that the candidate selection process can be further tuned to produce algorithms that produce high-confidence safety guarantees that hold under demographic shift, but are more data efficient and are able to better identify solutions that achieve high accuracy.

CHAPTER 4

CONTRIBUTION: SELDONIAN ALGORITHMS FOR GENERAL DISTRIBUTION SHIFT

As described in Chapter 3, existing safety-aware training algorithms, including previous Seldonian algorithms, make the limiting assumption that the data used for training is representative of the data that will be encountered once the trained model is deployed. The central problem with this assumption is that if it is not satisfied, then any safety guarantees provided by these algorithms may fail to hold once the model is deployed, since the distribution of any random variables that are used to define safety may have changed after deployment. Furthermore, there are many real-world problem settings for which this assumption does not hold. Therefore, we propose that for the safety guarantees provided by safe machine learning algorithms to be useful in practice, they must be extended to be robust to the various differences that might exist between the training and testing environments.

The solutions we propose in Chapter 3 address this problem when the differences between the training and deployment environments can be explained by a change in the probability distribution of a specific variable, which we refer to as a demographic. Under demographic shift, we show that safety guarantees can still be established as long as the user is able to provide some information about how the demographic marginal distribution changes. While this strategy is appropriate in many settings, it may be the case that the user is unable to identify a single demographic that causes the differences between the training and deployment distributions. Specifically, in a setting where the deployment data distribution may differ from the training distribution in nearly arbitrary ways, the solutions proposed in Chapter 3 become impractical. For example, in image recognition tasks, the distribution of input data might shift in ways that are difficult to describe due to differences in the hardware of cameras used in the training and deployment settings [23]. In this case, it is unreasonable to assume that the user will be able to identify a demographic attribute and provide information needed to describe the observed differences between the training and deployment environments.

To address this problem, in this chapter we propose strategies for establishing robust safety guarantees that are based on assumptions on how much the training and deployment distributions will differ when measured using an appropriate divergence measure. Our algorithms account for the possibility that the deployment distribution might change over time, and output models with safety guarantees that continue to hold as long as the user's assumptions are satisfied. In particular, the solutions we propose are based on the *goal-oriented divergence* [18], which relates the amount that an expected value can change due to distribution shift, to the *Kullback-Leibler divergence* (KLD) between the training and deployment distributions. However, applying the goal oriented divergence directly is challenging because it assumes that the training distribution is known exactly, and because it depends on quantities that can be challenging to compute.

To compensate for these drawbacks, we first show that the uncertainty in the training distribution can be accounted for by performing a worst-case analysis of the bounds when the true training distribution is within some set with high probability. By defining this set based on quantities that are already computed in standard Seldonian algorithms, we show that our worst-case bounds can be used to derive Seldonian algorithms that are robust to general distribution shift. However, while the bounds we propose are theoretically useful, they are challenging to compute. To remedy this, we further show that if the observables that appear in each expected value are discrete, then the computational cost of using these algorithms can be reduced significantly. While this assumption appears limiting, it describes many important cases, such as when the variable defining safety of a model is a binary indicator for some unsafe event, such as an unfair or unsafe decision occurring.

In the remainder of this chapter, we first describe the bounds introduced by the goal-oriented divergence, and explain their benefits and limitations. Next, we present our strategies for extending these results to produce robust Seldonian algorithms that can be computed efficiently. After this, we briefly discuss how assumptions on the KLD between the training and deployment distributions can be related to other measures of difference between the two environments, which may be more appropriate in some problem settings. Finally, we present experiments demonstrating that the algorithms we propose are effective for enforcing safety guarantees for general definitions under distribution shift, while prior methods are not.

4.1 Background

In this chapter, we adopt the notation that $Z \in \mathcal{Z}$ is a random variable representing a single instance of data, such as a labeled feature vector in classification, or a tuple containing a context, an action, and a reward for contextual bandit problems. Defining training data D to represent a set of n i.i.d. samples of Z, the goal of our work is to construct algorithms that provide the same guarantees as existing Seldonian methods, even when the distribution of D has changed between training and deployment. Our algorithms are based on the Seldonian machine learning framework, which states that an algorithm, a, is safe with respect to a user-specified safety definition, g, and failure tolerance, δ, if it satisfies,

$$\Pr\big(g\big(a(D)\big) \leq 0\big) \geq 1{-}\delta. \tag{4.1}$$

Whereas existing Seldonian algorithms enforce behavioral constraints based on Z, here we consider the case in which these constraints are instead based on Z', which represents the data instances observed after the model is deployed. For example, consider the safety definition,

$$g(\theta) \coloneqq \mathbf{E}[h(Z,\theta)] - \tau,$$

where h is some real-valued observable that depends on the the model, θ, and τ is some tolerance. If $P(z) \coloneqq \Pr(Z{=}z)$ is the probability distribution of data instances during training and $Q(z) \coloneqq \Pr(Z'{=}z)$ is the corresponding distribution after the model is deployed, then prior Seldonian algorithms ignore the possibility of distribution shift, and would proceed by computing a high-confidence upper bound on $\mathbf{E}[h(Z,\theta)]$ using available data. However, because observations are distributed according to Q after deployment, the safety of θ is more appropriately assessed by measuring,

$$g'(\theta) \coloneqq \mathbf{E}[h(Z',\theta)] - \tau.$$

As a result, to design Seldonian algorithms that provide high-confidence safety guarantees under distribution shift, we are therefore interested in bounds on how $\mathbf{E}[h(Z,\theta)]$ might differ from $\mathbf{E}[h(Z,\theta)]$, given some assumptions on how much Q differs from P.

To this end, we describe the *goal-oriented divergence* [18], which provides bounds on the quantity $\mathbf{E}[h(Z',\theta)] - \mathbf{E}[h(Z,\theta)]$ that depend on the KLD between Q and P, which is defined as the following quantity [43]:

$$\text{KLD}\left(Q||P\right) \coloneqq \int_{\mathcal{Z}} \log\left(\frac{dQ}{dP}\right) dQ.$$

However, before introducing the result that we extend in this chapter, we first introduce some notation. First, for notational convenience, and because θ will be fixed when evaluating these bounds, we do not explicitly show the dependence of h on θ, and write $h(Z)$ instead of $h(Z,\theta)$ in the results below. Next, we introduce a quantity that appears in the goal-oriented divergence bounds, which is the *cumulant generating function* (CGF) of a random variable. Specifically, the cumulant generating function of a random variable with distribution P with respect to a functional h is denoted $\Lambda_{P,h}$, and is a real-valued function defined by,

$$\Lambda_{P,h}(c) \coloneqq \log \mathbf{E}\left[e^{ch(Z)}\right],$$

where $c \in \mathcal{R}$ is any real value. In particular, the goal oriented divergence depends on the *centered* CGF of P given h, denoted $\tilde{\Lambda}_{P,h}$, which is simply the CGF of $h(Z){-}\mathbf{E}[h(Z)]$. While intuition for the form of these functions can be difficult to express, cumulant generating functions have been extensively studied by statisticians and used to derive several important results. Finally, the notation $Q << P$ indicates that Q is *absolutely continuous* with respect to P which, for discrete random variables, is satisfied if $P(z){=}0$ implies $Q(z){=}0$ for all $z \in \mathcal{Z}$.

Given this notation, we now derive the *goal-oriented divergence* [18], which provides bounds on $\mathbf{E}[h(Z',\theta)] - \mathbf{E}[h(Z,\theta)]$ that depend on the KLD between Q and P, which will prove useful for deriving machine learning algorithms that provide high-confidence safety guarantees under distribution shift. Note that while the upper bound in this result was proven by Chowdhary and Dupuis [10] and the lower bound was proven by Li and Xiu [41], we include a proof showing how these bounds are derived in Theorem 4.1.1 for completeness.

Theorem 4.1.1. *Let Z and Z' be two random variables defined on the set $\mathcal{Z}$, and let P and Q denote the probability distribution functions of Z and Z', respectively. Let h denote a real-valued observable defined on $\mathcal{Z}$. If Z and Z' have well-defined CGFs and if Q is absolutely continuous with respect to P, then*

$$-\inf_{c>0}\left\{\frac{\mathit{KLD}\left(Q||P\right)+\tilde{\Lambda}_{P,h}(-c)}{c}\right\} \quad\leq\quad \mathbf{E}[h(Z')]{-}\mathbf{E}[h(Z)] \quad\leq\quad \inf_{c>0}\left\{\frac{\mathit{KLD}\left(Q||P\right)+\tilde{\Lambda}_{P,h}(c)}{c}\right\}.$$

Proof. To begin, we start with the Donsker–Varadhan variational formula, which states that for all $c \in \mathcal{R}$ and functional $f : \mathcal{Z} \to \mathcal{R}$, the following holds:

$$\Lambda_{P,h}(c) = \sup_{Q<<P}\Big\{c\mathbf{E}[f(Z')] - \text{KLD}\left(Q||P\right)\Big\}.$$

Applying this result using the functional, $f(z) = h(z){-}\mathbf{E}[h(Z)]$, yields the following:

$$\tilde{\Lambda}_{P,h}(c) = \sup_{Q<<P}\Big\{c\big(\mathbf{E}[h(Z')] - \mathbf{E}[h(Z)]\big) - \text{KLD}\left(Q||P\right)\Big\}, \tag{4.2}$$

where $\tilde{\Lambda}_{P,h}$ is the centered CGF of P given h.

This implies that for all Z and Z' with distributions P and Q that satisfy $Q << P$,

$$\tilde{\Lambda}_{P,h}(c) \geq c\big(\mathbf{E}[\, h(Z')\,] - \mathbf{E}[\, h(Z)\,]\big) - \text{KLD}\,(Q||P)\,,$$

for all $c \in \mathcal{R}$. Defining $c_+ \in [0, \infty)$ to be any non-negative real value, it follows that

$$\tilde{\Lambda}_{P,h}(c_+) \geq c_+\big(\mathbf{E}[\, h(Z')\,] - \mathbf{E}[\, h(Z)\,]\big) - \text{KLD}\,(Q||P)\,. \tag{4.3}$$

Rearranging terms, we have,

$$\begin{aligned} \mathbf{E}[\, h(Z')\,]-\mathbf{E}[\, h(Z)\,] \;&\leq\; \frac{\tilde{\Lambda}_{P,h}(c_+) + \text{KLD}\,(Q||P)}{c_+} \\ &=\; \inf_{c_+>0}\left\{\frac{\tilde{\Lambda}_{P,h}(c_+) + \text{KLD}\,(Q||P)}{c_+}\right\}. \end{aligned}$$

Thus, the first half of the theorem is proven.

To prove the lower bound, we note that (4.3) must hold for $\tilde{\Lambda}_{P,h}(-c_+)$ as well. Thus for all $Q << P$, we have:

$$\tilde{\Lambda}_{P,h}(-c_+) \geq -c_+\big(\mathbf{E}[\, h(Z')\,] - \mathbf{E}[\, h(Z)\,]\big) - \text{KLD}\,(Q||P)\,.$$

Rearranging terms once more produces,

$$\begin{aligned} \mathbf{E}[\, h(Z')\,]-\mathbf{E}[\, h(Z)\,] \;&\geq\; -\frac{\tilde{\Lambda}_{P,h}(-c_+) + \text{KLD}\,(Q||P)}{c_+} \\ &=\; -\inf_{c_+>0}\left\{\frac{\tilde{\Lambda}_{P,h}(-c_+) + \text{KLD}\,(Q||P)}{c_+}\right\}, \end{aligned}$$

completing the proof. □

As a consequence of Theorem 4.1.1, if the user of a safe machine learning algorithm is able to assume that the KLD between the deployment and training environments is bounded by some value, ϵ, and P and h are known during training, it is possible to compute bounds on $\mathbf{E}[h(Z')]$ that can be used to provide high-confidence safety guarantees under distribution shift. For example, by rearranging terms, an upper bound on $\mathbf{E}[h(Z')]$ given KLD $(Q||P) \leq \epsilon$ is given by

$$\mathbf{E}[h(Z')] \leq \mathbf{E}[h(Z)] + \inf_{c>0} \left\{ \frac{\epsilon + \tilde{\Lambda}_{P,h}(c)}{c} \right\}. \tag{4.4}$$

Unfortunately, the results of Theorem 4.1.1 cannot be directly applied to develop safe machine learning algorithms because of two limitations.

First, this result can only be applied when the distribution over training observations is known exactly, since it is required to compute the CGF. Unlike the methods proposed in Chapter 3, which require that the user knows the marginal distribution over demographics during training, this assumption implies that the user must know the entire distribution over observations, which is unrealistic in many applications.

Second, the centered CGF appearing in these bounds is generally intractable to compute. Consider, for example, the expression for the upper bound on $\mathbf{E}[h(Z')]$ given KLD $(Q||P) \leq \epsilon$, which is given by

$$\inf_{c>0} \left\{ \frac{\epsilon + \tilde{\Lambda}_{P,h}(c)}{c} \right\} = \inf_{c>0} \left\{ \frac{\epsilon + \log \mathbf{E}\left[e^{c(h(Z)-\mathbf{E}[h(z)])}\right]}{c} \right\}. \tag{4.5}$$

Computing this upper bound requires evaluating two expected values defined with respect to the distribution of the random variable, Z. Since there are no constraints on how Z is defined, these expected values can be intractable to compute, especially when Z is high-dimensional or when it contains a mixture of discrete and continuous features. For example, consider the task of performing binary classification using gray scale images that are 100×100 pixels in dimension. If each pixel takes values in $[0, 1]$, then the training distribution P is defined over the set of set of all observations, $\mathcal{Z} := [0,1]^{10{,}000} \times \{0,1\}$. Consequently, evaluating the expected values defining $\tilde{\Lambda}_{P,h}(c)$ will often be computationally expensive, even if P is known.

4.2 Bounds on Mean Shift due to General Distribution Shift

Theorem 4.1.1 provides bounds needed to produce safe machine learning methods that are robust to general distribution shift. However, whether or not these bounds can be successfully applied depends on the existence of strategies for estimating them using limited data, as well as whether or not KLD is a natural way to define distribution shift for a given application. Importantly, as described in the previous section, the centered CGF, in its most general form, is either often intractable to compute or requires unrealistic knowledge about the distribution of observations during training. Furthermore, ignoring these computational issues, computing the upper bound in this form requires the user to have complete knowledge of $\Pr(Z{=}z)$ for all $z \in \mathcal{Z}$, which is often an unreasonable assumption. Therefore, while Theorem 4.1.1 provides a theoretical result that can be used to ensure that safety guarantees are robust to general distribution shift, this initial formulation does not easily lead to practical algorithms.

To overcome these issues, we present two modifications to these results, which allow them to be used to design practical algorithms that provide safety guarantees under distribution shift.

4.2.1 Simplification by Assuming Discrete Observables

Our first proposed modification consists of assuming that functional, h, takes values in a discrete set. Concretely, we assume that $h(Z) \in \mathcal{V}$ for some finite set of values, $\mathcal{V}$. For example, this occurs when h is a binary-valued indicator function that detects that some undesirable event has occurred.

To formalize this setting, let $\mathbf{v}$ be the vector of values that h can take, and let $\mathbf{p} := [\mathbf{p}_1, ..., \mathbf{p}_k]$ represent the corresponding probability mass function of $h(Z)$ during training, so that $\mathbf{p}_i := \Pr(h(Z){=}\mathbf{v}_i)$. In this setting, the computation of the upper and lower bounds can be simplified significantly by noting that many distinct Z produce $h(Z)$ with the same value. Consequently, when computing $\hat{\Lambda}_{P,h}(c)$ within each bound, the expected values involved can be represented by a simpler form than an integration over $\mathcal{Z}$. This result is significant because it implies that bounds that account for variation of the potentially complicated random variable Z, can be computed based on observations of $h(Z)$, which are much simpler.

For example, consider a self-driving car navigating a route. In this case, Z might represent the collection of all data available from the car's sensors—including image data, current speed, and many other features—while $h(Z)$ might be a simple binary indicator for whether or not the car is currently near the edge of the road. While the full distribution over Z might be complex and high-dimensional, the distribution of $h(Z)$ is much simpler. While Theorem 4.1.1 provide bounds on how much the expected value of $h(Z)$ can change if the distribution of Z changes, the following lemma shows that if $h(Z)$ is discrete, then these bounds can be computed without explicitly integrating over the distribution of Z.

Lemma 4.2.1. *Let Z be a random variable taking values in some set $\mathcal{Z}$, and let $h : \mathcal{Z} \to \mathcal{V}$ be a functional that takes values in some finite set of real values, $\mathcal{V}$. Let P be the probability distribution of Z, and let $\mathbf{p}$ be a vector representing the corresponding probability mass function of $h(Z)$:*

$$\mathbf{p}_i \coloneqq \Pr\big(h(Z) = \mathbf{v}_i\big),$$

where $\mathbf{v}$ is a vector of the values in $\mathcal{V}$. It follows that

$$\tilde{\Lambda}_{P,h}(c) = \log \mathbf{p}^\intercal \exp\big(c(\mathbf{v} - \mathbf{p}^\intercal \mathbf{v})\big).$$

Proof. Let $\mathcal{Z}_v$ denote the set of $z \in \mathcal{Z}$ for which $h(z) = v$. First, we note that for any function $\xi : \mathcal{V} \to \mathbb{R}$, we have that $\mathbf{E}[\xi(h(Z))] = \sum_i \mathbf{p}_i \xi(\mathbf{v}_i)$:

$$\begin{aligned}\mathbf{E}[\xi(h(Z))] &= \int_{z\in\mathcal{Z}} \xi(h(z))\mathrm{d}P \\ &= \sum_i \int_{z\in\mathcal{Z}_{\mathbf{v}_i}} \Pr\left(h(Z){=}\mathbf{v}_i\right)\xi(\mathbf{v}_i) \\ &= \sum_i \mathbf{p}_i\,\xi(\mathbf{v}_i).\end{aligned}$$

Expanding the definition of $\hat{\Lambda}_{P,h}$, we have:

$$\hat{\Lambda}_{P,h}(c) = \mathbf{E}\left[\exp\left(c\left(h(Z) - \mathbf{E}[h(Z)]\right)\right)\right].$$

By applying the above substitution to the outermost expected value and defining ξ so that $\xi(h(Z)) = \exp\left(c\left(h(Z) - \mathbf{E}[h(Z)]\right)\right)$, we have:

$$\hat{\Lambda}_{P,h}(c) = \sum_i \mathbf{p}_i \exp\left(c\left(\mathbf{v}_i - \mathbf{E}[h(Z)]\right)\right].$$

Applying the previous substitution once more with $\xi(h(Z)) = f(Z)$, we have,

$$\hat{\Lambda}_{P,h}(c) = \sum_i \mathbf{p}_i \exp\left(c\left(\mathbf{v}_i - \sum_i \mathbf{p}_i\mathbf{v}_i\right)\right).$$

Noting that for any pair of vectors $\mathbf{p}$ and $\mathbf{x}$, $\sum_i \mathbf{p}_i\mathbf{x}_i = \mathbf{p}^\intercal\mathbf{x}$, this substitution completes the proof.

□

Lemma 4.2.1 shows that, if $\mathbf{p}$ is known and $\mathrm{KLD}\left(Q||P\right) \leq \epsilon$, then the centered CGF of P given h, and thus the bounds on $\mathbf{E}[h(Z')] - \mathbf{E}[h(Z)]$, can be computed exactly without explicitly integrating over the distribution of Z. Thus, if $g(\theta) \coloneqq \mathbf{E}[h(Z)]$ and the distribution of $h(Z)$ is known, then an upper bound on the value of $g(\theta)$ after distribution shift is straightforward to compute. However, in the context of designing Seldonian machine learning algorithms, knowledge of the values that $h(Z)$ can take, and its distribution during training, amounts to the assumption that

$g(\theta) = \mathbf{E}[h(Z)] - \tau$ can be computed exactly. Therefore, while the modification proposed in this section overcomes many of the computational challenges associated with computing the bounds in Theorem 4.1.1, it remains difficult to apply them to provide algorithms with high-confidence safety guarantees under distribution shift.

4.2.2 Extending Theorem 4.1.1 for Uncertain Training Distributions

As stated in the previous sections, the bounds in Theorem 4.1.1 assume that the distribution of Z is known. While the modification presented in Section 4.2.1 reduces the cost of computing these bounds and loosens this assumption by instead requiring that the distribution of $h(Z)$ is known, it is still challenging to use these bounds to construct safe machine learning algorithms.

To make our previous results more practical, we therefore consider cases in which the distribution of $h(Z)$ can be shown to be in some confidence region with high probability. In particular, consider a Seldonian algorithm designed according to Algorithm 1 for definitions of the form, $g(\theta) := \mathbf{E}[h(Z, \theta)] - \tau$. During the safety test, such algorithms proceed by computing a high-confidence upper bound on $g(\theta)$. Consequently, while $\mathbf{E}[h(Z)]$ is not known exactly, standard Seldonian algorithms construct an interval, $(-\infty, U(D) + \tau)$, that contains $\mathbf{E}[h(Z)]$ with high probability. In this section, we leverage this information to define a set of distributions over $\mathcal{V}$ that contains the true distribution of $h(Z)$ with high probability. Then, by maximizing the upper bounds described in the previous section over all $\mathbf{p}$ in this set, we obtain a high-confidence worst-case upper bound on $\mathbf{E}[h(Z')]$.

To define such a set, we first let $U(D)$ denote a high-confidence upper bound on $\mathbf{E}[h(Z)]$ that is computed by a standard Seldonian algorithm during the safety test, using the procedure `HighConfidenceUpperBound` shown in Algorithm 1. By assuming that h takes values in a discrete set, it follows that $\mathbf{E}[h(Z)] = \mathbf{p}^\intercal \mathbf{v}$. Therefore, we define $\mathcal{P}(D)$ to a subset of the k-dimensional probability simplex containing distributions that produce expected values that are smaller than $U(D)$, as follows,

$$\mathcal{P}(D) \coloneqq \Big\{\mathbf{p} \;\text{ s.t. }\; \mathbf{p}^\intercal\mathbf{v} \leq U(D), \;\; \forall i, \mathbf{p}_i \geq 0, \;\; \text{and} \;\; \sum_i \mathbf{p}_i = 1\Big\}. \tag{4.6}$$

If $\mathbf{p}^*$ denotes the true distribution of $h(Z)$ during training, then by construction, $\mathbf{p}^* \notin \mathcal{P}(D)$ implies $\mathbf{E}[h(Z)] > U(D)$, meaning the upper bound has failed. However, since the upper bound is computed to hold with probability $1{-}\delta$, it follows that,

$$\Pr(\mathbf{p} \in \mathcal{P}) \geq 1{-}\delta. \tag{4.7}$$

Given $\mathcal{P}(D)$, we present a high-confidence upper bound on $\mathbf{E}[h(Z')]$ by maximizing the bound provided in Section 4.2.1 over all $\hat{\mathbf{p}} \in \mathcal{P}(D)$. Concretely, we have the following theorem.

Theorem 4.2.2. *Let Z and Z' be random variables taking values in some set $\mathcal{Z}$ with probability distributions P and Q, respectively, and let P and Q satisfy $Q << P$. Let h be some functional that takes values in a finite set of k real numbers, $\mathcal{V}$, and let $\mathbf{p}$ be a vector representing the probability mass function of $h(Z)$—that is, $\mathbf{p}_i \coloneqq \Pr(h(Z){=}\mathbf{v}_i)$, where $\mathbf{v}$ is a vector of the values in $\mathcal{V}$. Finally, let D represent a set of i.i.d. observations used to evaluate a $(1{-}\delta)$-confidence upper bound on $\mathbf{E}[h(Z)]$,*

$$\Pr\big(\mathbf{E}[h(Z)] \leq U(D)\big) \geq 1{-}\delta,$$

for some $\delta \in [0,1]$. It follows that if KLD $(Q||P) \leq \epsilon$, then with probability at least $1{-}\delta$, the following inequalities hold jointly:

$$\begin{aligned}
\mathbf{E}\big[h(Z')\big] \;&\leq\; \max_{\hat{\mathbf{p}} \in \mathcal{P}(D)} \left\{ \hat{\mathbf{p}}^\intercal\mathbf{v} + \inf_{c>0} \left\{ \frac{\epsilon + \log \hat{\mathbf{p}}^\intercal \exp\big(c(\mathbf{v} - \hat{\mathbf{p}}^\intercal\mathbf{v})\big)}{c} \right\} \right\}, \quad \textit{and} \\
\mathbf{E}\big[h(Z')\big] \;&\geq\; \min_{\hat{\mathbf{p}} \in \hat{\mathcal{P}}(D)} \left\{ \hat{\mathbf{p}}^\intercal\mathbf{v} - \inf_{c>0} \left\{ \frac{\epsilon + \log \hat{\mathbf{p}}^\intercal \exp\big(-c(\mathbf{v} - \hat{\mathbf{p}}^\intercal\mathbf{v})\big)}{c} \right\} \right\},
\end{aligned}$$

where

$$\mathcal{P}(D) \coloneqq \Big\{\mathbf{p} \;\textit{ s.t. }\; \mathbf{p}^\intercal\mathbf{v} \leq U(D), \;\; \forall i, \mathbf{p}_i \geq 0, \;\; \textit{and} \;\; \sum_i \mathbf{p}_i = 1\Big\}.$$

Proof. Let $\mathbf{p}^*$ represent the true, unknown distribution of $h(Z)$ during training. If $\mathbf{p}^* \in \mathcal{P}(D)$, then it follows that,

$$\begin{aligned}&\mathbf{p}^{*\intercal}\mathbf{v} + \inf_{c>0}\left\{\tfrac{\epsilon+\log \mathbf{p}^{*\intercal}\exp\left(c(\mathbf{v}-\mathbf{p}^{*\intercal}\mathbf{v})\right)}{c}\right\}\\ &\leq \max_{\hat{\mathbf{p}}\in\mathcal{P}(D)}\left\{\hat{\mathbf{p}}^\intercal\mathbf{v} + \inf_{c>0}\left\{\tfrac{\epsilon+\log \hat{\mathbf{p}}^\intercal\exp\left(c(\mathbf{v}-\hat{\mathbf{p}}^\intercal\mathbf{v})\right)}{c}\right\}\right\}.\end{aligned}$$

Applying Theorem 4.1.1, Theorem 4.2.1, and the substitution, $\mathbf{E}[h(Z)] = \mathbf{p}^{*\intercal}\mathbf{v}$, we also have that

$$\mathbf{E}[h(Z')] \leq \mathbf{p}^{*\intercal}\mathbf{v} + \inf_{c>0}\left\{\tfrac{\epsilon+\log \mathbf{p}^{*\intercal}\exp\left(c(\mathbf{v}-\mathbf{p}^{*\intercal}\mathbf{v})\right)}{c}\right\},$$

since KLD $(Q||P) \leq \epsilon$. Combining these, it follows that if $\mathbf{p}^* \in \mathcal{P}(S_n)$, then,

$$\mathbf{E}\big[h(Z')\big] \leq \max_{\hat{\mathbf{p}}\in\mathcal{P}(D)}\left\{\hat{\mathbf{p}}^\intercal\mathbf{v} + \inf_{c>0}\left\{\frac{\epsilon+\log \hat{\mathbf{p}}^\intercal\exp\left(c(\mathbf{v}-\hat{\mathbf{p}}^\intercal\mathbf{v})\right)}{c}\right\}\right\}.$$

A similar derivation shows that if $\mathbf{p}^* \in \mathcal{P}(S_n)$, then,

$$\mathbf{E}\big[h(Z')\big] \geq \min_{\hat{\mathbf{p}}\in\hat{\mathcal{P}}(D)}\left\{\hat{\mathbf{p}}^\intercal\mathbf{v} - \inf_{c>0}\left\{\frac{\epsilon+\log \hat{\mathbf{p}}^\intercal\exp\left(-c(\mathbf{v}-\hat{\mathbf{p}}^\intercal\mathbf{v})\right)}{c}\right\}\right\}.$$

Because $\Pr\left(\mathbf{p}\in\mathcal{P}(D)\right) \geq 1-\delta$, it follows that the probability that $\mathbf{E}[h(Z')]$ is between these two values is at least $1-\delta$, completing the proof. □

4.2.3 Seldonian Algorithms for Distribution Shift

Given Theorem 4.2.2, we propose an algorithm that efficiently computes those bounds given a finite number of i.i.d. samples of $h(Z)$, which hold provided the training and deployment distributions differ by at most ϵ according to the KL divergence. While optimization of $\mathbf{p}$ is complicated by the fact that the feasible set of the optimization is a subset of a simplex, in Appendix A we

Algorithm 8 HighConfidenceUpperBoundGDS(θ, g, D, δ, ϵ, $\mathbf{v}$, c_0, $\mathbf{p}_0$)

```
$u \leftarrow$ HighConfidenceUpperBound$(\theta, g, D, \delta)$
$\mathcal{P} \leftarrow \{\mathbf{p}$ s.t. $\mathbf{p}^\intercal \mathbf{v} \leq u, \ \forall i, \mathbf{p}_i \geq 0,$ and $\sum_i \mathbf{p}_i = 1\}$
$\mathbf{p} \leftarrow \mathbf{p}_0$
$c \leftarrow c_0$
while $\mathbf{p}$ and $c$ have not converged do
    $\mathbf{p} \leftarrow$ MaximizeOverP $(c, \epsilon, \mathbf{v}, \mathbf{p}, \mathcal{P})$
    $c \leftarrow$ MinimizeOverC $(c, \epsilon, \mathbf{v}, \mathbf{p})$
end while
return $\mathbf{p}^\intercal \mathbf{v} + \frac{\epsilon + \log \hat{\mathbf{p}}^\intercal \exp(-c(\mathbf{v} - \hat{\mathbf{p}}^\intercal \mathbf{v}))}{c}$
```

prove several properties that allow this optimization to be computed accurately, and efficiently. Pseudocode for computing the robust upper bound on $g(\theta)$ is provided in Algorithm 8. In this pseudocode, the optimization steps involved with computing the upper bound is abstracted into `MinimizeOverC` and `MaximizeOverP`, which we describe in detail in the following subsections. Pseudocode for computing the robust lower bound on $g(\theta)$ is analogous to Algorithm 8 and is not included here.

Note that in Algorithm 8, we assume that $g(\theta) = \mathbf{E}[h(Z)]$. To consider conditional expected values, the user may define their assumptions on the KLD between the conditional distributions over $\mathcal{Z}$. Furthermore, to incorporate a tolerance, τ, the set $\mathcal{P}$ can be defined to contain distributions that satisfy $\mathbf{p}^\intercal \mathbf{v} \leq U(D) - \tau$. Finally, to leverage the general interface proposed in Chapter 2, one can compute the high-confidence upper bounds presented here, as well as corresponding lower bounds, for each statistic that defines $g(\theta)$ in (2.2). The resulting intervals for each statistic can then be used in place of the intervals described in Chapter 2.

4.2.3.1 Optimization of Over $c > 0$ Given Fixed $\mathbf{p}$

Optimization of the bounds over $c > 0$ is simplified by the fact that cumulant generating functions are, by definition, convex. Consequently, standard gradient-based optimization procedures are efficient. In Lemma 4.2.3, we derive expressions for the derivatives of the upper and lower bounds with respect to $c > 0$, for a fixed choice of $\mathbf{p} \in \mathcal{P}$. Then, we provide pseudocode for optimizing the bounds, which are based on standard gradient descent.

Lemma 4.2.3. *Let* $\mathbf{v}$ *be a vector of real values, and let* $\mathbf{p}$ *be a vector of probabilities satisfying* $\Pr(f(Z){=}v_i) = p_i$*, where* $f(Z)$ *is a discrete random variable. Let U and L be defined by,*

$$U_\epsilon(\mathbf{p}, c) = \mathbf{p}^\intercal \mathbf{v} + \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c)}{c}, \quad \textit{and} \quad L_\epsilon(\mathbf{p}, c) = \mathbf{p}^\intercal \mathbf{v} - \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(-c)}{c}.$$

The derivatives of U and L with respect to $c > 0$ *are given by,*

$$\frac{dU_\epsilon(\mathbf{p}, c)}{dc} = -\frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c)}{c^2} + \frac{\mathbf{v} - \mathbf{p}^\intercal \mathbf{v}}{c} \quad \textit{and} \quad \frac{dL_\epsilon(\mathbf{p}, c)}{dc} = \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(-c)}{c^2} - \frac{\mathbf{p}^\intercal \mathbf{v} - \mathbf{v}}{c}.$$

Proof. For notational convenience, let $\tilde{M}_{\mathbf{p},f}$ denote the centered moment-generating function of $\mathbf{p}$ given f, which is simply the natural exponential of the centered CGF, $\tilde{\Lambda}_{\mathbf{p},f}(c) = \log \tilde{M}_{\mathbf{p},f}(c)$.

To derive $\frac{\mathrm{d}U_\epsilon(\mathbf{p},c)}{\mathrm{d}c}$, we have,

$$\begin{aligned}
\frac{\mathrm{d}U_\epsilon(\mathbf{p}, c)}{\mathrm{d}c} &= -\frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c)}{c^2} + \frac{1}{c}\frac{\mathrm{d}}{\mathrm{d}c}\tilde{\Lambda}_{\mathbf{p},f}(c) \\
&= -\frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c)}{c^2} + \frac{1}{c\tilde{M}_{\mathbf{p},f}(c)}\,\mathbf{p}^\intercal \frac{\mathrm{d}}{\mathrm{d}c}\exp(c\mathbf{v}{-}c\mathbf{p}^\intercal\mathbf{v}) \\
&= -\frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c)}{c^2} + \frac{\tilde{M}_{\mathbf{p},f}(c)}{c\tilde{M}_{\mathbf{p},f}(c)}\,\frac{\mathrm{d}}{\mathrm{d}c}(c\mathbf{v}{-}c\mathbf{p}^\intercal\mathbf{v}) \\
&= -\frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c)}{c^2} + \frac{1}{c}\,\frac{\mathrm{d}}{\mathrm{d}c}(c\mathbf{v}{-}c\mathbf{p}^\intercal\mathbf{v}) \\
&= -\frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c)}{c^2} + \frac{\mathbf{v} - \mathbf{p}^\intercal\mathbf{v}}{c}.
\end{aligned}$$

A similar derivation shows that

$$\frac{\mathrm{d}L_\epsilon(\mathbf{p}, c)}{\mathrm{d}c} = \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(-c)}{c^2} - \frac{\mathbf{p}^\intercal \mathbf{v} - \mathbf{v}}{c}.$$

□

Algorithm 9 MinimizeOverC($c_0, \mathbf{p}, \epsilon, \mathbf{v}, \alpha$)

```
for $t = 1, 2, ...$ do
    $g_t \leftarrow \mathbf{p}^\intercal \mathbf{v} + \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},f}(c_{t-1})}{c_{t-1}}$
    $c_t \leftarrow c_{t-1} + \alpha g_t$
    if $c_t \approx c_{t-1}$ then
        return $c_t$
    end if
end for
```

4.2.3.2 Optimization of Over $\mathbf{p} \in \mathcal{P}$ Given Fixed c

Unfortunately, optimizing the bounds over $\mathbf{p} \in \mathcal{P}$ is more complicated than optimizing them over $c > 0$ for several reasons. First, $\mathbf{p}$ is constrained to lie within a particular subset of a k-dimensional probability simplex, $\mathcal{P} \subseteq \Delta_k$. As a result, the raw gradients of the upper and lower bounds must first be modified to ensure that they produce iterates that lie in $\mathcal{P}$. Second, while the upper and lower bounds are smooth functions of $\mathbf{p}$, it can be shown that the upper bound is not concave, and similarly the lower bound is not convex. Consequently, although gradient-based approaches to optimizing these bounds are reasonable, it can be challenging to derive schemes for computing step sizes that lead to efficient convergence.

Fortunately, we found that both of these challenges could be dealt with effectively. Below, we describe these results at a high level, and provide more thorough details in Appendix A

First, we show that the gradients of each bound with respect to $\mathbf{p}$ can be modified to enforce the constraint, $\mathbf{p} \in \mathcal{P}$. In particular, we describe a projection operator that can be used to modify the gradient vectors so that they preserve the property that the sum of the entries in $\mathbf{p}$ must be equal to one. In addition, we show that the projected gradients can then be further modified to ensure, for sufficiently small step sizes, that they result in a sequence of iterates that is guaranteed to be in $\mathcal{P}$. Finally, we prove two properties of the upper and lower bound functions that justify the use of gradient-based approaches, and show that the problem of determining the correct sequence of step sizes for efficient convergence can be dealt with by always choosing the largest possible step size that ensures the next iterate lies in $\mathcal{P}$.

Algorithm 10 MaximizeOverP($c, \epsilon, \mathbf{v}, \mathbf{p}, \mathcal{P}$)

```
for $t = 1, ...$ do
    $\mathbf{g} \leftarrow \frac{\exp(c\mathbf{v})}{c\,\mathbf{p_t}^\intercal \exp(c\mathbf{v})}$
    $\bar{\mathbf{g}} \leftarrow \mathbf{g} - \left(\sum_{j=1}^{k} \mathbf{g}_j\right) \mathbf{1}_k$
    if $\mathbf{p}_t$ lies on a boundary of $\mathcal{P}$ then
        for each tangent hyperplane, $i$, of the boundary of $\mathcal{P}$ at $\mathbf{p}_t$ do
            $\mathbf{n}_i \leftarrow$ the normal vector to hyperplane $i$
        end for
        $\hat{\mathbf{g}} \leftarrow \arg\min_{\mathbf{g}' \in \mathbb{R}^k} \|\bar{\mathbf{g}} - \mathbf{g}'\|^2 \quad \text{s.t.} \quad \forall i \in \{1, ..., d\},\ \mathbf{g}'^T \mathbf{n}_i \leq 0.$
    else
        $\hat{\mathbf{g}} \leftarrow \bar{\mathbf{g}}$
    end if
    $\alpha^* \leftarrow \max\{\alpha > 0 \ : \ \mathbf{p}_t + \alpha\hat{\mathbf{g}} \in \mathcal{P}\}$
    if $\hat{\mathbf{g}} \approx \mathbf{0}_k$ then
        return $\mathbf{p}_t$
    else
        $\mathbf{p}_{t+1} \leftarrow \mathbf{p}_t + \alpha\hat{\mathbf{g}}$
    end if
end for
```

Next, while these modified gradients form the basis for optimizing the upper and lower bounds, we show that the upper and lower bound functions are only quasiconcave and quasiconcave and $\mathbf{p}$, respectively, which implies that standard schemes for setting the per-iteration step size in gradient-based algorithms will not be efficient. By further examining the expressions for the upper and lower bounds, we show that the direction of the gradient vector is independent of the choice of $\mathbf{p}$, and show that this allows the step size to be chosen to be as large as possible subject to the constraint that each iterate lies in $\mathcal{P}$. By combining our procedure for modifying the gradient vectors to ensure that all iterates remain in $\mathcal{P}$ and our strategy for selecting step sizes, we produce an efficient algorithm for computing optimizing the upper and lower bounds over $\mathbf{p} \in \mathcal{P}$. Pseudocode for this algorithm is presented in Algorithm 10, and additional details and proofs justifying the algorithm are provided in Appendix A.

4.3 Robustness Bounds for Alternative Divergence Measures

One concern with the robustness bounds presented in Theorem 4.1.1 are that they depend on a bound on the KLD between Q and P, which in many practical cases may be unknown or difficult to quantify. In some cases, it may be more natural to use other distances (or divergences) to describe the types of variations for which robustness is desired.

As a practical example, consider the task of deciding whether or not to approve a loan application, with the goal of being robust to differences between the training distribution and the distribution when the model is deployed. Compared to the statement, "the model will be safe on all distributions that have at most ϵ KLD with respect to the training distribution", it might be more helpful to provide guarantees such as "the model will be safe as long as, compared to the training data, real-world applicants are no more than twice as likely to default on their loan." In this section, we show that alternative definitions of variation can be converted into corresponding bounds on KLD for use in Algorithm 8.

4.3.1 Robustness to Variation in the Relative Frequency of Any Event

Having provided algorithms that provide safety guarantees that are robust to general distribution shift, we finally address the concern that, for many practitioners that are not well-versed in machine learning or statistics, it might be challenging to specify reasonable bounds on the KLD between the training and testing distributions. If the user is unable to easily derive reasonable assumptions on this term, then the algorithms presented above are of limited practical use. Therefore, in this section we show that it is possible to relate the KLD to other divergence measures that are potentially more intuitive for particular applications.

For example, suppose a user seeks to train a policy for guiding a robot that must interact in a factory environment. In this environment, the robot might be tasked with completing several different objectives, and the user might seek high-confidence guarantees that the policy will not cause damage to itself or its environment. If some objectives carry a higher risk of damage than others, these guarantees might be highly sensitive to changes in the frequency with which each objective is requested in the future. In this setting, it might be difficult for the user to specify a reasonable assumption on the KL divergence between the training environment and the future environment. However, the user may be able to easily make statements such as, “in the future, objective A will be requested no more than 20% as often as it was during training.”

Motivated by this example, we consider distances based on relative frequency of events occurring during training versus deployment. In particular, let $\mathcal{E} : \mathcal{Z} \rightarrow \{\texttt{True}, \texttt{False}\}$ be any Boolean-valued event. We consider a divergence that measures the maximum factor by which any event might become more probable after deployment:

$$d(Q||P) \coloneqq \max_{\mathcal{E}} \frac{\Pr_Q(\mathcal{E}(Z))}{\Pr_P(\mathcal{E}(Z))}.$$

Lemma 4.3.1 shows that if the user assumes that $d(Q||P) \leq \epsilon'$, then robustness bounds for this type of variation can be readily obtained using theorem 4.1.1 with $\epsilon = \log \epsilon'$.

Lemma 4.3.1. *Let Z be a real-valued random variable, and let P and Q be two probability distributions defined on Z. Let $d(Q||P)$ be defined as follows, where any $\mathcal{E} : \mathcal{Z} \rightarrow \{\texttt{True}, \texttt{False}\}$ defines an event based on Z:*

$$d(Q||P) \coloneqq \max_{\mathcal{E}} \frac{\Pr_Q(\mathcal{E}(Z))}{\Pr_P(\mathcal{E}(Z))}.$$

Then, it follows that:

$$d(Q||P) \leq \lambda \quad \implies \quad \textit{KLD}\,(Q||P) \leq \log \lambda.$$

Proof. First, we note that because any $\mathcal{E}$ can be associated with a subset $\mathcal{Z}_{\mathcal{E}} \subseteq \mathcal{Z}$ such that $\mathcal{E}(z) = \mathbb{I}(z \in \mathcal{Z}_{\mathcal{E}})$, it follows that $d(Q||P)$ is equivalent to

$$d(Q||P) \coloneqq \max_{\mathcal{Z}_{\mathcal{E}} \subseteq \mathcal{Z}} \frac{\mathrm{Pr}_Q(Z \in \mathcal{Z}_{\mathcal{E}})}{\mathrm{Pr}_P(Z \in \mathcal{Z}_{\mathcal{E}})}. \tag{4.8}$$

Next, because $\mathcal{Z}_{\mathcal{E}}$ can be *any* subset of $\mathcal{Z}$, including singleton subsets (if $\mathcal{Z}$ is discrete) or infinitesimally small subsets (if $\mathcal{Z}$ is continuous), it follows that $d(Q||P) \le \lambda$ holds if and only if $Q(z) \le \lambda P(z)$ for all $z \in \mathcal{Z}$.

Because $\log x$ is increasing in x and $d(Q||P) \le \lambda$ implies that $Q(z) \le \epsilon' P(z)$ for all z, it follows from the definition of KLD $(Q||P)$ that:

$$\begin{aligned}
\text{KLD}\,(Q||P) &= \int_{z \in \mathcal{Z}} Q(z) \log \frac{Q(z)}{P(z)} \mathrm{d}z \\
&\le \int_{z \in \mathcal{Z}} Q(z) \log \frac{\lambda P(z)}{P(z)} \mathrm{d}z \\
&= \int_{z \in \mathcal{Z}} Q(z) \log \lambda \mathrm{d}z \\
&= \left(\int_{z \in \mathcal{Z}} Q(z) \mathrm{d}z \right) \log \lambda \\
&= \log \lambda.
\end{aligned}$$

□

This result illustrates that the algorithms we propose in this chapter can often be extended to use alternative divergence measures that might be more intuitive for particular applications. As a consequence, our proposed algorithms can be extended to these settings without modification to the algorithm itself.

4.4 Evaluation and Results

To illustrate the effectiveness of our proposed robust safety guarantees, we designed experiments to evaluate the ability of our proposed algorithms to provide safety guarantees that are robust to general distribution shift. In particular, we seek to test the following hypotheses.

4.4.1 Hypotheses

In our experiments, we seek to answer three questions regarding the behavior of our proposed algorithms for overcoming general distribution shift compared to existing approaches.

Robustness of safety guarantees The primary hypothesis we test in our experiments is the claim that the safety guarantees provided by the robust algorithms we propose above will continue to hold under general distribution shift, even when the shift is selected antagonistically. In contrast, we hypothesize that existing algorithms, such as standard Seldonian algorithms, would be unsafe after deployment.

Model accuracy Next, we hypothesize that existing baseline algorithms will tend to achieve higher accuracy on average than our robust methods. This is expected because our robust algorithms are more constrained than alternative methods when selecting a model to return. However, as in our experiments presented in Chapter 3, we also seek to determine if any observed accuracy loss is due to additional constraints posed by the problem—namely the user's requirement that the algorithm behave safely even if general distribution shift occurs, or if the performance loss is due to properties of the algorithms we propose.

Data efficiency Finally, we consider the data efficiency of the algorithms we test. Specifically, we expect that our robust algorithms will be less data efficient than standard Seldonian algorithms because, for any candidate model θ, the confidence intervals on each statistic defining g are strictly larger for our robust algorithms compared to standard Seldonian algorithms due to the fact that they additionally account for general distribution shift. As a result, we expect our methods to require more data than baselines to begin returning solutions consistently, and use these experiments to assess this difference empirically.

4.4.2 Experimental Design

Similar to our experiments for evaluating safety after demographic shift in Chapter 3, we consider a classification problem in which a classifier must predict whether or not a student's *grade point average* (GPA) will be above a certain threshold, given features describing the students performance on nine exams. In addition to the student's exam grades and true GPA scores, the training algorithms are provided with the sex of each student, in order to assess whether the classifier's predictions show bias according to several standard definitions. In these experiments, we have integrated the bounds proposed in this section with the interface proposed in Chapter 2, allowing us to perform separate trials evaluating the fairness of the classifier according several commonly used definitions.

The experimental design we use in this chapter is similar to the design described in Chapter 3, with several modifications. As in our the demographic shift experiments, we simulate the impact of general distribution shift by reasmpling observations from a single, fixed population. However, since these experiments do not assume that the distribution shift is captured by a well-defined demographic, we do not use information about the race of each student in these experiments. Using this methodology, we are able to control the exact nature of the distribution shift, which is generally unknown in real-world applications, and able to compute exact values for accuracy and safety.

At a high level, each experiment is based on a distinct definition of fairness, and a fixed constraint on demographic shift. On each trial of a given experiment, we randomly sample a set of observations uniformly from a fixed population, which is then used as input to each training algorithm that we evaluate. Once each training algorithm has produced a trained model given the training dataset, we then antagonistically compute a new distribution over the population for

Algorithm 11 GeneralDistributionShiftTrial($\mathcal{P}$)

```
Specify safety constraints                                              {Sec. 2.5.1}
Specify distribution shift                                              {Sec. 2.5.1}
Uniformly sample $D_0$ from $\mathcal{P}$
for each training algorithm, $a$ do
    $\theta^*_a \leftarrow a(D_0)$
    Find a distribution $Q$ to maximize $g'(\theta^*_a)$ subject to D.S. assumptions    {Sec. 3.4.5}
    Record values for accuracy and $g'(\theta^*_a)$ under $Q$
end for
```

each model, which is carefully selected to maximize the prevalence of unfair behavior for that model while not violating the initial constraint on distribution shift. Finally, we leverage the oracle knowledge of the population and the distribution-shifted distribution to compute exact values for accuracy and safety for each model. This process is then repeated multiple times using a new randomly sampled training set, producing multiple independent samples describing the effectiveness of each algorithms for that configuration.

To provide a thorough evaluation, we evaluate our proposed algorithms, standard Seldonian algorithms, and safety-agnostic baselines. Pseudocode for a single trial of our experiments is given in Algorithm 11. In the following sections, we describe our methodology in more detail, and provide specific experimental details.

4.4.2.1 Problem Statement

In our experiments, we consider the binary classification task of predicting whether or not a student's GPA is above a certain threshold, while avoiding discriminatory behavior based on sex. Furthermore, we allow the distribution of the data observations to shift after training, subject to an initial constraint on the KLD. In contrast to our demographic shift experiments in Chapter 3, this form of distribution shift can potentially amplify the effects of discriminatory behaviors for *any* subgroup of the population.

Formally, we assume a training data observation consists of a tuple, (X, Y, S), where X is a vector of predictive features, $Y \in \{0, 1\}$ is a binary label representing whether or not the student's

GPA was above 3.0, the demographic attribute, and S describes the sex of the student. We note that S is not directly used for prediction, and is not assumed to be available once the model is deployed.

The objective of each classification algorithm is to learn a model that achieves low classification error, which is measured using the following loss function,

$$\ell(\theta) = \mathbf{E}\left[\mathbb{I}[\theta(X) \neq Y]\right],$$

where $\mathbb{I}[c]$ is 1 if c is `true` and 0 otherwise. We evaluate the fairness of each model based on whether or not its predictions are discriminatory with respect to the sex of the student.

4.4.2.2 Specifying User Inputs

In this section, we describe the example user inputs we used in our experiments. The precise choices made in this section do not impact the validity of our results, but were made to illustrate inputs that might be provided to our algorithms in practical settings.

First, we quantify discriminatory behavior of models based on three definitions of fairness, which were also used in the experiments to evaluate safety guarantees under demographic shift in Chapter 3. Full details for these definitions are provided in Section 2.5.1, but the specific definitions are shown in Figure 4.4.2.2 for completeness.

To specify the constraints on how much distribution shift will be applied after training, we use the results of Section 4.3 and base our constraints on the condition that the relative probability of any event before and after distribution shift should be bounded by come constant. In particular, if P is the training data distribution—that is, a uniform distribution over the observations in the population—then we constrain the deployment distribution Q, to satisfy,

$$P(Z{=}z) \leq \lambda Q(Z{=}z) \quad \forall\, z \in \mathcal{Z},$$

Demographic Parity [20, 6]:

$$g_{\text{DP}}(\theta) \coloneqq \Big| \, \mathbf{E}[\theta(X)|S{=}\texttt{female}] - \mathbf{E}[\theta(X)|S{=}\texttt{male}] \, \Big| - 0.1$$

Disparate Impact [27, 9, 63]:

$$g_{\text{DI}}(\theta) \coloneqq 0.8 - \min \left\{ \frac{\mathbf{E}[\theta(X)|S{=}\texttt{female}]}{\mathbf{E}[\theta(X)|S{=}\texttt{male}]}, \frac{\mathbf{E}[\theta(X)|S{=}\texttt{male}]}{\mathbf{E}[\theta(X)|S{=}\texttt{female}]} \right\}$$

Equal Opportunity [30, 9]:

$$g_{\text{EOp}}(\theta) \coloneqq \Big| \, \mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{female}] - \mathbf{E}[\theta(X)|Y{=}0, S{=}\texttt{male}] \, \Big| - 0.1$$

Figure 4.1. Definitions of fairness used for our experiments on evaluating safety guarantees under general distribution shift. Details motivating these definitions can be found in Section 3.4.4. These definitions were selected to evaluate whether the algorithms proposed in this chapter were able to provide safety guarantees under general distribution shift for a variety of practical definitions. In our experiments, these definitions were specified as text input, and were parsed and bounded using the interface proposed in Chapter 2.

which corresponds to the constraint that $\text{KLD}\left(Q||P\right) \leq \log\lambda$. In particular, we set $\lambda = 1.2$ for each experiment, which corresponds to the requirement that

$$\text{KLD}\left(Q||P\right) \leq 0.182.$$

4.4.2.3 Simulating and Evaluating the Impact of General Distribution Shift

To simulate the impact of general distribution shift, we require a procedure for generating Q that satisfies $\text{KLD}\left(Q||P\right) \leq \epsilon$, where P is the uniform distribution over the observations in the population, $\mathcal{D}_{pop}$. Fortunately, this process is simplified by the fact that, in our experiments, we apply the alternative divergence measure provided in Section 4.3, which states that Q must instead satisfy $Q(z) \leq \lambda P(z)$ for all observations $z \in \mathcal{Z}$. Consequently, Q can be optimized to maximize $g'(\theta)$ for a model θ using standard constrained optimization methods, where the constraints on Q are linear. This simplifies the optimization of Q compared to optimizing $g'(\theta)$ under the combination of the linear constraints defining the simplex and the nonlinear constraint defined by $\text{KLD}\left(Q||P\right) \leq \epsilon$.

Once Q has been computed, it it used to compute exact values for various statistics of interest during evaluation, such as the expected classification accuracy or the value of $g'(\theta)$ for any model, θ, in a manner similar to the procedure outlined in Section 3.4.5. For example, consider estimating the post-shift classification accuracy of a model, θ, given by $\mathbf{E}_Q[\mathbb{I}[\theta(X){=}Y]]$. If $\bar{\mathcal{D}}_{pop}$ denotes the set of unique observations in $\mathcal{D}_{pop}$, then we have

$$\mathbf{E}_Q[\mathbb{I}[\theta(X){=}Y]] \quad = \sum_{(x,y,s)\in\bar{\mathcal{D}}_{pop}} \mathbb{I}[\theta(x){=}y]Q(X{=}x, Y{=}y, S{=}s).$$

Analogous expressions can be used to compute exact values for post-shift model accuracy, as well as the value of $g'(\theta)$ for any model θ.

4.5 Results

In our experiments, we evaluated four training algorithms to assess our test our hypotheses. As in our experiments in Chapter 3, these algorithms were configured to produce linear models on the features, X, to ensure that no algorithm had an advantage over the others due to using more complex, nonlinear models to generate predictions. First, we include results for the algorithms proposed in this chapter. We evaluated two versions of this algorithm, which were based on Hoeffding's inequality and based on inversion of the Student's t-test, respectively. These are denoted by *Seldonian*$_{GDS}$ and *quasi-Seldonian*$_{GDS}$, and shown in green in our figures. Then, to assess whether or not our algorithms would be more effective under distribution shift than prior Seldonian algorithms, we include results for two comparable standard Seldonian algorithms. In particular, we selected Seldonian algorithms based on Hoeffding's inequality and based on inversion of the Student's t-test, which are referred to as *Seldonian* and *quasi-Seldonian* in our figures and shown in blue [54]. Finally, to assess whether our algorithms were effective at finding models that performed well while satisfying safety constraints that must hold after general distribution shift occurs, we also compared the accuracy of models trained using our algorithms to the best-case accuracy that could be found under these constraints. More details describing our procedure for estimating these bast-case values are provided in the next section.

4.5.1 Evaluation and Reporting

In order to assess the hypotheses mentioned above, we recorded several values produced for each trial of our experiments, using the same procedure outlined in Chapter 3. In particular, we recorded whether or not each algorithm returned a solution or `NO_SOLUTION_FOUND`. Next, we recorded whether or not that each trial produced a *safety failure* during training and after deployment. In our experiments, a safety failure during training occurs when an algorithm returns a model θ where $g(\theta) > 0$, and a safety failure during deployment occurs when $g'(\theta) > 0$. Finally, we record the classification accuracy of the trained models and the occurrence of safety failures when evaluated on the demographic-shifted data distribution. However, to determine if the cause of any accuracy loss when using our proposed algorithms was due to constraints posed by the user's requirements or shortcomings of our algorithms, we used a similar procedure as described in Chapter 3 and estimated the best-case accuracy of linear models under various constraints. Specifically, we computed best-case values of accuracy when no safety constraints were applied, when the models were constrained to be safe on the training distribution, and when the models were constrained to be safe after general distribution shift occurs. While we originally computed these values using a brute force random search over the set of all linear classifiers using oracle knowledge, we found that this search procedure was unable to find models with accuracy that matched those returned by standard Seldonian algorithms. This suggests that among the set of all linear models, there is a small set of linear models that perform significantly better than most other models, and are highly improbable to find using random search. As a consequence, we estimated the best-case accuracy values by starting with a model returned by a standard Seldonian algorithm, and fine-tuning the accuracy using *covariance matrix adaptation evolution strategy* (CMA-ES) [29]. These values therefore may not represent the true best-case accuracy under each set of constraints, but serves as a reasonable lower bound to assess our hypothesis.

We evaluate trials using various amounts of training data to assess the data efficiency of each algorithm. For each configuration, we conduct several randomized trials. Using this data, we report average values for *acceptance rate*, the rate with which each algorithm outputs a model, as well as average values for accuracy and failure probability for both the pre- and post-shift distributions. In addition, we show standard error for these quantities.

Concretely, for each experiment, we report five plots. First, we show the acceptance rate of each algorithm as a function of the size of the training dataset. Then, we provide two rows of plots, featuring the results on the training and deployment distributions, respectively. In each row, we first show the average accuracy of the models produced by each algorithm for various training dataset sizes. In our accuracy plots, the best-case accuracy we computed without safety constraints

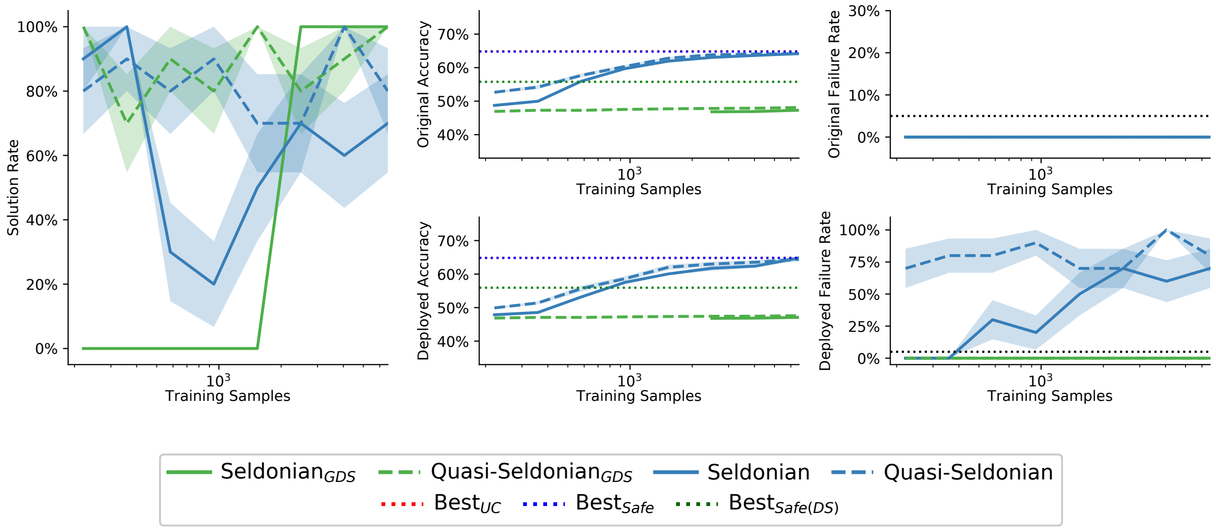

Figure 4.2. Results for experiments enforcing safety constraints based on the principle of disparate impact to preclude discrimination based on student sex when general distribution shift occurs after deployment. The rightmost column of plots displays the frequency with which each algorithm returns a solution that is unsafe before and after deployment, and demonstrates that the algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after demographic shift, whereas prior Seldonian algorithms (blue) do not. However, empirically, these added safety benefits come at the cost of accuracy (shown in the middle column of plots) and data efficiency (shown in the leftmost plot). Nonetheless, these results shown that for safety-critical applications for which ensuring safety after deployment is the primary requirement, our algorithms are effective.

is shown using a red, dashed horizontal line, while the best-case accuracy obtained under safety constraints on the training distribution and safety constraints with demographic shift are shown by blue and green horizontal dashed lines, respectively. Next, we show the failure rate for each algorithm as the size of the training dataset increases. In all figures, we show standard error for each statistic using a shaded region around the curve that delineates the average values.

4.5.2 Results

The results of our experiments using each definition of fairness are shown in Figures 4.2, 4.3, and 4.4.

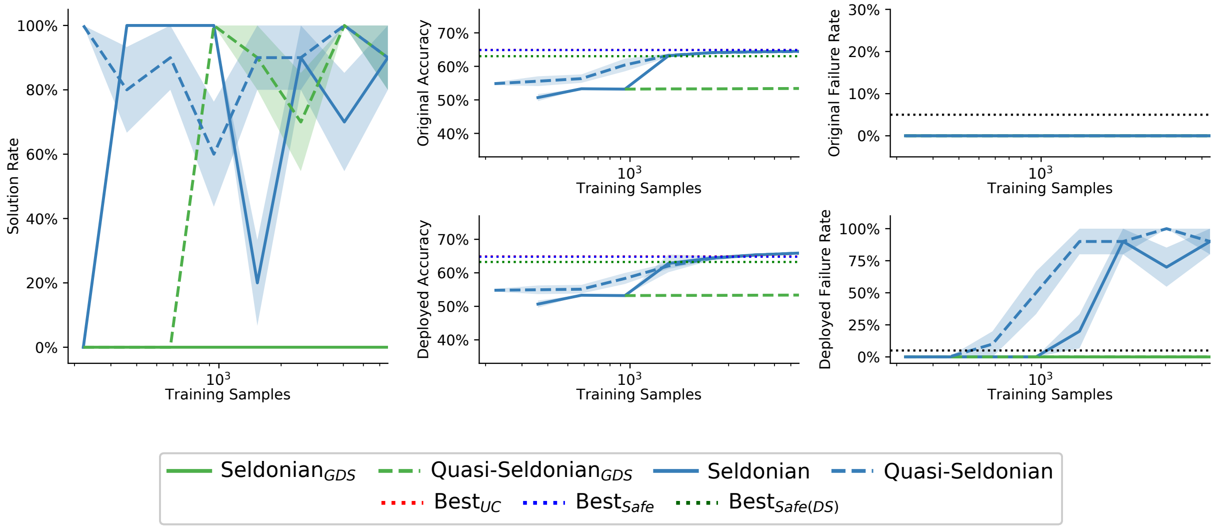

Figure 4.3. Results for experiments enforcing safety constraints based on the principle of demographic parity to preclude discrimination based on student sex when general distribution shift occurs after deployment. These results demonstrate a similar pattern as shown in Figure 4.2: the algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after deployment, while prior Seldonian algorithms (blue) do not. These results demonstrate that our proposed algorithms are effective solutions in safety-critical applications that are subject to general distribution shift, and demonstrate that these benefits are consistent for a variety of practical safety definitions.

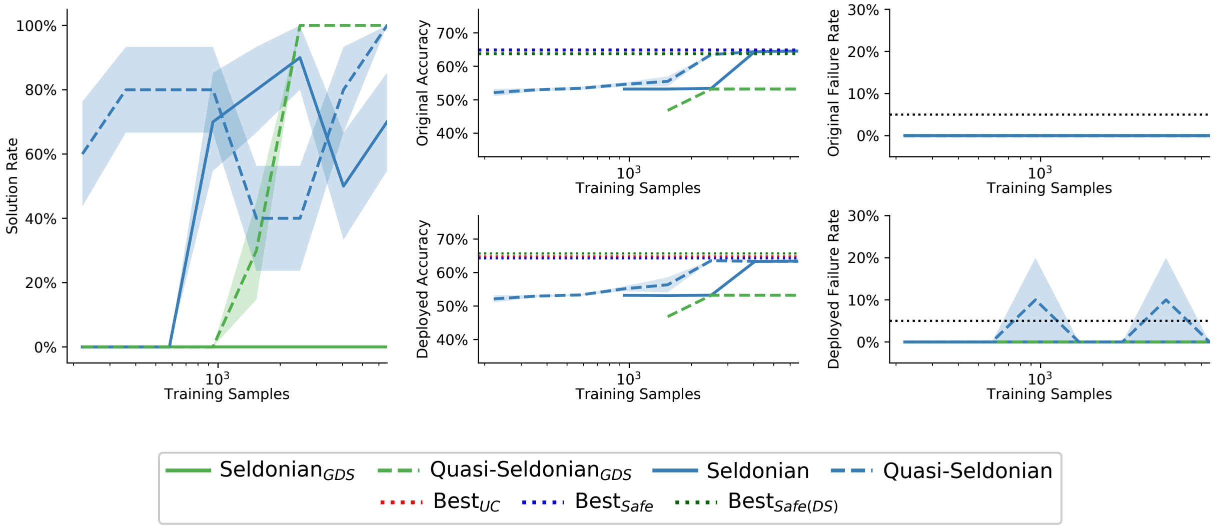

Figure 4.4. Results for experiments enforcing safety constraints based on equal opportunity to preclude discrimination based on student sex when general distribution shift occurs after deployment. These results demonstrate a similar pattern as shown in Figure 4.2: the algorithms proposed in this chapter (shown in green) provide safety guarantees that hold after deployment, while prior Seldonian algorithms (blue) do not. These results demonstrate that our proposed algorithms are effective solutions in safety-critical applications that are subject to general distribution shift, and demonstrate that these benefits are consistent for a variety of practical safety definitions.

First, we consider our hypothesis that our robust Seldonian algorithms would provide safety guarantees that hold with the required confidence even after the distribution of observations shifts, and that existing baselines would not. Examining the rightmost column of plots in each figure, we first note that as expected, both our robust methods and the standard Seldonian algorithms provided safety guarantees that are valid under the training distribution. However, the models produced by standard Seldonian algorithms exhibit intolerable unsafe behavior after general distribution shift is encountered. In contrast, the robust algorithms we propose in this chapter mitigate this effect and consistently produce models that are safe even after the distribution of observations changes. This pattern is consistent across all of our experiments, providing evidence that the methods we propose are effective at accounting overcoming the effects of distribution shift for a variety of safety definitions.

Next, we consider our hypothesis regarding the classification accuracy of our proposed methods. Examining the middle column of plots, we find that standard Seldonian algorithms produce the highest classification accuracy on average in our experiments, both before and after model deployment. In addition, the classification accuracy of these methods increases as more data is supplied, and flatten out as they approach the best-case accuracy of linear models under safety constraints. This can be explained by noting that Seldonian algorithms can be viewed as being based on constrained optimization, where the constraint set is defined by some true constraint set defined by $g(\theta) \leq 0$ that is further shrunk due to uncertainty that arises when estimating $g(\theta)$ using finite data. As a result, the optimization problems for smaller training dataset sizes are considerably more constrained than those that have access to more training data, and are therefore less capable of returning performant models.

On the other hand, we find that our robust Seldonian algorithms produce models with lower classification accuracy on average than those produced by standard Seldonian algorithms. Furthermore, our proposed algorithms consistently fail to produce models that achieve accuracy close to the best-case accuracy of models that satisfy safety constraints that must hold under distribution shift. In particular, is is fascinating to note that, despite the intuition that the set of models that satify safety constraints under distribution shift would be smaller than the set of models that satisfy safety constraints on the training distribution alone, we find that in all but one of our experiments, the difference in best-case accuracy of models under these constraints was extremely small. However, while models exist that are safe under distribution shift and achieve high accuracy, our algorithms were unable to return these models. We hypothesize that this shortcoming is due to the fact that the best-case accuracy values are computed using oracle knowledge of the data distribution during training and after distribution shift, while our proposed algorithms are evaluated using finite amounts of training data. Consequently, if high-accuracy models result in $g(\theta)$ close to zero, it may be impossible for our algorithms to establish the required confidence that these models are safe without extremely large amounts of training data. When applied to larger amounts of training data, we hypothesize that the confidence intervals used by our robust algorithms would shrink to be sufficiently small to return these models.

Lastly, we consider the data efficiency of the algorithms we evaluated, particularly with regard to the frequency with which each algorithm is able to return a solution given various amounts of training data. We find that standard Seldonian algorithms are more data efficient than our robust methods. This is indicated by the observation that, in the leftmost plot of each figure, the standard Seldonian algorithms begin consistently returning solutions for smaller input dataset sizes than the robust algorithms.

Summarizing these results, we find that the robust Seldonian algorithms we proposed in this chapter successfully provide guarantees that continue to hold under general distribution shift, although they exhibit certain drawbacks, as expected. In order to ensure that the safety guarantees provided are robust to distribution shift, these algorithms require more data to achieve the same level of accuracy and solution rate that are observed for standard Seldonian algorithms. However, these drawbacks are a natural consequence of the fact that the robust algorithms must solve more heavily constrained optimization problems than baselines. In addition, we find that the increase in the amount of data required by our methods is not prohibitive in these experiments, indicating that such methods can be effective tools for ensuring that deployed models will continue to be safe as the environment they are deployed into changes over time.

4.6 Limitations and Future Work

While the algorithms proposed in this chapter are effective at providing safety guarantees that hold under general distribution shift, there are several ways that these algorithms can be improved.

First, the assumption that the user is able to specify an upper bound on the KLD between the training distribution and the deployment distribution is limiting. First, the KLD itself may not be an intuitive divergence measure for many users, especially those that are non-experts in machine learning. Therefore, one direction of future research might investigate strategies for computing high-confidence upper bounds on $g(\theta)$ that hold under distribution shift and are based on other divergence measures. In some cases, this can be accomplished by showing a relation between a more intuitive divergence measure and the KLD, as demonstrated in Section 4.3. However, in other cases this relationship might introduce significant looseness in the resulting bounds on post-deployment safety. For example, suppose the user would like to predict a binary label associated with a vector of real-valued predictive features, $X \in \mathcal{R}^d$. If the user knows that the shape of the probability distribution of X will not change after deployment, but the mean of the distribution might shift, then a more intuitive divergence measure between the training and deployment distributions might be the *earth mover's distance* (EMD) [40]. However, it is not straightforward to convert an upper bound on EMD into an upper bound on KLD without introducing significant looseness in the bounds proposed in this chapter. Therefore, it is desirable to devise bounds similar to those presented in this chapter for divergences such as EMD, which might be more intuitive in many problem settings.

Next, it is possible that the high-confidence upper bounds presented in this chapter can be computed more efficiently. In particular, when optimizing the upper and lower bounds on post-deployment safety described in Section 4.2.3.2, we showed that gradient of the upper and lower bounds with respect to the choice of training distribution, $\mathbf{p}$, points in the same direction regardless of the choice of $\mathbf{p}$. This implies that this optimization might be solvable using constrained linear programming approaches, which may be more efficient than the gradient-based approach we propose.

Finally, we note that it may be possible to improve the accuracy of models returned by our proposed algorithms. As shown in our experimental results, our algorithms were unable to return models that achieved accuracy close to the best-case accuracy that could be found using oracle knowledge. While this is likely due to uncertainty that arises when evaluating our algorithms using finite data, we hypothesize that there are ways to improve these results. First, while the strategy used for candidate selection does not impact the validity of safety guarantees under distribution shift and were therefore not emphasized in this chapter, it is possible that enhancements to this component of our safe algorithms could improve the accuracy of the models produced. Second, it is likely that using tighter confidence intervals than those based on Hoeffding's inequality or inversion of the Student's t-test would improve result in tighter upper bounds on $g(\theta)$, and consequently allow our algorithms to return models that achieve higher accuracy.

CHAPTER 5
CONCLUSION

In this dissertation, we addressed several challenges that have prevented existing safe machine learning algorithms from being effective in many practical, but safety-critical, applications. In particular, we proposed algorithms that simultaneously 1) provide high-confidence guarantees that the models they produce will be safe, 2) do not require extensive data analysis or experience in machine learning to use, 3) are capable of enforcing a wide range of practical safety definitions, and 4) provide safety guarantees that hold even when the data used for training is not representative of what will be encountered once a model is deployed. Compared to recent work proposing algorithms that are empirically safe in some settings, algorithms that provide safety guarantees for select definitions of undesirable behavior, and even algorithms that are designed as practical options for ensuring safety [54], the algorithms proposed in this dissertation present significant advancements towards the goal of ensuring that models trained using machine learning are reliably safe in these practical settings. Specifically, we identified several shortcomings of existing safe machine learning algorithms that prevent them from being used in practical settings, and presented algorithms that overcome these problems.

First, we presented algorithms designed according to the Seldonian machine learning framework that are able to enforce complex, real-world definitions of safety. To do so, we proposed a new mathematical formulation for how safety is defined by the user, as well as algorithms that enforce these definitions using only text input from the user. This presents a significant advancement over existing safe machine learning algorithms, which are often either designed to enforce particular definitions of safety, particular classes of definitions [1], or otherwise make strong assumptions that limit their ability to enforce definitions that are encountered in real-world applications [57, 54]. Our algorithms are based on two steps. First, the user's input text is parsed into a representation that captures the functional structure of their definition, using a pair of grammars we introduce in Chapter 2. Next, this representation is used to recursively propagate confidence intervals on each uncertain term in the safety definition, eventually producing a high-confidence upper bound that can be used to design Seldonian algorithms that are safe with high-probability. Based on the results of our experiments, this approach is effective in practice, and produces high-confidence upper bounds that hold with their required probability for several real-world definitions of safety that are unable to be estimated using existing methods.

Next, we considered the challenge presented when safety guarantees are required to hold when the environment a model is trained in does not match the environment it is deployed into. Despite recent successes in designing algorithms that provide high-confidence safety guarantees, we demonstrated that these guarantees fail to hold in this setting, both theoretically and empirically. Due to the many ways that differences between the training and deployment environments can manifest, we addressed two instances of this problem, which are distinguished by the assumptions the user is able to make about these differences when the training algorithm is applied.

In the first, we showed that if these differences can be described by a change in the distribution of a single variable that the user can describe, then strategies from importance sampling can be used to produce safety guarantees that are robust to this change. In particular, if the user is able to describe how the distribution of this variable might change after deployment, then we demonstrated how observations available during training can be reweighted to produce high-confidence safety guarantees that hold after deployment. These algorithms are particularly useful in social applications, where the probability of encountering certain demographics of individuals might change over time. In such cases, our algorithms provide safety guarantees that account for this shift, while existing safe machine learning algorithms may behave unsafely. These theoretical results were verified empirically in our experiments, which demonstrated that our algorithms are effective in this setting, while existing algorithms are not.

In the second instance, we considered the more general setting in which the user cannot identify a single variable responsible for the differences between the training and deployment distributions, and provided alternative strategies for achieving robust safety guarantees. Instead of assuming that these differences exhibit structure—such as being caused by an underlying change in a single variable—the approaches we propose in this general setting assume that the user is able to place assumptions on the "size" of the difference between the training and testing environments. In particular, if P denotes the distribution that generates observations during training, and Q denotes the distribution that will generate observations after deployment, then these approaches are based on assumptions on how much Q will differ from P when measured using the Kullback-Leibler divergence. Using this assumption, we proposed Seldonian algorithms that account for the shift between the training and deployment distributions, and provide high-confidence safety guarantees that are robust to such changes. Then, we provided experimental results that validate the theoretical guarantees of our algorithms. Specifically, we showed that previous safe algorithms are unable to provide safety guarantees that hold after deployment, whereas our proposed algorithms are able to provide such guarantees for several real-world safety definitions.

In summary, this dissertation proposed novel safe machine learning algorithms designed according to the Seldonian machine learning framework, which provide high-confidence safety guarantees in practical settings. Whereas prior approaches often require extensive data analysis to use effectively or place restrictive limitations on how safety is defined, we showed that our algorithms can enforce a wide range of practical safety definitions based on simple text input from the user. In addition, in the realistic setting in which a model is trained using data that is not representative of what will be encountered once the model is deployed, the algorithms we presented are able to provide safety guarantees that hold after deployment, while existing approaches fail to do so. Consequently, the contributions presented in this dissertation represent a significant advancement towards the goal of providing safe alternatives to existing machine learning approaches that can be used for practical practical yet safety-critical applications.

APPENDIX

APPENDIX: OPTIMIZATION OF GENERAL DISTRIBUTION SHIFT BOUNDS

A.0.1 Optimization of Bounds Over $\mathbf{p} \in \mathcal{P}$ Given Fixed c

In this section, we provide results describing how the optimization of the bounds presented in Theorem 4.2.2 over $\mathbf{p}$ can be computed efficiently. First, we show that the raw gradients can be projected to produce gradient vectors that preserve the property that the sum of the entries in $\mathbf{p}$ must sum to one. In addition, we show that the projected gradients can then be further modified to ensure, for sufficiently small step sizes, they result in a sequence of iterates that is guaranteed to be in $\mathcal{P}$. Finally, we prove two properties of the upper and lower bound functions that justify the use of gradient-based approaches, and show that the problem of determining the correct sequence of step sizes for efficient convergence can be dealt with by always choosing the largest possible step size that ensures the next iterate lies in $\mathcal{P}$.

In the rest of this section, we first provide expressions for the gradients of the upper and lower bounds as a function of $\mathbf{p}$. Because these gradients are *raw gradients* that do not account for the constraint that the sum of the elements in $\mathbf{p}$ must sum to one, we additionally provide expressions for the projected gradients that preserve this constraint. In addition, we show that the projected gradients can be further modified efficiently to ensure that they do not produce iterates that lie outside of $\mathcal{P}$. While these modified gradients form the basis for iteratively optimizing the upper and lower bounds, we show that the upper and lower bound functions are only quasiconcave and quasiconcave in $\mathbf{p}$, respectively, which implies that standard schemes for setting the per-iteration step size in gradient-based algorithms will not be efficient. By further examining the expressions for the upper and lower bounds, we show that the direction of the gradient vector is independent of the choice of $\mathbf{p}$, and show that this allows the step size to be chosen to be as large as possible subject to the constraint that each iterate lies in $\mathcal{P}$. Finally, we present pseudocode for our optimization procedure.

First, we provide expressions for the raw gradients gradients of the upper and lower bounds with respect to $\mathbf{p}$ in Lemma A.0.1.

Lemma A.0.1. *Let $\mathbf{v}$ be a vector of real values, and let $\mathbf{p}$ be a vector of probabilities satisfying $\Pr(h(Z){=}v_i) = p_i$, where $h(Z)$ is a discrete random variable. Let U and L be defined by,*

$$U_\epsilon(\mathbf{p}, c) = \mathbf{p}^\intercal\mathbf{v} + \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},h}(c)}{c}, \quad \textit{and} \quad L_\epsilon(\mathbf{p}, c) = \mathbf{p}^\intercal\mathbf{v} - \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},h}(-c)}{c}.$$

The partial derivatives of U and L with respect to $\mathbf{p}$ are given by,

$$\frac{\partial}{\partial\mathbf{p}}U_\epsilon(\mathbf{p}, c) = \frac{\exp\left(c\mathbf{v}\right)}{c\,\mathbf{p}^\intercal\exp\left(c\mathbf{v}\right)}, \quad \textit{and} \quad \frac{\partial}{\partial\mathbf{p}}L_\epsilon(\mathbf{p}, c) = -\frac{\exp\left(-c\mathbf{v}\right)}{c\,\mathbf{p}^\intercal\exp\left(-c\mathbf{v}\right)}.$$

Proof. For notational convenience, let $\tilde{M}_{\mathbf{p},h}$ denote the centered moment-generating function of $\mathbf{p}$ given h, which is simply the natural exponential of the centered CGF, $\tilde{\Lambda}_{\mathbf{p},h}(c) = \log \tilde{M}_{\mathbf{p},h}(c)$.

To derive $\frac{\partial U_\epsilon(\mathbf{p},c)}{\partial\mathbf{p}}$, we have,

$$\begin{aligned}
\frac{\partial U_\epsilon(\mathbf{p},c)}{\partial \mathbf{p}} &= \mathbf{v} + \frac{1}{c}\frac{\partial}{\partial \mathbf{p}}\tilde{\Lambda}_{\mathbf{p},h}(c) \\
&= \mathbf{v} + \frac{1}{c}\frac{\partial}{\partial \mathbf{p}}\log \tilde{M}_{\mathbf{p},h}(c) \\
&= \mathbf{v} + \frac{1}{c\tilde{M}_{\mathbf{p},h}(c)}\frac{\partial}{\partial \mathbf{p}}\left\{\mathbf{p}^\intercal \exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right)\right\} \\
&= \mathbf{v} + \frac{1}{c\tilde{M}_{\mathbf{p},h}(c)}\Big\{ \exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right) + \mathbf{p}^\intercal\frac{\partial}{\partial \mathbf{p}}\exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right) \Big\} \\
&= \mathbf{v} + \frac{1}{c\tilde{M}_{\mathbf{p},h}(c)}\Big\{ \exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right) + \mathbf{p}^\intercal \exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right)(-c\mathbf{v}) \Big\} \\
&= \mathbf{v} + \frac{1}{c\tilde{M}_{\mathbf{p},h}(c)}\Big\{ \exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right) - c\tilde{M}_{\mathbf{p},h}(c)\mathbf{v} \Big\} \\
&= \mathbf{v} + \frac{\exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right)}{c\tilde{M}_{\mathbf{p},h}(c)} - \mathbf{v} \\
&= \frac{\exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right)}{c\,\mathbf{p}^\intercal\exp\left(c\mathbf{v} - c\mathbf{p}^\intercal\mathbf{v}\right)} \\
&= \frac{\exp\left(c\mathbf{v}\right)}{c\,\mathbf{p}^\intercal\exp\left(c\mathbf{v}\right)}.
\end{aligned}$$

A similar derivation shows that

$$\frac{\partial L_\epsilon(\mathbf{p},c)}{\partial \mathbf{p}} = -\frac{\exp\left(-c\mathbf{v}\right)}{c\,\mathbf{p}^\intercal\exp\left(-c\mathbf{v}\right)}.$$

□

The gradients presented in Lemma A.0.1 are *raw gradients*, indicating that they do not necessarily point along the hyperplane defined by the constraint that the sum of the entries in $\mathbf{p}$ must equal one. Fortunately, it is straightforward to project these gradients onto the tangent space of this hyperplane. In particular, the set of $\mathbf{p}$ that satisfy $\mathbf{1}_k^T\mathbf{p} = 1$ forms a hyperplane with a normal vector given by $\mathbf{n} = \frac{1}{\sqrt{k}}\mathbf{1}_k$. Consequently, the gradient vectors in Lemma A.0.1 can be projected to lie along this hyperplane by pre-multiplying it by the projection matrix, $I_{k\times k} - \mathbf{n}\mathbf{n}^T$. Conveniently, $\mathbf{n}\mathbf{n}^T$ is simply the $k \times k$ matrix with each entry equal to $1/k$, so that this pre-multiplication corresponds exactly to subtracting from each entry of the gradient vector the average value of all entries in the gradient vector:

$$\bar{\mathbf{g}}_i = \mathbf{g}_i - \frac{1}{k}\sum_{j=1}^{k}\mathbf{g}_j.$$

Using this observation, it follows that the constraint that $\mathbf{1}_k^T\mathbf{p} = 1$ can be enforced during optimization by initializing the process at some feasible $\mathbf{p}_0 \in \mathcal{P}$, and following the projected gradients given below in Lemma A.0.2.

Lemma A.0.2. *Let* $\mathbf{p} \in \mathbb{R}^k$ *be constrained to satisfy* $\sum_{j=1}^{k}\mathbf{p}_j = 1$. *Then the projected gradients of the upper and lower bounds provided in Lemma A.0.1 with respect to* $\mathbf{p}$ *are given by:*

$$\begin{aligned}\bar{\mathbf{g}}_U(\mathbf{p}) &= \frac{\exp\left(c\mathbf{v}\right) - \left(\frac{1}{k}\sum_{j=1}^{k}\exp\left(c\mathbf{v}_j\right)\right)\mathbf{1}_k}{c\,\mathbf{p}^{\mathsf{T}}\exp\left(c\mathbf{v}\right)}, \quad \textit{and}\\ \bar{\mathbf{g}}_L(\mathbf{p}) &= -\frac{\exp\left(-c\mathbf{v}\right) + \left(\frac{1}{k}\sum_{j=1}^{k}\exp\left(-c\mathbf{v}_j\right)\right)\mathbf{1}_k}{c\,\mathbf{p}^{\mathsf{T}}\exp\left(-c\mathbf{v}\right)}.\end{aligned}$$

Using the projected gradients in Lemma A.0.2 ensures that the sequence of iterates produced during optimization have components that sum to one, but does not ensure that they remain within the convex subset of the k-dimensional probability simplex defined by $\mathcal{P}$. In particular, if any iterate $\mathbf{p}_t$ lies within the interior of $\mathcal{P}$, then the projected gradients in Lemma A.0.2 are acceptable, since for a sufficiently small step size α, the next iterate, $\mathbf{p}_{t+1} = \mathbf{p}_t + \alpha\bar{\mathbf{g}}(\mathbf{p}_t)$, can be made to also satisfy $\mathbf{p}_{t+t} \in \mathcal{P}$. However, if $\mathbf{p}_t$ lies on the boundary of $\mathcal{P}$, then it is possible that $\mathbf{p}_{t+1} \notin \mathcal{P}$ for any choice positive step size. Furthermore, it is important to consider this possibility since, as shown in the following lemma, the $\mathbf{p}$ that optimizes the upper or lower bound functions lies on the boundary of $\mathcal{P}$ in all but trivial cases.

Lemma A.0.3. *Let* $\mathbf{v}$ *be a vector of real values, and let* $\mathbf{p}$ *be a vector of probabilities satisfying* $\Pr(h(Z){=}v_i) = p_i$, *where* $h(Z)$ *is a discrete random variable. Let* U *and* L *be defined by,*

$$U_\epsilon(\mathbf{p}, c) = \mathbf{p}^{\mathsf{T}}\mathbf{v} + \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},h}(c)}{c}, \quad \textit{and} \quad L_\epsilon(\mathbf{p}, c) = \mathbf{p}^{\mathsf{T}}\mathbf{v} - \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},h}(-c)}{c}.$$

Finally, let $\mathcal{P}$ denote some convex subset of the k-dimensional probability simplex, denoted Δ_k.

If $\mathbf{v}$ has more than one unique value, then if either $\mathbf{p}^* \in \arg\max_{\mathbf{p}\in\mathcal{P}} U_\epsilon(\mathbf{p}, c)$ or $\mathbf{p}^* \in \arg\min_{\mathbf{p}\in\mathcal{P}} L_\epsilon(\mathbf{p}, c)$, then it follows that $\mathbf{p}^*$ lies on the boundary of $\mathcal{P}$.

Proof. We prove this result by showing that the first-order KKT conditions for optimality cannot be satisfied within the interior of $\mathcal{P}$ if $\mathbf{v}$ contains two or more unique values.

To show this for $U_\epsilon(\mathbf{p}, c)$, we first write the expression for the projected gradient of U_ϵ with respect to $\mathbf{p}$ as given in Lemma A.0.2:

$$\bar{\mathbf{g}}_U(\mathbf{p}) = \frac{\exp(c\mathbf{v}) - \left(\frac{1}{k}\sum_{j=1}^{k}\exp(c\mathbf{v}_j)\right)\mathbf{1}_k}{c\,\mathbf{p}^\intercal\exp(c\mathbf{v})}.$$

The first-order KKT conditions for optimality imply that if $\mathbf{p}^*$ lies on the interior of $\mathcal{P}$ and maximizes U_ϵ, then each entry of $\bar{\mathbf{g}}(\mathbf{p}^*)$ must be equal to zero. Setting $\bar{\mathbf{g}}(\mathbf{p}^*) = \mathbf{0}_k$ and rearranging terms, we have,

$$\begin{aligned}
&\frac{\exp(c\mathbf{v}) - \left(\frac{1}{k}\sum_{j=1}^{k}\exp(c\mathbf{v}_j)\right)\mathbf{1}_k}{c\,\mathbf{p}^\intercal\exp(c\mathbf{v})} &&= \mathbf{0}_k\\
\implies\quad &\exp(c\mathbf{v}) - \left(\frac{1}{k}\sum_{j=1}^{k}\exp(c\mathbf{v}_j)\right)\mathbf{1}_k &&= \mathbf{0}_k\\
&\qquad\implies\quad \exp(c\mathbf{v}) &&= \left(\frac{1}{k}\sum_{j=1}^{k}\exp(c\mathbf{v}_j)\right)\mathbf{1}_k.
\end{aligned}$$

As a consequence, we see that $\mathbf{p}^*$ cannot be a maximizer of U_ϵ and lie within the interior of $\mathcal{P}$ unless each entry in $\mathbf{v}$ is identical.

The proof for L_ϵ follows analogously. □

To allow optimization when an iterate $\mathbf{p}_t$ lies on the boundary of $\mathcal{P}$, we propose to first compute a set of normal vectors describing the tangent hyperplanes to $\mathcal{P}$ at $\mathbf{p}_t$. If the boundary of $\mathcal{P}$ is locally flat at $\mathbf{p}$, then there is only a single tangent hyperplane, $\mathbf{n}$, and the gradient can be modified by projecting away this dimension by pre-multiplying the gradient by $I_{k \times k} - \mathbf{n}\mathbf{n}^T$. However, for many $\mathcal{P}$, such as those proposed in Chapter 4, $\mathcal{P}$ forms a polytope within Δ_k, and $\mathbf{p}_t$ might lie on edges where two or more supporting hyperplanes meet. Because simultaneously projecting out the component of the gradient that lies orthogonal to a set of hyperplanes is difficult when the hyperplanes themselves are not orthogonal, we instead propose to modify the gradient by solving a linear-constrained quadratic program. In particular, if $\mathbf{p}_t$ lies at the intersection of d locally flat surfaces forming the boundary of $\mathcal{P}$, then we first compute the set of normal vectors to this surface at $\mathbf{p}_t$, and denote them by $\mathbf{n}_i$ for $i = 1, ..., d$. Then, we compute the modified gradient, denoted $\hat{\mathbf{g}}(\mathbf{p}_t)$ as the solution to the following linear constrained quadratic programming problem:

$$\begin{aligned} \hat{\mathbf{g}}(\mathbf{p}_t) &= \arg\min_{\mathbf{g} \in \mathbb{R}^k} \|\bar{\mathbf{g}}(\mathbf{p}_t) - \mathbf{g}\|^2 \\ \text{s.t.} \quad & \forall i \in \{1, ..., d\},\ \mathbf{g}^T \mathbf{n}_i \leq 0. \end{aligned}$$

This quadratic programming problem can be solved efficiently with standard solvers, and produces a $\hat{\mathbf{g}}(\mathbf{p}_t)$ that can be viewed as the closest vector to $\bar{\mathbf{g}}(\mathbf{p}_t)$ that does not point out of the set $\mathcal{P}$ at $\mathbf{p}_t$. Furthermore, since $\bar{\mathbf{g}}(\mathbf{p}_t)$ lies in the tangent space of $\mathbf{P}$, it follows that $\hat{\mathbf{g}}(\mathbf{p}_t)$ will as well. Finally, if $\bar{\mathbf{g}}(\mathbf{p}_t)$ points directly out of the set $\mathcal{P}$, the $\hat{\mathbf{g}}(\mathbf{p}_t)$ returned will be uniformly zero, indicating that the optimal $\mathbf{p}$ has been found.

Given expressions for the gradients of U_ϵ and L_ϵ with respect to $\mathbf{p}$, and procedures for modifying these gradients to ensure that the sequence of iterates produced during gradient-based optimization remain in $\mathcal{P}$, it is reasonable to construct such an optimization algorithm and assume that it should find the optimal values for $\mathbf{p}$ efficiently. Unfortunately, there is one last step to complete, which is necessary due to the property that the upper and lower bound functions are not concave and convex, respectively. Therefore, standard techniques for selecting step sizes at each iteration, which assume convexity or concavity, are not suitable. Fortunately, the upper and lower bound functions can be shown to be quasiconcave and quasiconvex, respectively, which implies that any local optima of these functions over $\mathbf{p} \in \mathcal{P}$ are global optimizers. Furthermore, we show that the direction of the gradient of these functions does not depend on the specific choice of $\mathbf{P}$, which implies that the problem of selecting the step size on each iteration can be avoided by always setting the step size to be the largest value that ensures that the next iterate lies in $\mathcal{P}$. Below, we prove these two results, and provide pseudocode for our procedure for optimizing the upper and lower bounds over $\mathcal{P}$.

First, we prove that the upper and lower bound functions are quasiconcave and quasiconvex in $\mathbf{p}$.

Theorem A.0.4. *Let P be a probability distribution defined on a discrete set of real values, $\mathcal{X} := \{x_i\}_{i=1}^k$, so that the centered MGF of P is defined by*

$$M_P(c) = \sum_{i=1}^{k} P_i \exp\big(c(x_i - \mathbf{E}_P[\,X\,])\big). \tag{A.1}$$

For a fixed choice of $c > 0$, the centered MGF above is quasi-concave in P.

Proof. Let P and Q be a pair of probability distributions defined on a discrete set of real values, $\mathcal{X} := \{x_i\}_{i=1}^k$. Let $M_P(1) = \sum_{i=1}^k P_i \exp\big(X - \mathbf{E}_P[\,X\,]\big)$ and $M_P(1) = \sum_{i=1}^k P_i \exp\big(X - \mathbf{E}_P[\,X\,]\big)$ be the centered MGFs of P and Q evaluated at $c = 1$. Now, define R such that $R_i = (1-\alpha)P_i + \alpha Q_i$, for $\alpha \in [0, 1]$. We have,

$$\begin{aligned} M_R(1) \;\; &= \;\; (1-\alpha)\sum_{i=1}^{k} P_i \exp\big(X - \mathbf{E}_R[\,X\,]\big) \\ &\qquad + \alpha \sum_{i=1}^{k} Q_i \exp\big(X - \mathbf{E}_R[\,X\,]\big). \end{aligned}$$

By linearity of expectation, $\mathbf{E}_R[\,X\,] = (1-\alpha)\mathbf{E}_P[\,X\,] + \alpha\mathbf{E}_Q[\,X\,]$. Rearranging terms and defining $d := \mathbf{E}_P[\,X\,] - \mathbf{E}_Q[\,X\,]$, it follows that $\mathbf{E}_R[\,X\,]$ can also be written as:

$$\begin{aligned}\mathbf{E}_R[\,X\,] &= \mathbf{E}_P[\,X\,] - \alpha d \\ \mathbf{E}_R[\,X\,] &= \mathbf{E}_Q[\,X\,] + (1{-}\alpha)d.\end{aligned}$$

Substituting these into the previous expression for $M_R(1)$, we have:

$$\begin{aligned}M_R(1) &= (1{-}\alpha)\sum_{i=1}^{k} P_i \exp\big(X - \mathbf{E}_P[\,X\,] + \alpha d\big) \\ &\quad +\alpha\sum_{i=1}^{k} Q_i \exp\big(X - \mathbf{E}_Q[\,X\,] + (\alpha{-}1)d\big) \\ &= (1{-}\alpha)\exp\big(\alpha d\big)M_P(1) \\ &\quad +\ \alpha \exp\big((\alpha{-}1)d\big)M_Q(1). \\ &\geq \big[(1{-}\alpha)\exp\big(\alpha d\big) + \alpha\exp\big((\alpha{-}1)d\big)\big]\, M_{-}(1),\end{aligned}$$

where $M_{-}(1) \coloneqq \min\{M_P(1), M_Q(1)\}$. Noting that $\exp(x) \geq 1 + x$ for all $x \in \mathbb{R}$, we have

$$\begin{aligned}M_R(1) &\geq \big[(1{-}\alpha)\big(1 + \alpha d\big) + \alpha\big(1 + (\alpha{-}1)d\big)\big]\, M_{-}(1) \\ &= \big[\big(1{+}\alpha d{-}\alpha{-}\alpha^2 d\big) + \big(\alpha{+}\alpha^2 d{-}\alpha d\big)\big]\, M_{-}(1) \\ &= M_{-}(1).\end{aligned}$$

Thus, for $c = 1$ and $R = (1{-}\alpha)P + \alpha Q$, we have shown that $M_R(c) \geq \min\{M_P(c), M_Q(c)\}$—that is, $M_P(c)$ is quasi-concave in P for $c = 1$. To extend this result for $c \neq 1$, we note that for any $c > 0$, we can define a new variable, X', to take values in the set $\mathcal{X}' = \{cx_i\}_{i=1}^{k}$. Consequently, we have $x'_i = cx_i$ and $\mathbf{E}_P[\,X'\,] = c\mathbf{E}_P[\,X\,]$. If $M'_P(1)$ is the centered MGF using X' instead of X, we have $M_P(c) = M'_P(1)$. Applying the above result, it follows that $M'_P(1)$ is quasi-concave in P, and therefore $M_P(c)$ is quasi-concave in P for all $c > 0$. □

A defining property of quasiconcave functions is that they have convex upper contour sets. As a result, if $\mathbf{p}$ achieves a particular value of the upper bound, it implies that $\mathbf{p}$ lies within some convex subset of $\mathcal{P}$ with equal or greater values. Since this applies for all $\mathbf{p} \in \mathcal{P}$, it follows that the maximizer of U_ϵ in $\mathcal{P}$ is either unique, or it lies within a convex subset of $\mathcal{P}$ that achieve identically large values. A similar argument shows that, as a quasiconvex function, the minimizer of L_ϵ is either unique, or part of a convex subset of $\mathcal{P}$ that achieves the same minimum.

Finally, since the upper and lower bound functions are not concave and convex respectively, standard approaches for selecting step sizes will not ensure efficient convergence to the global optima. In the following corollary, we show that fortunately, the precise expressions for the upper and lower bounds allow the step size at each iteration to be arbitrarily large, so that they can be selected to be as large as possible provided they do not cause the next iterate to lie outside $\mathcal{P}$.

Corollary A.0.4.1. *Let* $\mathbf{v}$ *be a vector of real values, and let* $\mathbf{p}$ *be a vector of probabilities satisfying* $\Pr(h(Z){=}v_i) = p_i$*, where* $h(Z)$ *is a discrete random variable. Let* U *and* L *be defined by,*

$$U_\epsilon(\mathbf{p}, c) = \mathbf{p}^\intercal \mathbf{v} + \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},h}(c)}{c}, \quad \textit{and} \quad L_\epsilon(\mathbf{p}, c) = \mathbf{p}^\intercal \mathbf{v} - \frac{\epsilon + \tilde{\Lambda}_{\mathbf{p},h}(-c)}{c}.$$

It follows that, at any point $\mathbf{p} \in \Delta$*, the projected gradients of* U *and* L *can be written as,*

$$\bar{\mathbf{g}}_U(\mathbf{p}) = \beta_U(\mathbf{p})\mathbf{n}_U \quad \textit{and} \quad \bar{\mathbf{g}}_L(\mathbf{p}) = \beta_L(\mathbf{p})\mathbf{n}_L,$$

where $\beta_U(\mathbf{p})$ *and* $\beta_L(\mathbf{p})$ *are non-negative scaling factors, and* $\mathbf{n}_U$ *and* $\mathbf{n}_L$ *are unit-norm vectors that do not depend on* $\mathbf{p}$.

Proof. First, we examine the projected gradient of the upper bound with respect to $\mathbf{p}$, provided in Lemma A.0.2:

$$\bar{\mathbf{g}}_U(\mathbf{p}) = \frac{\exp(c\mathbf{v}) - \left(\frac{1}{k}\sum_{j=1}^{k} \exp(c\mathbf{v}_j)\right)\mathbf{1}_k}{c\,\mathbf{p}^\intercal \exp(c\mathbf{v})}.$$

First, let $\mathbf{d}_U \coloneqq \exp(c\mathbf{v}) - \left(\frac{1}{k}\sum_{j=1}^{k} \exp(c\mathbf{v}_j)\right)\mathbf{1}_k$. Substituting $\mathbf{d}_U$ and multiplying by $\|\mathbf{d}_U\|/\|\mathbf{d}_U\|$, we have,

$$\bar{\mathbf{g}}_U(\mathbf{p}) = \left(\frac{\|\mathbf{d}_U\|}{c\,\mathbf{p}^\intercal \exp(c\mathbf{v})}\right)\frac{\mathbf{d}}{\|\mathbf{d}_U\|}.$$

Importantly, $\mathbf{d}_U$ is defined by the specific value of $c > 0$ and $\mathbf{v}$, and does not depend on $\mathbf{p}$. It follows that $\mathbf{g}_U(\mathbf{p})$ can be written as,

$$\bar{\mathbf{g}}_U(\mathbf{p}) = \beta_U(\mathbf{p})\mathbf{n}_U,$$

where the scaling factor $\beta_U(\mathbf{p})$ is given by,

$$\beta_U(\mathbf{p}) = \frac{\|\mathbf{d}_U\|}{c\,\mathbf{p}^\intercal \exp{(c\mathbf{v})}},$$

and $\mathbf{n} = \mathbf{d}_U/\|\mathbf{d}_U\|$ is a unit-norm vector that does not depend on $\mathbf{p}$.

Finally, we show that the scaling factor is non-negative. First, since the numerator of the scaling factor defines the length of a vector, it is non-negative by definition. Second, since the entries of $\mathbf{p}$ are constrained to be non-negative and sum to one, $c > 0$, and the exponential function is strictly positive, it follows that the denominator of the scaling factor is also positive. Consequently, $\beta_U(\mathbf{p}) \geq 0$.

The proof for the projected gradient of the lower bound is analogous. Lemma A.0.2 states that the projected gradient of the lower bound with respect to $\mathbf{p}$ is given by,

$$\bar{\mathbf{g}}_L = -\frac{\exp{(-c\mathbf{v})} + \left(\frac{1}{k}\sum_{j=1}^{k}\exp{(-c\mathbf{v}_j)}\right)\mathbf{1}_k}{c\,\mathbf{p}^\intercal \exp{(-c\mathbf{v})}}.$$

Defining $\mathbf{d}_L := -\exp{(c\mathbf{v})} - \left(\frac{1}{k}\sum_{j=1}^{k}\exp{(c\mathbf{v}_j)}\right)\mathbf{1}_k$, it follows that $\bar{\mathbf{g}}_L$ can be written as,

$$\bar{\mathbf{g}}_L(\mathbf{p}) = \left(\frac{\|\mathbf{d}_L\|}{c\,\mathbf{p}^\intercal \exp{(-c\mathbf{v})}}\right)\frac{\mathbf{d}_L}{\|\mathbf{d}_L\|} = \beta_L(\mathbf{p})\mathbf{n}_L,$$

where $\beta_L(\mathbf{p}) := \|\mathbf{d}_L\|/\left(c\,\mathbf{p}^\intercal \exp{(-c\mathbf{v})}\right)$ is a non-negative scaling factor and $\mathbf{n}_L := \mathbf{d}_L/\|\mathbf{d}_L\|$ is a unit-norm vector that does not depend on $\mathbf{p}$. □

Corollary A.0.4.1 is significant because it implies that, when optimizing the bounds over $\mathbf{p} \in \Delta$, the step size at each iteration can be chosen to be arbitrarily large. For example, let $\mathbf{p}_t$ be an iterate at iteration t when maximizing the upper bound, and let $\mathbf{p}_{t+1} = \mathbf{p}_t + \alpha\bar{\mathbf{g}}(\mathbf{p}_t)$ be a potential next iterate produced by following the projected gradient at $\mathbf{p}_t$ with any positive step size α that satisfies $\mathbf{p}_{t+1} \in \Delta$. Typically, the projected gradient would be recomputed at $\mathbf{p}_{1+1}$, and this process would be repeated until convergence. However, we can instead consider the impact of continuing from $\mathbf{p}_{t+1}$ using the old gradient, $\bar{\mathbf{g}}(\mathbf{p}_t)$. The directional derivative of the upper bound at $\mathbf{p}_{t+1}$ along the direction $\bar{\mathbf{g}}(\mathbf{p}_t)$ is given by,

$$\bar{\mathbf{g}}(\mathbf{p}_t)^T\bar{\mathbf{g}}(\mathbf{p}_{t+1}) = \beta_U(\mathbf{p}_t)\beta_U(\mathbf{p}_{t+1})\mathbf{n}_U^T\mathbf{n}_U = \beta_U(\mathbf{p}_t)\beta_U(\mathbf{p}_{t+1}) \geq 0.$$

Intuitively, this implies that if, instead of generating $\mathbf{p}_{t+1}$ using step size α, we had used a larger step size, α', the value of the upper bound at $\mathbf{p}_{t+1}$ would become larger. A similar argument shows that the decrease in the lower bound between iterates becomes larger when using larger step sizes. Consequently, it is possible to optimize the upper and lower bounds using the largest possible step size that satisfies $\mathbf{p}_{t+1} \in \Delta$ for each iteration, t. Because $\mathcal{P}$ is a convex set, the upper and lower bound functions have convex upper and lower contour sets, respectively, and because the values of the upper and lower bounds are guaranteed to increase and decrease, respectively, on each iteration, the sequence of iterates converges to the optima in each case. This scheme for selecting step sizes overcomes many of the difficulties that arise when optimizing general quasiconvex and quasiconcave functions, and in practice we have found that the resulting optimization algorithms require only a small number of iterations to reach convergence.

Having provided gradients, procedures for updating them to ensure that the sequence of iterates lie in $\mathcal{P}$, proofs that the local optima of the upper and lower bound functions are global optima within $\mathcal{P}$, and having derived a scheme for setting step sizes that rapidly converges to the

Algorithm 12 MaximizeOverP($c, \epsilon, \mathbf{v}, \mathbf{p}, \mathcal{P}$)

```
for $t = 1, ...$ do
    $\mathbf{g} \leftarrow \frac{\exp(c\mathbf{v})}{c\,\mathbf{p_t}^{\intercal}\exp(c\mathbf{v})}$
    $\bar{\mathbf{g}} \leftarrow \mathbf{g} - \left(\sum_{j=1}^{k} \mathbf{g}_j\right) \mathbf{1}_k$
    if $\mathbf{p}_t$ lies on a boundary of $\mathcal{P}$ then
        for each tangent hyperplane, $i$, of the boundary of $\mathcal{P}$ at $\mathbf{p}_t$ do
            $\mathbf{n}_i \leftarrow$ the normal vector to hyperplane $i$
        end for
        $\hat{\mathbf{g}} \leftarrow \arg\min_{\mathbf{g}' \in \mathbb{R}^k} \|\bar{\mathbf{g}} - \mathbf{g}'\|^2 \quad \text{s.t.} \quad \forall i \in \{1, ..., d\},\ \mathbf{g}'^T \mathbf{n}_i \leq 0.$
    else
        $\hat{\mathbf{g}} \leftarrow \bar{\mathbf{g}}$
    end if
    $\alpha^* \leftarrow \max\{\alpha > 0 \ : \ \mathbf{p}_t + \alpha\hat{\mathbf{g}} \in \mathcal{P}\}$
    if $\hat{\mathbf{g}} \approx \mathbf{0}_k$ then
        return $\mathbf{p}_t$
    else
        $\mathbf{p}_{t+1} \leftarrow \mathbf{p}_t + \alpha\hat{\mathbf{g}}$
    end if
end for
```

optima, we now provide pseudocode for optimizing the upper and lower bounds over $\mathbf{p} \in \mathcal{P}$ in Algorithm 12.

BIBLIOGRAPHY

[1] Agarwal, Alekh, Beygelzimer, Alina, Dudík, Miroslav, Langford, John, and Wallach, Hanna. A reductions approach to fair classification. In *International Conference on Machine Learning (ICML)* (Stockholm, Sweden, July 2018), vol. PMLR 80, pp. 60–69.

[2] Angwin, Julia, Larson, Jeff, Mattu, Surya, and Kirchner, Lauren. Machine bias. *ProPublica* (May 2016).

[3] Berk, Richard, Heidari, Hoda, Jabbari, Shahin, Kearns, Michael, and Roth, Aaron. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research* (2018).

[4] Boyd, Stephen, and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, 2004.

[5] Bruzzone, Lorenzo, and Marconcini, Mattia. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 5 (2009), 770–787.

[6] Calders, Toon, and Verwer, Sicco. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery 21*, 2 (2010), 277–292.

[7] Cao, Yuan, and Gu, Quanquan. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. In *Association for the Advancement of Artificial Intelligence (AAAI)* (2020), pp. 3349–3356.

[8] Chammah, Maurice. Policing the future. *The Marshall Project* (Febuary 2016).

[9] Chouldechova, Alexandra. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data 5*, 2 (2017), 153–163.

[10] Chowdhary, Kamaljit, and Dupuis, Paul. Distinguishing and integrating aleatoric and epistemic variation in uncertainty quantification. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique 47*, 3 (2013), 635–662.

[11] Cook, James. Amazon scraps ‘sexist ai’ recruiting tool that showed bias against women. *The Telegraph* (October 2018).

[12] Corbett-Davies, Sam, Pierson, Emma, Feller, Avi, Goel, Sharad, and Huq, Aziz. Algorithmic decision making and the cost of fairness. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)* (2017), pp. 797–806.

[13] Dantzig, George B, Orden, Alex, Wolfe, Philip, et al. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics 5*, 2 (1955), 183–195.

[14] Dawood, Hend. *Theories of interval arithmetic: Mathematical foundations and applications*. LAP Lambert Academic Publishing, 2011.

[15] DeStefano, Joseph, and Learned-Miller, Erik. A probabilistic upper bound on differential entropy. *arXiv preprint cs/0504091* (2005).

[16] DeVries, Terrance, and Taylor, Graham W. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865* (2018).

[17] Dudík, Miroslav, Phillips, Steven J, and Schapire, Robert E. Correcting sample selection bias in maximum entropy density estimation. In *Advances in Neural Information Processing Systems* (2006), pp. 323–330.

[18] Dupuis, Paul, Katsoulakis, Markos A, Pantazis, Yannis, and Plechác, Petr. Path-space information bounds for uncertainty quantification and sensitivity analysis of stochastic dynamics. *SIAM/ASA Journal on Uncertainty Quantification 4*, 1 (2016), 80–111.

[19] Dwork, Cynthia, Feldman, Vitaly, Hardt, Moritz, Pitassi, Toniann, Reingold, Omer, and Roth, Aaron. The reusable holdout: Preserving validity in adaptive data analysis. *Science 349*, 6248 (2015), 636–638.

[20] Dwork, Cynthia, Hardt, Moritz, Pitassi, Toniann, Reingold, Omer, and Zemel, Richard. Fairness through awareness. In *Innovations in Theoretical Computer Science Conference* (2012), ACM, pp. 214–226.

[21] Endres, Stefan C, Sandrock, Carl, and Focke, Walter W. A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization 72*, 2 (2018), 181–217.

[22] Estrin, Daniel. Face recognition lets palestinians cross israeli checkposts fast, but raises concerns. *NPR* (August 2019).

[23] Fernando, Basura, Habrard, Amaury, Sebban, Marc, and Tuytelaars, Tinne. Subspace alignment for domain adaptation. *arXiv preprint arXiv:1409.5241* (2014).

[24] Gong, Boqing, Shi, Yuan, Sha, Fei, and Grauman, Kristen. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 2066–2073.

[25] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016.

[26] Gopalan, Raghuraman, Li, Ruonan, and Chellappa, Rama. Domain adaptation for object recognition: An unsupervised approach. In *International Conference on Computer Vision* (2011), IEEE, pp. 999–1006.

[27] Griggs v. Duke Power Co. 401 U.S. 424. `https://supreme.justia.com/cases/federal/us/401/424/`, 1971.

[28] Haddadin, S., Suppa, M., Fuchs, S., Bodenmüller, T., Albu-Schäffer, A., and Hirzinger, G. Towards the robotic co-worker. *Robotics Research 70* (2011), 261–282.

[29] Hansen, N. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, Eds. Springer, 2006, pp. 75–102.

[30] Hardt, M., Price, E., and Srebro, N. Equality of opportunity in supervised learning. In *Conference on Neural Information Processing Systems (NIPS)* (Barcelona, Spain, 2016), pp. 3323–3331.

[31] Hendrycks, Dan, and Gimpel, Kevin. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[32] Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*, 301 (1963), 13–30.

[33] Huang, Jiayuan, Gretton, Arthur, Borgwardt, Karsten, Schölkopf, Bernhard, and Smola, Alex. Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems 19* (2006), 601–608.

[34] Hwang, Cheng-Neng, Yang, Joe-Ming, and Chiang, Chung-Yen. The design of fuzzy collision-avoidance expert system implemented by h-autopilot. *Journal of Marine Science and Technology 9*, 1 (2001), 25–37.

[35] Kearon, J. Computer program for collision avoidance and track keeping. In *Proceedings of the International Conference on Mathematics Aspects of Marine Traffic. London* (1977), pp. 229–242.

[36] Kim, J., Kim, H., Lakshmanan, K., and Rajkumar, R. Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car. In *International Conference on Cyber-Physical Systems* (April 2013), pp. 31–40.

[37] Kleinberg, Jon M., Mullainathan, Sendhil, and Raghavan, Manish. Inherent trade-offs in the fair determination of risk scores. In *Innovations in Theoretical Computer Science Conference (ITCS)* (Berkeley, CA, USA, September 2017), vol. 67, pp. 43:1–43:23.

[38] Kouw, Wouter M, and Loog, Marco. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806* (2018).

[39] Kusner, Matt J., Loftus, Joshua R., Russell, Chris, and Silva, Ricardo. Counterfactual fairness. In *Conference on Neural Information Processing Systems (NIPS)* (Long Beach, CA, USA, Dec. 2017).

[40] Levina, Elizaveta, and Bickel, Peter. The earth mover's distance is the mallows distance: Some insights from statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001* (2001), vol. 2, IEEE, pp. 251–256.

[41] Li, Jing, and Xiu, Dongbin. Computation of failure probability subject to epistemic uncertainty. *SIAM Journal on Scientific Computing 34*, 6 (2012), A2946–A2964.

[42] Lipton, Zachary C, Wang, Yu-Xiang, and Smola, Alex. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916* (2018).

[43] MacKay, David JC, and Mac Kay, David JC. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[44] Metevier, Blossom, Giguere, Stephen, Brockman, Sarah, Kobren, Ari, Brun, Yuriy, Brunskill, Emma, and Thomas, Philip S. Offline contextual bandits with high probability fairness guarantees. In *Advances in Neural Information Processing Systems* (2019), pp. 14922–14933.

[45] Mohseni, Sina, Pitale, Mandar, Singh, Vasu, and Wang, Zhangyang. Practical solutions for machine learning safety in autonomous vehicles. *arXiv preprint arXiv:1912.09630* (2019).

[46] Ortega, Pedro A, Maini, Vishal, and Team, DeepMind Safety. Building safe artificial intelligence: specification, robustness, and assurance. *DeepMind Safety Research Blog* (2018).

[47] Pan, Sinno Jialin, and Yang, Qiang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering 22*, 10 (2009), 1345–1359.

[48] Pinelis, Iosif. Hoeffding's inequality for sums of pairs of random variables. MathOverflow, 2016. URL: http://mathoverflow.net/q/245604 (version: 2016-08-01).

[49] Simoiu, Camelia, Corbett-Davies, Sam, and Goel, Sharad. The problem of infra-marginality in outcome tests for discrimination. *The Annals of Applied Statistics 11*, 3 (2017), 1193–1216.

[50] Student. The probable error of a mean. *Biometrika* (1908), 1–25.

[51] Thapa, D., Jung, I., and Wang, G. Agent based decision support system using reinforcement learning under emergency circumstances. *Advances in Natural Computation 3610* (2005), 888–892.

[52] Thomas, Philip, and Learned-Miller, Erik. Concentration inequalities for conditional value at risk. In *International Conference on Machine Learning* (2019), PMLR, pp. 6225–6233.

[53] Thomas, Philip, Theocharous, Georgios, and Ghavamzadeh, Mohammad. High confidence policy improvement. In *International Conference on Machine Learning* (2015), pp. 2380–2388.

[54] Thomas, Philip S, da Silva, Bruno Castro, Barto, Andrew G, Giguere, Stephen, Brun, Yuriy, and Brunskill, Emma. Preventing undesirable behavior of intelligent machines. *Science 366*, 6468 (2019), 999–1004.

[55] Valkonen, Janne, Koskimies, Matti, Björkman, Kim, Heljanko, Keijo, Niemelä, Ilkka, and Hämäläinen, Jari J. Formal verification of safety automation logic designs. *Automaatio XVIII* (2009), 139.

[56] Varshney, Kush R. Engineering safety in machine learning. In *2016 Information Theory and Applications Workshop (ITA)* (2016), IEEE, pp. 1–5.

[57] Verma, Sahil, and Rubin, Julia. Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)* (2018), IEEE, pp. 1–7.

[58] Vyas, Apoorv, Jammalamadaka, Nataraj, Zhu, Xia, Das, Dipankar, Kaul, Bharat, and Willke, Theodore L. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 550–564.

[59] Walpole, R. E., Myers, R. H., Myers, S. L., and Ye, K. *Probability & Statistics for Engineers & Scientists*, eighth ed. Prentice Hall, 2007.

[60] Weiss, Karl, Khoshgoftaar, Taghi M, and Wang, DingDing. A survey of transfer learning. *Journal of Big Data 3*, 1 (2016), 9.

[61] Zadrozny, Bianca. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the Twenty-First International Conference on Machine learning* (2004), p. 114.

[62] Zafar, Muhammad Bilal, Valera, Isabel, Gomez Rodriguez, Manuel, and Gummadi, Krishna P. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web* (2017), pp. 1171–1180.

[63] Zafar, Muhammad Bilal, Valera, Isabel, Rodriguez, Manuel Gomez, and Gummadi, Krishna P. Fairness constraints: Mechanisms for fair classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2017), pp. 797–806.

[64] Zhang, Xiang, and LeCun, Yann. Universum prescription: Regularization using unlabeled data. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[65] Zliobaite, Indre. On the relation between accuracy and fairness in binary classification. *arXiv preprint arXiv:1505.05723* (2015).

[66] Zwillinger, Daniel, and Kokoska, Stephen. *CRC standard probability and statistics tables and formulae*. CRC Press, 1999.